

# GNU/Linux Intermedio

---

*-Quinta lezione:*

**-AGGIORNAMENTO E**  
**-GESTIONE DEI DISCHI**



# Aggiornare il software

---

- Abbiamo già visto come funziona l'installazione di GNU/Linux.
- A seconda che si sia scelto di eseguire un'installazione “da rete” o da memoria di massa (cdrom e dvd essenzialmente), il software che trovate installato sul sistema può essere più o meno aggiornato.
- In ogni caso, prima o poi, ci si trova di fronte alla necessità di aggiornare il software installato, per diversi motivi:
  - Aggiornamenti di sicurezza
  - Nuove funzionalità
  - Mancata compatibilità con nuovi software

# Modalità di installazione

---

- L'installazione di software su sistemi GNU/Linux si può fare in 3 diversi modi:
  - Compilazione del sorgente
  - Installazione di un programma precompilato
  - Installazione di un pacchetto
- Compilazione del sorgente
  - Si applica praticamente solo al software opensource ed al software prodotto eventualmente da noi.
  - Il sorgente si può reperire su cdrom allegati alle riviste, o cercando su internet ( <http://www.sourceforge.net> )
  - E' necessaria la presenza sul sistema del compilatore (gcc) e spesso anche dell'interprete dei Makefile (make)

# Compilazione del sorgente

---

- Compilazione del sorgente
  - `./configure`
    - Esegue una “configurazione” del sorgente. Verifica alcune dipendenze (la presenza del compilatore, ad esempio, o di alcune particolari librerie necessarie alla compilazione). Produce un Makefile adatto al nostro sistema.
  - `make` (o `make all`)
    - Esegue materialmente la compilazione. Si basa sulle indicazioni contenute nel Makefile. L'ordine con cui i files vengono compilati è importante, e definito dallo sviluppatore nel Makefile (tramite il configure script)
  - `make install`
    - Esegue l'installazione dell'applicativo compilato

# Applicativi precompilati

---

- Installazione di un applicativo precompilato
  - Solitamente viene inserito nel file compresso anche un “installer”, un programma cioè che si occupa di eseguire l'installazione.
  - `./installer.sh`
- Il problema davanti al quale ci troviamo, utilizzando queste due tipologie di installazione, è che il software installato non è più gestito. Da nessuna parte c'è scritto che quel determinato file appartiene a quel determinato applicativo, o dove siano finiti i files.
- I problemi si pongono in fase di aggiornamento o di disinstallazione: che files elimino? Questo file posso sovrascriverlo durante l'aggiornamento?
- Risposta: utilizziamo un package manager

# I package manager

---

- I “package manager” ( o gestori di pacchetti) sono stati inventati proprio per rispondere a questa tipologia di problemi.
- Il loro funzionamento è molto semplice:
  - Il software viene suddiviso in “pacchetti”
  - Un applicativo consente di gestirne l'installazione, la disinstallazione e l'aggiornamento
- A seconda del package manager poi:
  - Il pacchetto può essere scaricato direttamente da internet
  - Verifica automatica delle dipendenze
  - Risoluzione automatica delle dipendenze
- Spesso i package manager vengono “estesi” da altre applicazioni che sopprimono a queste mancanze

# Apt-get fa scuola

---

- Il package manager piu completo, che viene spesso e volentieri preso come esempio di funzionalità potenza e flessibilità è sicuramente “apt-get”, il package manager ideato dai distributori di Debian
- apt-get in realtà è solo una parte del package manager, visto che agisce come “interfaccia” ad un altro applicativo, che si chiama dpkg, che si occupa direttamente della gestione dei pacchetti
- apt-get consente di scaricare l'elenco dei pacchetti disponibili tramite la rete (da server detti repository), di verificare la disponibilità di aggiornamenti, di scaricare ed installare o aggiornare pacchetti verificandone e risolvendone automaticamente le dipendenze, di rimuoverli, di verificarne l'integrità e addirittura di gestirne parte della configurazione.

# Le release (debian)

---

- apt-get si basa per il suo funzionamento su “elenchi di pacchetti disponibili”, reperibili in rete.
- A seconda della versione (release) installata, ci sono elenchi differenti: in alcuni, ad esempio, sono inclusi pacchetti di versioni meno recenti in quanto più stabili
  - Stable (sarge)
    - Contiene solo il software considerato stabile, e si avvantaggia del lavoro del “Security Team”
  - Testing (etch)
    - Contiene quel software che non è ancora considerato stabile, ma che è “in coda” per il passaggio
  - Unstable (sid)
    - Distribuzione di sviluppo (“Still in Development”)



# I rami (debian)

---

- I pacchetti inoltre vengono suddivisi in “rami” a seconda di alcune caratteristiche
  - Main
    - Contiene tutti i pacchetti che compongono la “main release”
  - Non-Free
    - Contiene tutti i pacchetti che hanno problemi legati al copyright (non sono considerati “liberi”) e quindi non possono essere inclusi in Main
    - Contiene ad esempio i driver proprietari di NVidia e ATI
  - Contrib
    - Sono pacchetti “liberi” ma presentano dipendenze da pacchetti “non-free”

# La source.list

---

- Gli indirizzi dei server da cui scaricare gli elenchi dei pacchetti relativi alla release ed ai rami prescelti, vengono indicati nel file `/etc/apt/source.list`
- La struttura del file è piuttosto semplice:
- **deb** `http://debian.fastweb.it/debian stable main contrib non-free`
- **deb-src** `http://debian.fastweb.it/debian stable main contrib non-free`
- Il primo campo della riga è la tipologia di pacchetti che ci aspettiamo di trovare in quell'elenco
  - deb
    - Pacchetti binari
  - deb-src
    - Pacchetti sorgente che vengono compilati da apt

# La source.list

---

- Gli indirizzi dei server da cui scaricare gli elenchi dei pacchetti relativi alla release ed ai rami prescelti, vengono indicati nel file `/etc/apt/source.list`
- La struttura del file è piuttosto semplice:
- `deb`      <http://debian.fastweb.it/debian> stable main contrib non-free
- `deb-src` <http://debian.fastweb.it/debian> stable main contrib non-free
- Il secondo campo riporta l'URI del repository in cui troveremo gli elenchi.
  - `http://*`
  - `ftp://*`
  - `file://*`

# La source.list

---

- Gli indirizzi dei server da cui scaricare gli elenchi dei pacchetti relativi alla release ed ai rami prescelti, vengono indicati nel file `/etc/apt/source.list`
- La struttura del file è piuttosto semplice:
- `deb http://debian.fastweb.it/debian stable main contrib non-free`
- `deb-src http://debian.fastweb.it/debian stable main contrib non-free`
- Il terzo campo riporta il nome della release che vogliamo utilizzare (sarge o stable, etch o testing, ...)
- Infine, troviamo l'elenco dei rami di cui vogliamo scaricare gli elenchi.

# La source.list

---

- Gli indirizzi dei server da cui scaricare gli elenchi dei pacchetti relativi alla release ed ai rami prescelti, vengono indicati nel file `/etc/apt/source.list`
- La struttura del file è piuttosto semplice:
- `deb http://debian.fastweb.it/debian stable main contrib non-free`
- `deb-src http://debian.fastweb.it/debian stable main contrib non-free`
- Il terzo campo riporta il nome della release che vogliamo utilizzare (sarge o stable, etch o testing, ...)
- Infine, troviamo l'elenco dei rami di cui vogliamo scaricare gli elenchi.

# Update

---

- Tutte le operazioni di apt, sono eseguite basandosi sull'elenco dei pacchetti scaricati
- Per scaricare gli elenchi dei pacchetti, si digita il comando
- `apt-get update`
- Apt si collega ad ognuno dei server elencati nella `source.list`, e scarica l'elenco
- `${REPOS}/dists/${RELEASE}/${ARCH}/binary-i386/Packages.gz`
- Una volta scaricato ed estratto, questo file contiene alcuni dati su tutti i pacchetti disponibili in quel ramo:
  - Checksum
  - Dipendenze
  - Indirizzo del pacchetto vero e proprio

# Aggiornamento

---

- Una volta che abbiamo un elenco aggiornato dei pacchetti, possiamo scaricare ed installare eventuali aggiornamenti disponibili:
- `apt-get dist-upgrade`
- Per aggiornare un pacchetto alla volta, possiamo usare il comando
- `apt-get upgrade nomepacchetto`
- La differenza tra `dist-upgrade` e `upgrade`, è che il primo tiene conto anche delle eventuali nuove “interrelazioni” (ad esempio le dipendenze) tra i pacchetti, ed è quindi più indicato quando si esegue un aggiornamento “system wide”.

# Installazione/rimozione

---

- La mera installazione di un nuovo pacchetto, la si fa digitando il comando
- `apt-get install nomepacchetto`
- Questo risolve le dipendenze ed installa tutto quello che è necessario, eventualmente suggerendoci pacchetti che non sono dipendenze ma che potrebbero essere importanti per il nostro utilizzo del pacchetto in questione.
- E' possibile reinstallare un pacchetto già installato (ad esempio perchè danneggiato) usando l'opzione `--reinstall`
- La rimozione del pacchetto invece, si esegue tramite un
- `apt-get remove nomepacchetto`
- Apt rimuove anche tutti i pacchetti che dipendono da



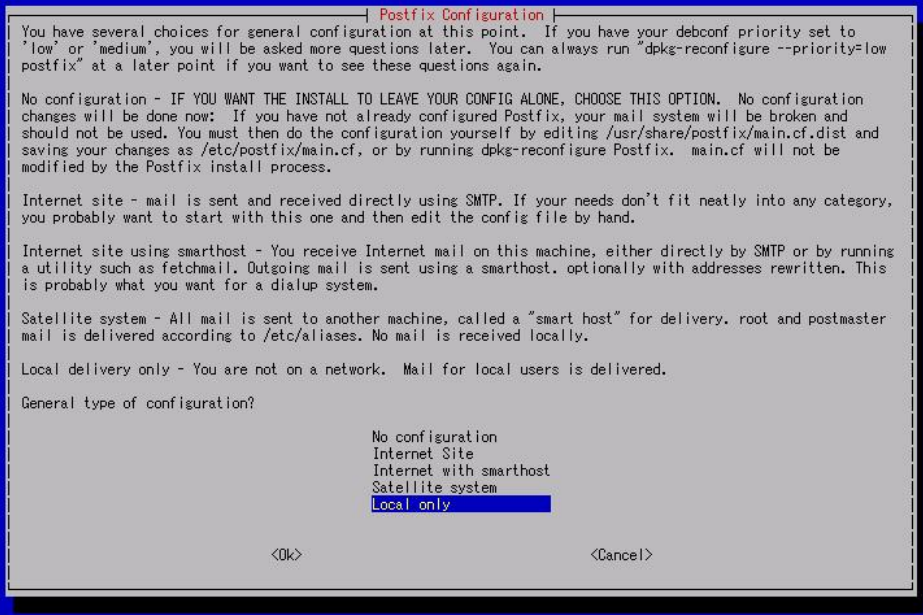
# La cache

---

- Apt mantiene sul sistema locale una “cache” dei pacchetti installati, in modo da poterli eventualmente reinstallare senza doverli scaricare nuovamente.
- Visto che solitamente gli utenti non si divertono a scaricare, eliminare e reinstallare pacchetti, nella nostra ottica questa cache fa poco più che occupare spazio di sistema.
- Per eliminare tutti i pacchetti in cache, possiamo andare a rimuoverli manualmente, oppure più semplicemente utilizzare il comando
- `apt-get clean`
- La cache si trova in `/var/cache/apt/archives/` e può tranquillamente raggiungere dimensioni davvero notevoli (pensate ai pacchetti che compongono OpenOffice.org )

# Configure

- Spesso, quando un pacchetto necessita di una configurazione al fine di funzionare, durante la fase di installazione si apre una finestra (ncurses, grafica via terminale) che pone delle domande relative alla configurazione del sistema, e genera il file di configurazione necessario.
- Spesso la scelta proposta come default è quella più adatta e meno rischiosa, ma vale ovviamente la pena leggere la domanda.
- Può essere richiamato anche in seguito, tramite l'opzione `--reconfigure`



```
Postfix Configuration
You have several choices for general configuration at this point. If you have your debconf priority set to
'low' or 'medium', you will be asked more questions later. You can always run "dpkg-reconfigure --priority=low
postfix" at a later point if you want to see these questions again.

No configuration - IF YOU WANT THE INSTALL TO LEAVE YOUR CONFIG ALONE, CHOOSE THIS OPTION. No configuration
changes will be done now: If you have not already configured Postfix, your mail system will be broken and
should not be used. You must then do the configuration yourself by editing /usr/share/postfix/main.cf.dist and
saving your changes as /etc/postfix/main.cf, or by running dpkg-reconfigure Postfix. main.cf will not be
modified by the Postfix install process.

Internet site - mail is sent and received directly using SMTP. If your needs don't fit neatly into any category,
you probably want to start with this one and then edit the config file by hand.

Internet site using smarthost - You receive Internet mail on this machine, either directly by SMTP or by running
a utility such as fetchmail. Outgoing mail is sent using a smarthost, optionally with addresses rewritten. This
is probably what you want for a dialup system.

Satellite system - All mail is sent to another machine, called a "smart host" for delivery. root and postmaster
mail is delivered according to /etc/aliases. No mail is received locally.

Local delivery only - You are not on a network. Mail for local users is delivered.

General type of configuration?

No configuration
Internet Site
Internet with smarthost
Satellite system
Local only

<Ok> <Cancel>
```

# Altre distribuzioni

---

- Spesso e volentieri, le diverse distribuzioni si avvalgono di package manager propri.
- Ubuntu utilizza apt-get, anche se sfruttando nomi di release e di repository diversi.
- ArchLinux utilizza un package manager chiamato Pacman, che offre le stesse potenzialità di apt-get.
- Fedora, Mandriva, SuSe Linux e piu in generale tutte le distribuzioni basate su pacchetti “rpm”, devono utilizzare programmi che consentano di interfacciarsi con rpm, perchè questo non consente la gestione “online” degli aggiornamenti.
- Gentoo utilizza un gestore di pacchetti molto flessibile (emerge) che solitamente viene usato per compilare pacchetto e dipendenze sul sistema locale.

# Altre distribuzioni

---

- Da qualche tempo però, Gentoo consente di fare l'installazione tramite pacchetti precompilati, e di utilizzare pacchetti precompilati anche nella manutenzione ordinaria del sistema.
- Oltretutto anche molti dei package manager delle altre distribuzioni consentono la compilazione del pacchetto e delle relative dipendenze.
- Si pone inoltre il problema di installare quei software che non sono inclusi tra quelli disponibili.
- Ogni distribuzione affronta il problema a suo modo, ma spesso la soluzione è quella di fornire un tool che consenta di creare un pacchetto personalizzato, e successivamente installarlo tramite il “package manager”

# I filesystems

---

- I dati sono un insieme di files.
- I files sono composti da bits.
- Questi bits devono essere “immagazzinati” sui supporti fisici in un ordine noto, in modo che sia poi possibile ricostruirne la struttura originale al momento della lettura.
- Questo “ordine noto” viene detto filesystem.
- Esistono naturalmente molti modelli diversi di filesystem, con caratteristiche anche estremamente diverse, a seconda delle esigenze per cui sono stati sviluppati.
- I principi base però, sono grossomodo gli stessi per tutti.
- Vediamo qualche esempio...

# Gli inode

---

- Come abbiamo già visto, un filesystem risiede all'interno di una partizione.
- Non è possibile avere più filesystem all'interno di una singola partizione fisica.
- Immaginiamo di dividere il filesystem in tanti “pezzettini”: sono gli inode ed i blocchi
  - gli inode sono le descrizioni dei files. Ogni inode è identificato da un numero (inumber), ma non presenta il nome del file (vedremo perchè).
  - Ogni inode può presentare puntatori ad altri inode (directory), o a blocchi (e quindi è un file)
  - i blocchi contengono i dati veri e propri.
- Tutto fa capo ad un “superblocco”, che contiene le informazioni generali sul filesystem.

# Struttura di un inode

---

- 

| 2     |     |
|-------|-----|
| • bin | 3   |
| boot  | 4   |
| • dev | 5   |
| etc   | 6   |
| • lib | 7   |
| mnt   | 8   |
| • ... | ... |

- 

- 

- 

- Prendiamo un inode, quale potrebbe essere la cartella /.
- Troviamo associati alcuni nomi di “files” (directory in questo caso) ai relativi inumber.

# Struttura di un inode

- 

| 2      |     |
|--------|-----|
| • bin  | 3   |
| • boot | 4   |
| • dev  | 5   |
| • etc  | 6   |
| • lib  | 7   |
| • mnt  | 8   |
| • ...  | ... |



| 3      |     |
|--------|-----|
| arch   | 100 |
| awk    | 101 |
| bash   | 102 |
| bunzip | 103 |
| bzcat  | 105 |
| bzip2  | 106 |
| ...    | ... |

- 

- 

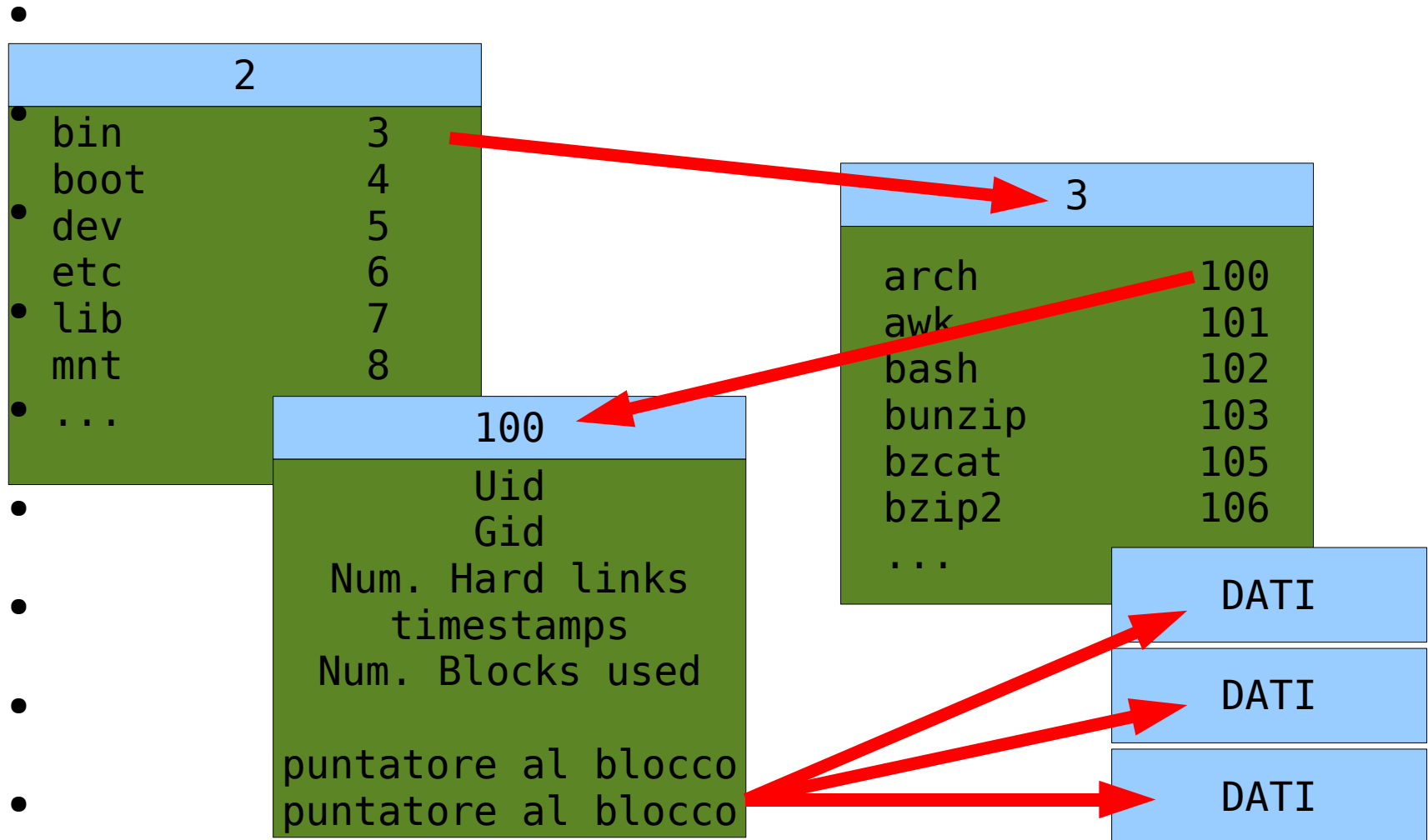
- 

- 

- Andando a prendere il contenuto della directory, troveremo qualcosa di simile.



# Struttura di un inode



- I singoli files invece, contengono alcuni dati riguardo i permessi ed i blocchi dei dati che lo compongono.

# I nomi dei files

---

- Come abbiamo detto, tra i dati scritti all'interno di un inode, non è presente il nome. Ne che si tratti di una directory, ne che si tratti di un file.
- Il nome all'inode infatti, viene assegnato dall'inode “contenitore” (la directory solitamente), come abbiamo potuto vedere.
- In questo modo, è possibile che un inode abbia due diversi nomi, all'interno di due directory magari diverse, associando a due entry lo stesso inumber.
- Sono gli “hard-links”, e possono essere fatti solo all'interno dello stesso filesystem.
- I collegamenti simbolici invece, presentano il nome ed il path del file a cui “puntano”. In questo modo è possibile farli anche su partizioni distinte.

# Proprietario e gruppo

---

- Tra le varie informazioni che invece sono presenti all'interno di ogni singolo inode, troviamo un campo “uid” un campo “gid” ed un campo “modi”
- I primi due campi contengono rispettivamente l'identificativo numerico dell'utente proprietario del file e del gruppo di riferimento.
- Il terzo invece, contiene la definizione dei permessi che devono essere applicati a quel file.
- Per osservare gli attributi di un file/cartella, è sufficiente dare il comando `ls -li nomefile`
- 
- 
- `64344 -rw-r--r-- 1 alt-os users 118509 2007-02-06 19:19 Lezione 5.odp`

# I permessi

---

- Come avevamo già accennato qualche lezione fa, i permessi sui files di Linux vengono gestiti tramite 3 privilegi che possono essere assegnati o negati
  - Lettura
  - Scrittura
  - Esecuzione
- per 3 diverse categorie di utenti
  - Proprietario
  - Membri del gruppo
  - Altri utenti
- 
- `64344 -rw-r--r-- 1 alt-os users 118509 2007-02-06 19:19 Lezione 5.odp`

# I permessi

---

- Dove troviamo il segno “-”, il permesso è stato negato, dove troviamo la relativa lettera, il permesso è concesso.
- I permessi si differenziano tra files e cartelle:
  - Mentre per i files è semplice associare lettura, scrittura ed esecuzione
  - Per le directory la solfa cambia leggermente:
    - Lettura = lista del contenuto (ls)
    - Scrittura = aggiungere e/o rimuovere (rm) files
    - Esecuzione = accesso (cd) alla directory
- I permessi “di default” di un file sono quelli che vedono come proprietario l'utente che lo ha creato, come gruppo il suo gruppo “principale”, ed i permessi derivanti dall'analisi della variabile d'ambiente UMASK.

# I permessi in ottale

---

- I permessi possono anche essere specificati in forma ottale:
  - Lettura = +4
  - Scrittura = +2
  - Esecuzione = +1
- Dalla loro somma sono possibili tutte le combinazioni:
  - 0 ---
  - 1 --x
  - 2 -w-
  - 3 -wx
  - 4 r--
  - 5 r-x
  - 6 rw-
  - 7 rwx

# UMASK

---

- La variabile d'ambiente UMASK contiene il valore in ottale dei permessi che si vogliono negare al momento della creazione di un nuovo file.
- Esempio:
  - Umask = 022
    - Permessi files =  $666 - 022 = 644$
    - Permessi directory =  $777 - 022 = 755$
  - Umask = 044
    - Permessi files =  $666 - 044 = 622$
    - Permessi directory =  $777 - 044 = 733$
  - Umask = 007
    - Permessi files =  $666 - 007 = 660$
    - Permessi directory =  $777 - 007 = 770$

# Il Journaling

---

- Una delle principali, storiche, differenze tra i vari filesystem, è sicuramente l'implementazione del Journaling.
- Questo meccanismo consente nel tenere un “diario” delle operazioni da eseguire sul filesystem, aggiornandolo prima di eseguire le operazioni richieste.
- In questo modo, nel momento in cui si dovesse verificare un problema (ad esempio lo spegnimento del sistema) durante la fase di scrittura, sarà sufficiente ripetere le operazioni non segnate come “terminate” sul diario, riportando rapidamente il filesystem ad uno stato di coerenza.
- Una scansione dell'intero filesystem viene comunque prevista di tanto in tanto per “sicurezza”.



# I filesystems di Linux

---

- Linux supporta una quantità enorme di filesystems diversi. I principali sono qui elencati:
  - Ext2
    - Filesystem storico di Unix, veloce e stabile.
  - Ext3
    - Si tratta di una versione “aggiornata” di ext2 in modo da implementare, tra l'altro, il meccanismo del Journal. Mantiene la piena compatibilità con ext2.
  - Ext4
    - In fase di sviluppo (annunciato il 10 ottobre 2006 da Andrew Morton), sarà completamente retrocompatibile con ext3, a patto di disabilitare le “extents” (una particolare metodologia di allocazione dello spazio disco

# I filesystems di Linux

---

- Linux supporta una quantità enorme di filesystems diversi. I principali sono qui elencati:
  - ReiserFS
    - E' stato il primo filesystem Journalized ad essere incluso nel kernel Linux “Vanilla”
    - Utilizzando un albero binario, è estremamente efficiente nell'allocare spazio per directory contenenti grandi quantità di piccoli files (archivi Usenet/email).
    - Nota di “colore”: il suo creatore, Hans Reiser, è stato arrestato il 10 ottobre 2006, sospettato di aver ucciso la moglie, Nina Reiser, scomparsa oltre un mese prima dopo aver lasciato i due bambini a casa del padre.
    - La cosa ha destato parecchio scalpore nel mondo dello sviluppo del software libero, dove

# I filesystems di Linux

---

- Linux supporta una quantità enorme di filesystems diversi. I principali sono qui elencati:
  - Reiser4
    - Sviluppato da Namesys in collaborazione con DARPA e Linspire, è il successore di ReiserFS (interamente riscritto).
    - Nonostante i notevoli vantaggi di velocità (dovuti all'utilizzo di “dancing trees”), non è ancora stato incluso nel kernel Linux per problemi di “stile di programmazione”.
    - L'introduzione nella release ufficiale del kernel è l'obiettivo principale di Namesys.

# I filesystems di Linux

---

- Sono inoltre supportati:
  - FAT16/32
    - Il classico filesystem di Microsoft, sia in versione 16 che in versione 32 (28) bit.
  - NTFS
    - Il filesystem Journaled il cui sviluppo era stato iniziato in collaborazione tra IBM e Microsoft (l'epoca di OS/2).
    - Il supporto è stato mantenuto “in sola lettura” per alcuni anni, poi con il kernel 2.6 è finalmente diventato utilizzabile con sicurezza anche in scrittura.
    - C'è un interessante progetto, chiamato captive-ntfs, che promette una lettura/scrittura sicura utilizzando direttamente il driver NTFS di Windows. Il problema risiede nella

# I filesystems di Linux

---

- Sono inoltre supportati:
  - ISO9660
    - Il filesystem standard utilizzato per i CD-ROM.
    - Presenta 3 livelli:
      - 1: come il DOS; supporta nomi di caratteri lunghi al massimo 8 caratteri più 3 per l'estensione, composti da lettere (maiuscole e minuscole), numeri ed underscore.
      - 2: estende a 32 caratteri la lunghezza dei nomi di files e cartelle, ma mantiene il limite di nidificazione a 8 livelli.
      - 3: consente la scrittura dei files in settori non contigui.
  - UDF
    - E' stato creato per sostituire ISO9660 che non si adattava a CD-RW e DVD. Consente nomi di

# I filesystems di Linux

---

- Sono inoltre supportati:
  - affs - Amgia Fast Filesystem
  - hpfs - High Performance Filesystem (OS/2)
  - minix - Minix Filesystem (first filesystem used by Linux)
  - NFS - Network Filesystem
  - smbfs - Samba Filesystem
  - NCPFS - Novell Filesystem
  - sysv - System V Filesystem (System V Unix variants)
  - ufs - Uniform Filesystem ( BSD, Solaris, NeXTStep)
  - xfs - Silicon Graphics' (SGI's) IRIX
  - adfs, autofs, coda, coherent, cramfs, devpts, efs, hfs, jfs, qnx4, romfs, xenix, ...

# Dischi lenti

---

- Un'altro grosso problema dei dischi fissi, è quello di essere estremamente lenti, come abbiamo piu volte ripetuto.
- Per ovviare a questo problema, GNU/Linux (ma non solo) implementa un meccanismo di “caching”, cercando di ottimizzare e ridurre al massimo il numero di letture e scritture che si devono fare fisicamente sul disco.
- Fintanto che c'è della RAM libera, il kernel ne utilizza quanta piu possibile riservandola a “cache”, e mantenendovi una copia dei files aperti e/o recentemente modificati.
- In questo modo, un file appena letto da un'applicazione, risulta immediatamente disponibile in caso di nuove richieste, senza attendere che il disco lo legga di nuovo.

# Dischi lenti

---

- Lo stesso meccanismo funziona in scrittura: le modifiche vengono apportate alle copie “in ram” (in cache) dei files, e non alle copie reali sul filesystem, finchè non si è accumulata una quantità tale di modifiche che vale la pena eseguire una scrittura e liberare la cache.
- Questo è particolarmente utile nel caso delle memorie di tipo Flash, che hanno un numero limitato di scritture possibili e vengono così ottimizzate.
- Questa operazione (sync) viene comunque effettuata ad intervalli regolari dal kernel.
- L'uso della cache migliora notevolmente le prestazioni del sistema, ma ci espone ad un rischio nel momento in cui la device viene “spenta” (o scollegata) prima che siano terminate le operazioni di sync...



# Integrità

---

- A volte è necessario eseguire un “controllo di integrità del filesystem”
- Questo è possibile tramite il comando `fsck` il quale, dopo aver rilevato la tipologia di filesystem che gli chiediamo di controllare, richiama l'apposita utility per eseguire il controllo (`fsck.${tipo}`).
- E' opportuno (a volte necessario) che il filesystem sia smontato oppure montato in sola lettura
  - `mount -f -r -o remount /`
- Un controllo “fisico” del disco e delle partizioni, invece, è possibile utilizzando il comando `badblocks`, che può ad ogni modo essere richiamato anche da `fsck` passandogli l'opzione `-c`

# Frammentazione in Windows

---

- In un ambiente Windows, la frammentazione rappresenta un grosso problema, visto che può influenzare in maniera pesantemente negativa le performance del sistema.
- Parzialmente ridotto con l'introduzione del filesystem NTFS, che sfrutta con più coscienza lo spazio-disco, l'utilizzo di uno strumento di deframmentazione è un'operazione abituale da parte di un utente mediamente evoluto dei sistemi Microsoft Windows.
- Questo è legato principalmente allo scarsissimo uso della cache da parte dei sistemi citati, e dalla poco attenta gestione dello spazio disco.
- Ma cos'è esattamente la frammentazione?

# La frammentazione interna

---

- Esistono due diversi tipi di frammentazione:
  - Frammentazione Interna
    - Ogni volta che allochiamo una serie di blocchi per memorizzare dati relativi ad un file, l'ultimo di questi blocchi non viene riempito completamente (salvo casi fortunati).
    - Tanenbaum, nel suo libro “Sistemi Operativi”, definisce questo genere di “perdita di spazio” come “Frammentazione interna”
    - Non rappresenta un grosso problema per le prestazioni del sistema, perchè una volta giunti alla fine dei dati, il file è letto e quindi la testina del disco non deve andarne a cercare altri.

# La frammentazione esterna

---

- Esistono due diversi tipi di frammentazione:
  - Frammentazione esterna
    - La frammentazione esterna invece si verifica quando i blocchi che compongono un file non vengono scritti “consecutivamente” sul disco.
    - La lettura da disco infatti è seriale, e quindi se dopo aver terminato di leggere un blocco la testina non trova quello successivo nel settore adiacente, deve andare a prendere quello successivo in un'altra area del disco, perdendo tempo prezioso.
    - Questa frammentazione è quella che penalizza maggiormente le performance del sistema, se non affiancata da un'adeguata cache su disco.

# Frammentazione in Linux

---

- Sui sistemi Linux, questo non rappresenta praticamente un problema.
- Non perchè i filesystem Linux non subiscano totalmente la frammentazione, ma perchè adottano politiche di storage pensate appositamente per ridurla al minimo.
- Ad esempio, sin dai tempi di ext2, quando un file è in crescita, una parte dello spazio disco attiguo viene riservata alla sua crescita, e viene “rilasciata” nel momento in cui il file viene chiuso, recuperando lo spazio.
- Ext3 esegue inoltre periodiche “riorganizzazioni” del disco per ridurre ulteriormente la frammentazione.
- ReiserFS può implementare un meccanismo detto “Tail Packing” per ridurre anche la frammentazione interna.