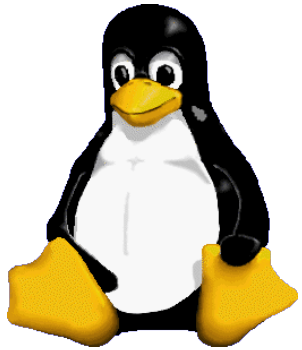
Elementi di LINUX a livello utente

Cosa è LINUX



• È un **SO** UNIX standard
• Tesi di laurea di **Linus Torwalds** all'università di Helsinki in Finlandia
• Sviluppato a partire da Minix, un sistema UNIX molto piccolo
• Nel **1991** è stata rilasciata la prima versione (0.02)
• Nel 1994 fu rilasciata la **versione 1.0 del Kernel Linux**
• Il 12 marzo 2015 - Viene rilasciata la release stabile di Linux 4.0 e … lo sviluppo di Linux continua.

• È considerato un SO eccellente, economico alternativo ad altri SO molto costosi.(i.e. UNIX, NT, W2000, …..).

Linux: le caratteristiche

Multitasking: più programmi funzionano contemporaneamente.

- **Multiuser**: più utenti nella stessa macchina contemporaneamente
- **Scalabile**: in modo testo ha pretese hardware minime
- Funziona su **diverse architetture**: PC, SPARC, Mac,…
- Affianca **diversi SO**: Windows, Solaris, MS-DOS, MacOs,…Ha infatti accesso **trasparente a partizioni MS-DOS** (o partizioni OS/2 FAT) tramite il filesystem; non è necessario nessun comando speciale per usare la partizione MS-DOS, appare come un normale filesystem Unix (eccetto per ovvie restrizioni sui nomi di file, permessi e così via).
- Gestisce **multiprocessor**
- **Supporto rete** TCP/IP, incluso ftp, telnet, NFS...
- **Modalità protetta**: ogni programma in esecuzione occupa specifici indirizzi di memoria, "protetti" dalla possibilità che altri programmi vadano ad occupare gli stessi indirizzi causando in tal modo il blocco del sistema.
- Possibilità di utilizzare **un completo ambiente grafico**, una GUI (Graphical User Interface) conosciuta con il nome di **X Windows**.
- **Ambiente di sviluppo completo** per applicazioni e programmi (i.e. il C ed il C++, il Perl il Tcl/Tk). Attualmente Linux è fortemente rivolto verso JAVA.

GNU

Il **software libero** richiede prima di tutto il sistema operativo. L'obiettivo che si prefiggeva ***Richard Stallman*** era quello di realizzare, con l'aiuto di volontari, un sistema operativo completo.

- Il **Progetto GNU** è stato lanciato nel 1984 per sviluppare un sistema operativo Unix-compatibile completo che fosse **software libero.**

- La **GPL (GNU Public Licence)** - applicazioni libere da diritti.

GPL *fa in modo che il codice sorgente rimanga libero: qualsiasi applicazione derivata da sw con licenza GPL deve essere distribuita con licenza GPL.*

- **Non obsolescenza dei programmi**: ricompilare per ogni nuovo kernel.

Linux: GNU e GPL

Il software libero

Al problema dell'ambiguità del concetto, si affiancava l'ambiguità della denominazione: in inglese, **free software** poteva essere inteso come software gratuito (***free of charge***).

Nel 1998, nasce la definizione **Open Source**, a identificare i principi secondo cui il software può essere ritenuto «libero» ma dandogli un nome inequivocabile e non modificabile (<*http://www.opensource.org*>).
Open Source, ovvero «sorgente aperto», non fa pensare alla «libertà» che invece è il motivo alla base del software libero.

La qualità «open» del sorgente («source») di un certo prodotto commerciale (proprietario) non ha nulla a che vedere con il software libero.

Vantaggi di Linux

- ***Libertà:*** codici sorgenti aperti; chiunque può utilizzarli, modificarli, etc. sempre in evoluzione
- ***Stabilità:*** Linux non si blocca, praticamente, mai. È un sistema completamente multitasking e multiutente. Se un programma si blocca è possibile terminarlo senza alterare la stabilità dell'intero sistema.
- ***Sicurezza:*** accesso ad utenti autorizzati. Diritti d'accesso differenziati per ogni utente.
- ***Trasparenza:*** Gli errori Linux hanno la tendenza ad essere scoperti e corretti rapidamente per mezzo di patch scaricabili da internet.
- ***Gratuità:*** è gratuito (si paga eventualmente solo il prezzo della distribuzione) - applicativi gratuiti

Svantaggi di Linux

- ***Mancanza di una "controparte"*** a cui gli utenti possono far riferimento per esigenze specifiche
- ***Applicazioni ridondanti***
- ***Applicazioni difficili da manutenere*** perché poco o per nulla documentate

Alcune distribuzioni

Linux Red Hat
La distribuzione più nota per gli utenti Linux
Sicurezza: 🐧🐧
Stabilità: 🐧🐧🐧
Semplicità: 🐧🐧🐧🐧

Mandrake
La francese Mandrake: il miglior compromesso
Sicurezza: 🐧🐧🐧🐧
Stabilità: 🐧🐧🐧🐧
Semplicità: 🐧🐧🐧🐧🐧

Corel Linux
Corel Linux: una debian molto user friendly
Sicurezza: 🐧🐧
Stabilità: 🐧🐧
Semplicità: 🐧🐧🐧🐧

Debian
La distribuzione tutta open source, stabile e aggiornata
Sicurezza: 🐧🐧🐧🐧🐧
Stabilità: 🐧🐧🐧🐧🐧
Semplicità: 🐧🐧

SuSe
La distribuzione tedesca semplice da installare
Sicurezza: 🐧🐧🐧🐧
Stabilità: 🐧🐧🐧🐧
Semplicità: 🐧🐧🐧

Slackware
La preferita dagli utenti esperti e dai puristi
Sicurezza: 🐧🐧🐧🐧🐧
Stabilità: 🐧🐧🐧🐧🐧
Semplicità: 🐧🐧

Madeinlinux
La distribuzione interamente italiana per il mercato italiano
Sicurezza: 🐧🐧🐧
Stabilità: 🐧🐧🐧🐧
Semplicità: 🐧🐧🐧

Caldera
Una distribuzione commerciale piena di strumenti
Sicurezza: 🐧🐧🐧🐧
Stabilità: 🐧🐧🐧🐧
Semplicità: 🐧🐧🐧🐧

- **UBUNTU** derivata da Debian, facile per installazione ed uso
- **ANDROID** per dispositivi mobili
- ……
- ……

Principali progetti in corso

- ***Kernel:*** base del sistema. Interfaccia fra il BIOS (hardware) e l'utilizzatore (il programma).
- ***Xfree86:*** implementazione libera di X-Window. E' l'interfaccia grafica di Linux.
- ***Gnome:*** interfaccia grafica conviviale e rapida. Il desktop di Gnome include applicazioni utili come editor, foglio elettronico, grafica (The Gimp)
- ***KDE:*** K Desktop Environment: interfaccia alternativa a Gnome. Piu' amichevole e simile a Windows ma + pesante di Gnome.
- ***Apache*:** Piattaforma server WEB

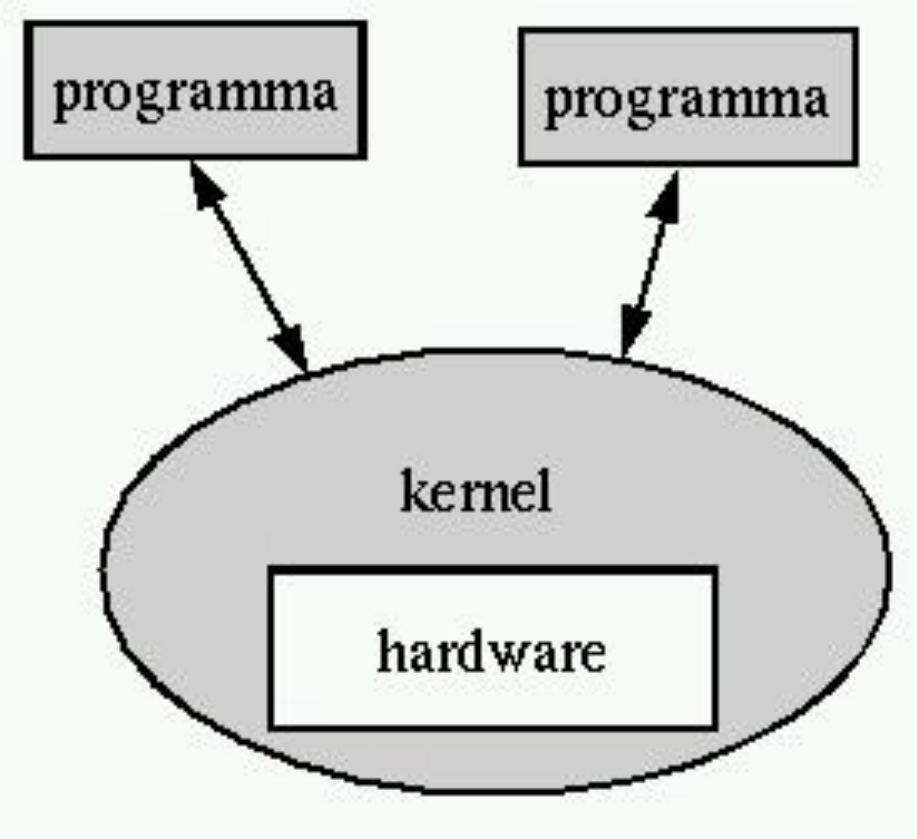
- ***Sendmail:*** Mail Transport Agent o server SMTP più diffuso su internet. Leggero altamente flessibile e configurabile
- ***Samba:*** Per scambiare e condividere files con Windows; consente a Linux di fare da server principale o secondario per reti microsoft.
- ***MySQL***: Sotto licenza GPL e' un server database SQL gratuito, robusto, con alte prestazioni e molto diffuso.

Panoramica su Linux

4 componenti principali:

- **kernel:** programma di base che esegue i programmi e gestisce i dispositivi HW
- **shell**: interfaccia con l'utente (riceve i comandi dall'utente e li invia al kernel per l'esecuzione)
- **file system:** definisce il modo in cui i file vengono organizzati su un dispositivo di memorizzazione (ad esempio HD)
- **programmi di servizio o applicazioni:** sono programmi specializzati (ie. Editor, web server, ecc.)

Kernel



Kernel

Programma di base che esegue i programmi e gestisce i dispositivi HW.

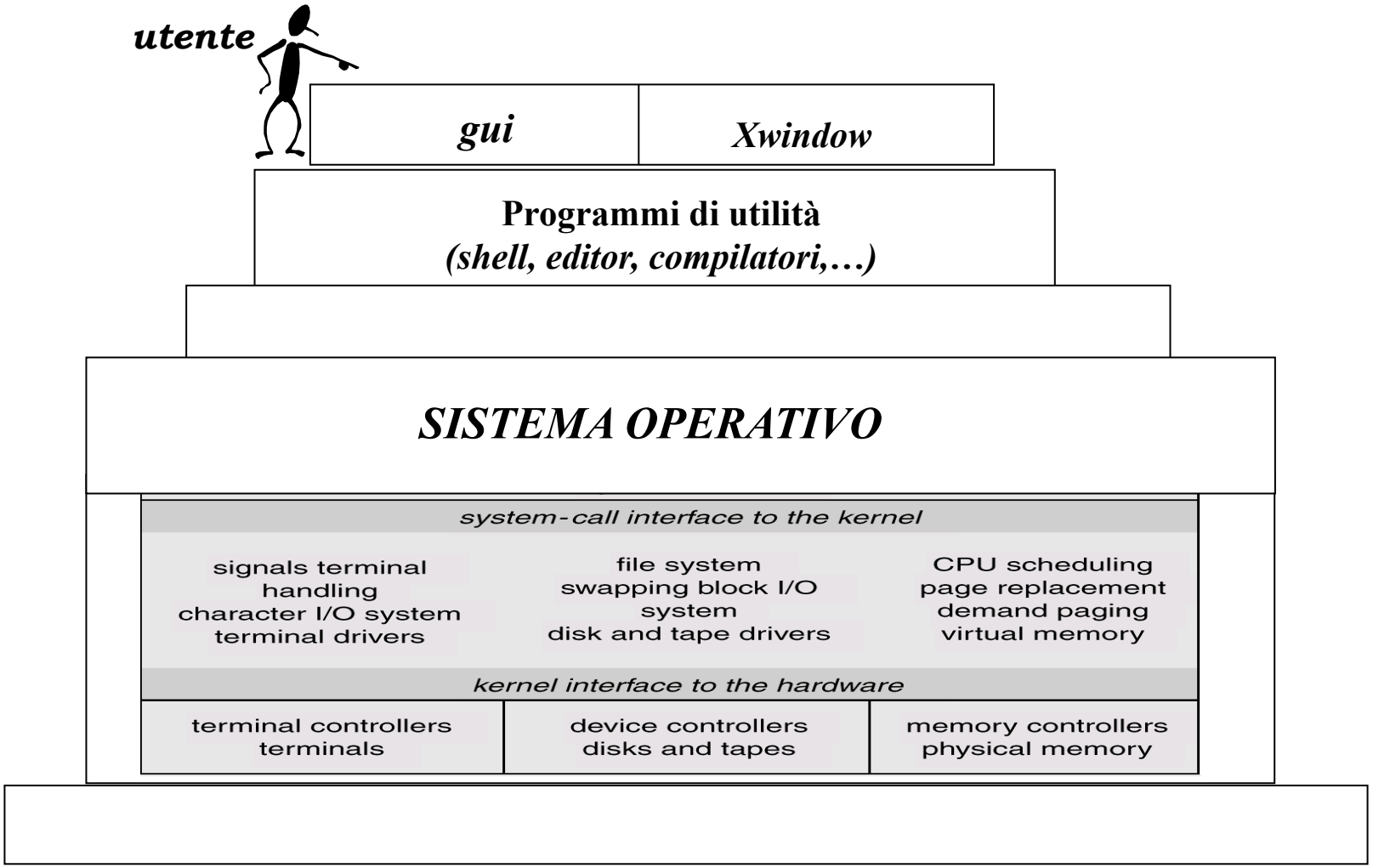
I programmi, utilizzando il kernel per la loro attività sono sollevati dall'interagire direttamente con la CPU.

Di solito è costituito da un file unico (vmlinuz o zImage,...), ma può comprendere anche moduli aggiuntivi per la gestione di periferiche o componenti specifici che devono poter essere attivati o disattivati durante il funzionamento del sistema.

Avviamento del Kernel (attraverso il sistema di avvio):

- **controlli diagnostici, in base ai tipi di dispositivi (HW) per il quale è stato predisposto in fase di installazione,**
- **monta (mount) il file system principale (root),**
- **avvia la procedura di inizializzazione del sistema (Init).**

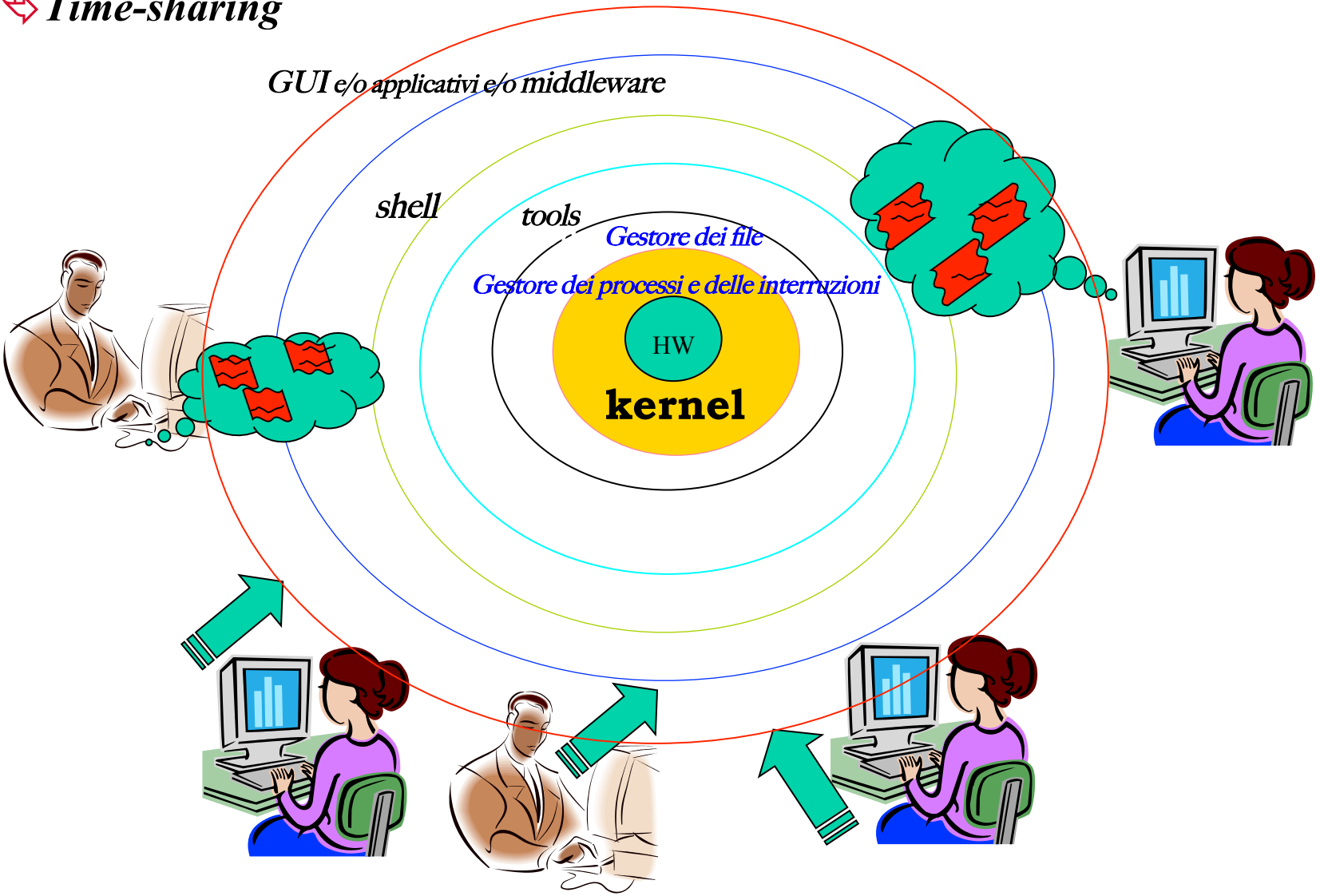
Sistemi Operativi multistrato



Modello di S.O. a Strati

- ***Multiuser***
- ***Multitask : multiprocessing - multithreading***
- ***Time-sharing***

processi e thread



Le Shell: Bourne e C-shell

Permette ad un utente di interagire con il SO. Si occupa di interpretare i comandi dell'utente ed inviarli al kernel per l'esecuzione.

Diversi tipi di Shell:

- Bourne Shell (AT&T Bell Labs)
- C-Shell (BSD)

L'implementazione della Bourne shell sotto Linux si chiama *bash* (Bourne-Again).

Una shell svolge i seguenti compiti:

- mostra il prompt per l'inserimento dei comandi,
- interpreta la riga di comando data dall'utente,
- esegue delle sostituzioni in base ai caratteri jolly e delle variabili d'ambient,e
- mette a disposizione alcuni comandi interni,
- mette in esecuzione i programmi,
- gestisce la redirezione dell'I/O,
- è in grado di interpretare file script di shell.

Shell: la storia

• '60: Dennis Ritchie e Ken Thompson di AT&T ⇒UNIX™: necessità della creazione di uno strumento con cui gli utenti potessero interagire con il nuovo sistema operativo.

• Altri S.O. utilizzavano interpreti di comandi (DOS= Disk Operating System).

• Nacque la ***Bourne shell*** (nota semplicemente come ***sh***), creata da S.R. Bourne.

• Sviluppate diverse shell, **come la C shell** (***csh***) e la **Korn shell** (***ksh***).

• **Bourne Again Shell** o ***bash***: per saperne di più sulla bash, leggete la relativa pagina *man*, che comparirà digitando ***man bash***.

• ***bash*** è installata di default con Red Hat Linux.

Il File System

E' quella parte del Sistema Operativo che fornisce i meccanismi di accesso e memorizzazione delle informazioni (programmi e dati) allocate nelle memorie di massa esterne.

Realizza i concetti:

- di **file**: unità logica di memorizzazione
- di **directory**: insieme di file (e directory)
- di **partizione**: insieme di file associato ad un particolare dispositivo fisico (o porzione di esso)

N.B. Le caratteristiche di file, directory e partizione sono del tutto indipendenti dalla natura e dal tipo di dispositivo utilizzato.

Il File System

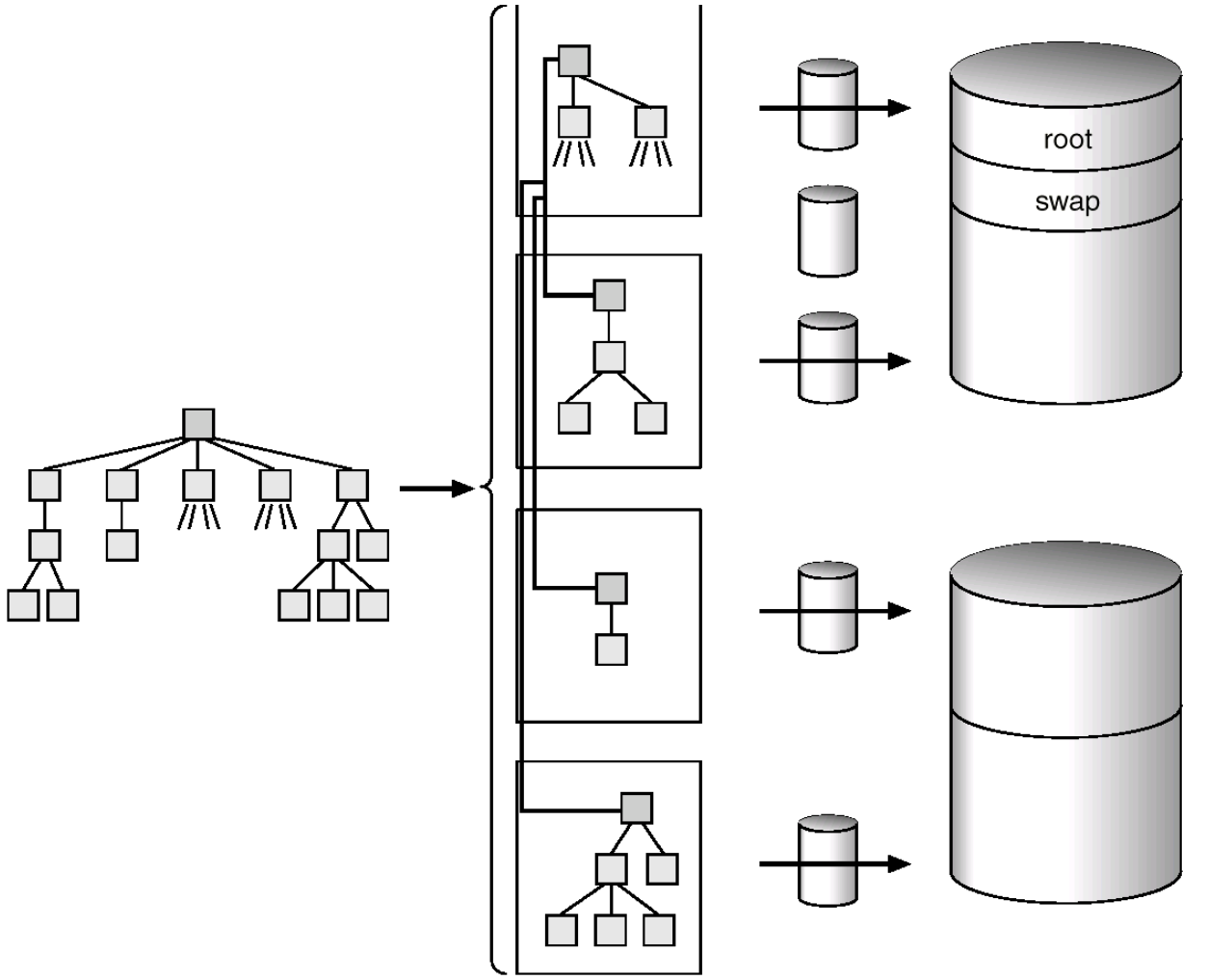
Dal punto di vista dell'utente il File System è caratterizzato dalle operazioni che ha a disposizione, cioè …

- come si definisce un file ,
- come il file viene denominato e protetto ,
- che operazioni sono permesse sui file e così via.

Per tenere traccia dei file, il File system mette a disposizione dell'utente directory contenenti un certo numero di elementi, uno per file

E' molto comune che un utente voglia raggruppare i suoi file in modo logico, attraverso una struttura gerarchica (ad albero)

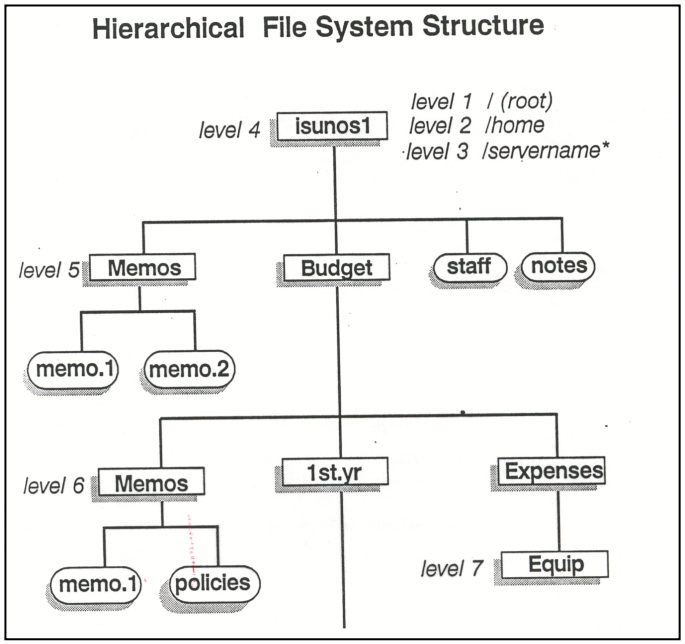
Associazione di un file system logico ai dispositivi fisici

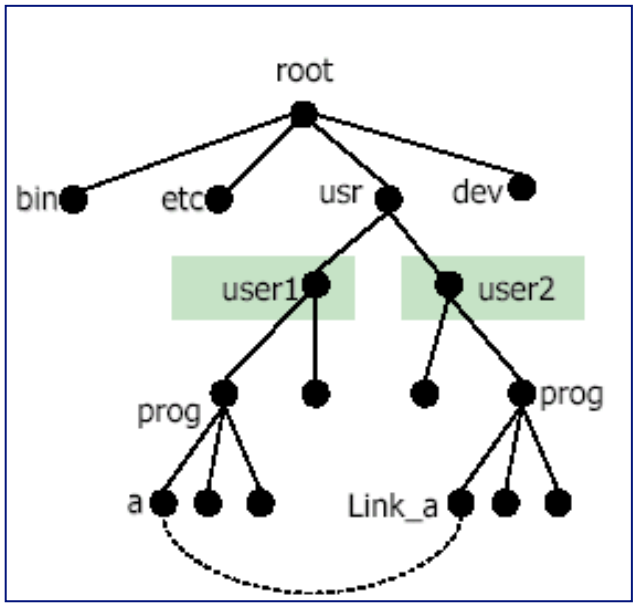


Memorizzazione dei File

Un file è di solito composto da una sequenza di blocchi .
Il File System deve tenere traccia dei blocchi occupati da ciascun file.

- *Negli ambienti Microsoft ad ogni disco è associata una Tabella di Allocazione dei File o* **FAT** *contenente un elemento per ogni blocco del disco.*
- *In LINUX ad ogni disco è associata un layout che descrive il contenuto in termini di blocchi, superblocchi ed i-node*





Il File System

In Linux *i file* sono organizzati in directory :

- più controllo e flessibilità ,
- non esiste la possibilità di distinguere tra un'unità di memorizzazione ed un'altra o fra diverse partizioni come avviene nel DOS (A:, B:, C:, ….),
- insieme interconnesso di directory contenti files,
- struttura gerarchica ad albero (nodo radice: root).

Interconnessione attraverso i cosidetti ***mount point*** che consentono di collegare un file system secondario a quello globale (operazioni di *mount* e *umount*).

File System

Ogni S.O. ha il proprio filesystem e non sempre questi sono tra loro compatibili:

Microsoft Windows : *FAT16, FAT32, NTFS*

Linux : *FAT16, FAT32, NTFS, EXT2, EXT3, EXT4*

Durante la fase di avvio della macchina (boot) per prima cosa vengono consultate le informazioni che risiedono nel BIOS (Basic Input/Output System) del PC.

Qui viene indicata la posizione del Master Boot Record (MBR) che prende il controllo della macchina e cerca nella tabella delle partizioni la prima partizione contrassegnata come attiva.

Da essa carica il Boot Record (BR) specifico del SO che vi risiede e lo esegue. Il BR contiene il codice necessario per caricare tutto il resto del SO.

Il journaling (presente in alcune versioni di File System) è una tecnica che consente di tenere traccia di modifiche effettuate sui file o sulle directory in un'area riservata del file system (journal), prima di effettuare la modifica sul File System.

Tale tecnica consente di ripristinare velocemente il File System in caso di crash del sistema o improvvise mancanze di corrente che potrebbero "corrompere" le informazioni contenute nel file system.

File Allocation Table

La FAT mantiene la traccia delle aree del disco disponibili e di quelle usate dai file e dalle directory: la differenza fra FAT12, FAT16 e FAT32 consiste appunto in quanti bit sono allocati per numerare i cluster del disco. Con 12 bit, il file system può indirizzare al massimo $2^{12} = 4096$ cluster, mentre con 32 si possono gestire $2^{32} = 4.294.967.296$ cluster.

Il file system **FAT** è un **file system con *allocazione concatenata***.
Una partizione FAT è strutturata in quattro sezioni diverse:

1. I **settori riservati**, che si trovano proprio all'inizio. Il primo settore riservato (settore zero) è il settore di avvio, seguito dal BIOS Parameter Block (con alcune informazioni di base del FS, in particolare il suo tipo, e puntatori alla posizione delle altre sezioni).
2. La **Regione FAT**: Contiene almeno due copie della FAT (per motivi di sicurezza). Rappresentano la mappa della regione dati. Una partizione è divisa in **cluster** (***insieme di settori fisici contigui***) dalle dimensioni variabili tra 2 e 32 KB. Ogni file è strutturato sul disco come una lista concatenata di ***cluster*** non necessariamente contigui: questa è la ragione principale per cui si parla di frammentazione del disco nei filesystem FAT.
3. La **Regione della ROOT directory**: è una tabella che memorizza le cartelle e i files presenti nella directory di root.
4. L'**area dati**: è dove files e cartelle sono realmente memorizzati e occupa la maggior parte della partizione

FAT utilizza il formato little endian per le voci nell'intestazione e la/le FAT.

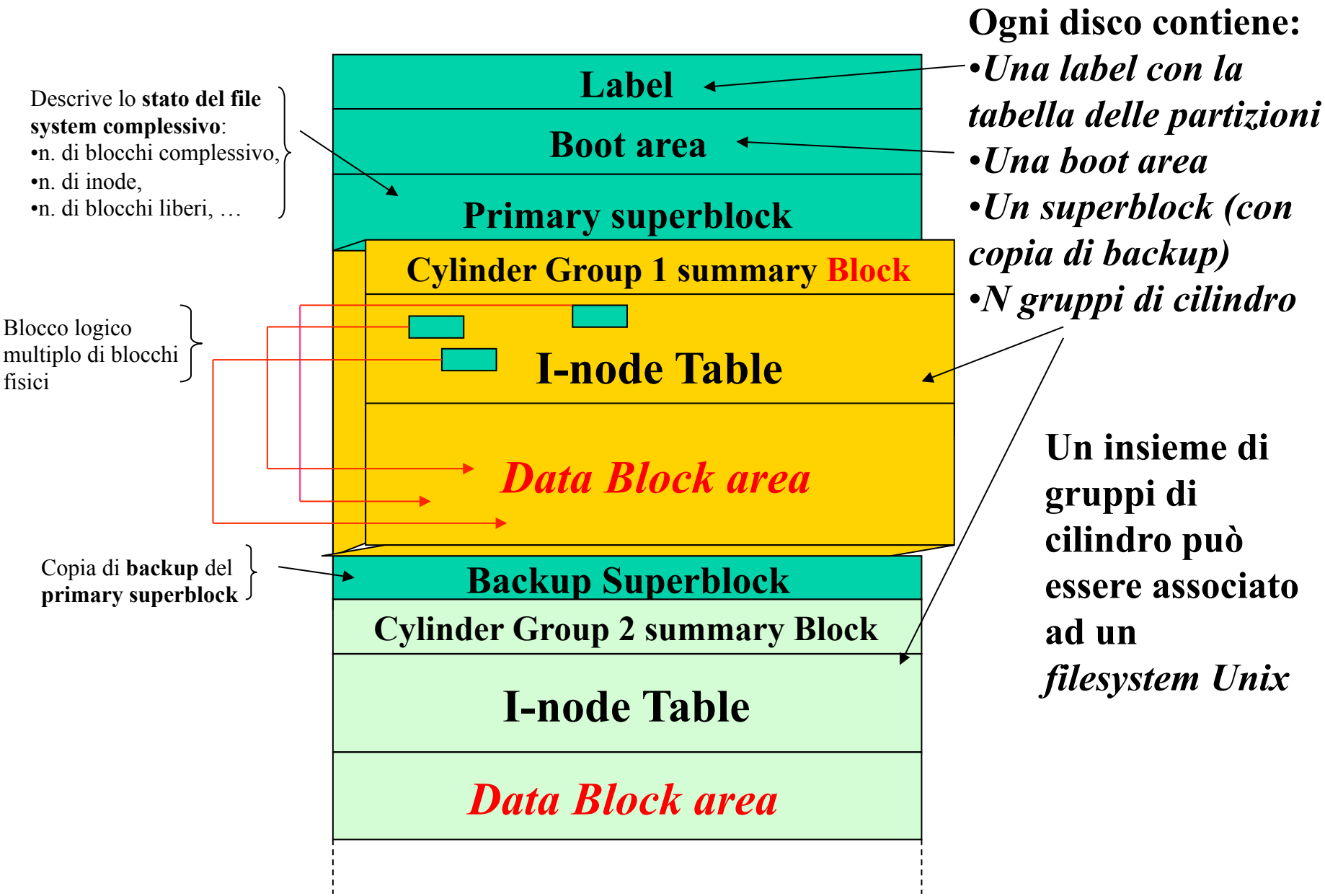
New Technology File System

NTFS è un file system dei sistemi operativi basati su kernel NT.

NTFS sfrutta un'indicizzazione a 64 bit.
La struttura principale di un filesystem NTFS è la *Master File Table* (MFT), una tabella strutturata in blocchi (solitamente in record di 1KB) che contiene gli attributi di tutti i file del volume.
Le directory sono memorizzate come file: in ogni *file-directory* sono presenti degli attributi speciali, che si riferiscono ai file contenuti in tale directory.
I dati veri e propri dei file sono memorizzati in ***stream*** (flussi sequenziali) puntati da appositi attributi ***Data***

- ✓ **NTFS è un sistema transazionale** (o **"Journaled"** come si dice nei sistemi operativi Apple come Mac OS X); questo vuol dire che se un'operazione è interrotta a metà (ad esempio per un blackout) viene persa solo quell'operazione ma non è compromessa l'integrità del file system il quale resta comunque leggibile dal computer.
- ✓ **A ciascun file o cartella è possibile assegnare dei diritti di accesso** (lettura, scrittura, modifica, cancellazione e altri).
- ✓ I **nomi dei file e delle cartelle possono essere lunghi fino a 255 caratteri** e possono contenere caratteri di tutte le lingue del mondo grazie alla codifica Unicode.
- ✓ **La dimensione dei volumi e il massimo numero di file sono praticamente illimitati**; la dimensione del volume può raggiungere al massimo i 256 Terabytes (2^{48} clusters - 1), il numero limite di file è invece di circa 4,3 miliardi (2^{32} - 1). **La dimensione massima di un singolo file è di 16 Terabytes.**

Esempio : Layout del disco in EXT3



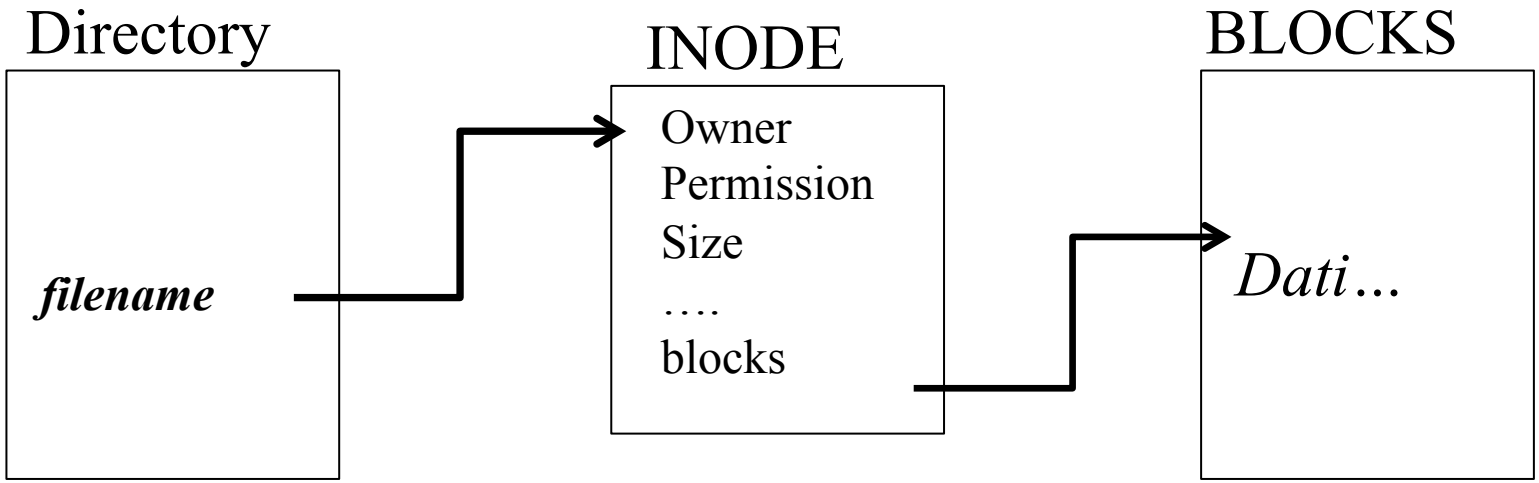
I-node

Ad ogni file è associata una piccola tabella, detta i-node ("index-node"), contenente

- ✓ *gli **attributi** del file (tipo, dimensione, permessi,....)*
- ✓ *gli **indirizzi** dei primi blocchi del disco su cui è memorizzato il file*
- ✓ *l'indirizzo di un **blocco a singola indirezione** contenente gli indirizzi di blocchi a singola indirezione*
- ✓ *l'indirizzo di un **blocco a doppia indirezione** contenente gli indirizzi di ulteriori blocchi di dati su disco*

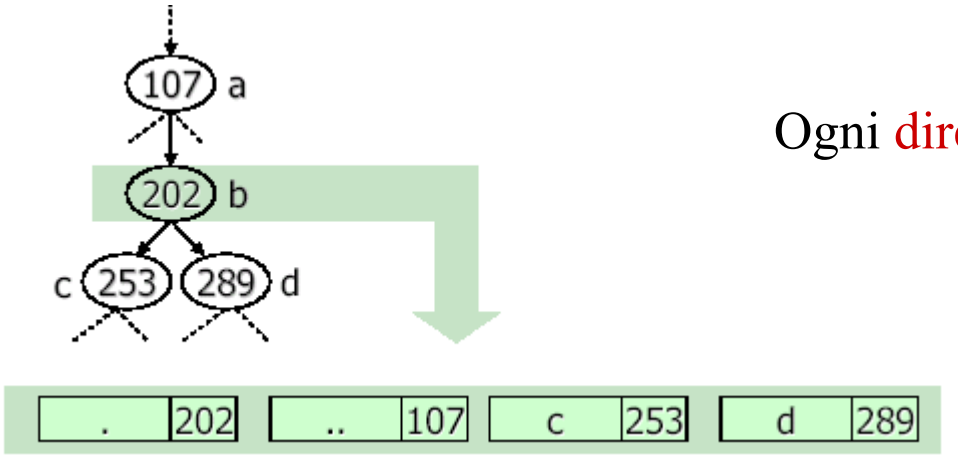
Ogni i-node è identificato da un i-number

Tutte le operazioni su un file o una directory vengono effettuate tramite il suo i-node, che contiene tutte le informazioni sul file stesso, esclusi i dati veri e propri



Directory

E' un ***file come tutti gli altri***, con l'unica differenza che **i dati in esso contenuti sono le informazioni sui files nella directory** e viene pertanto gestito in modo particolare dal **fs** e dal **kernel**.
Ciascuna **entry** di directory è un record di lunghezza variabile allineata alla word (4 bytes) ***contenente solamente il nome del file ed il suo numero di inode***.

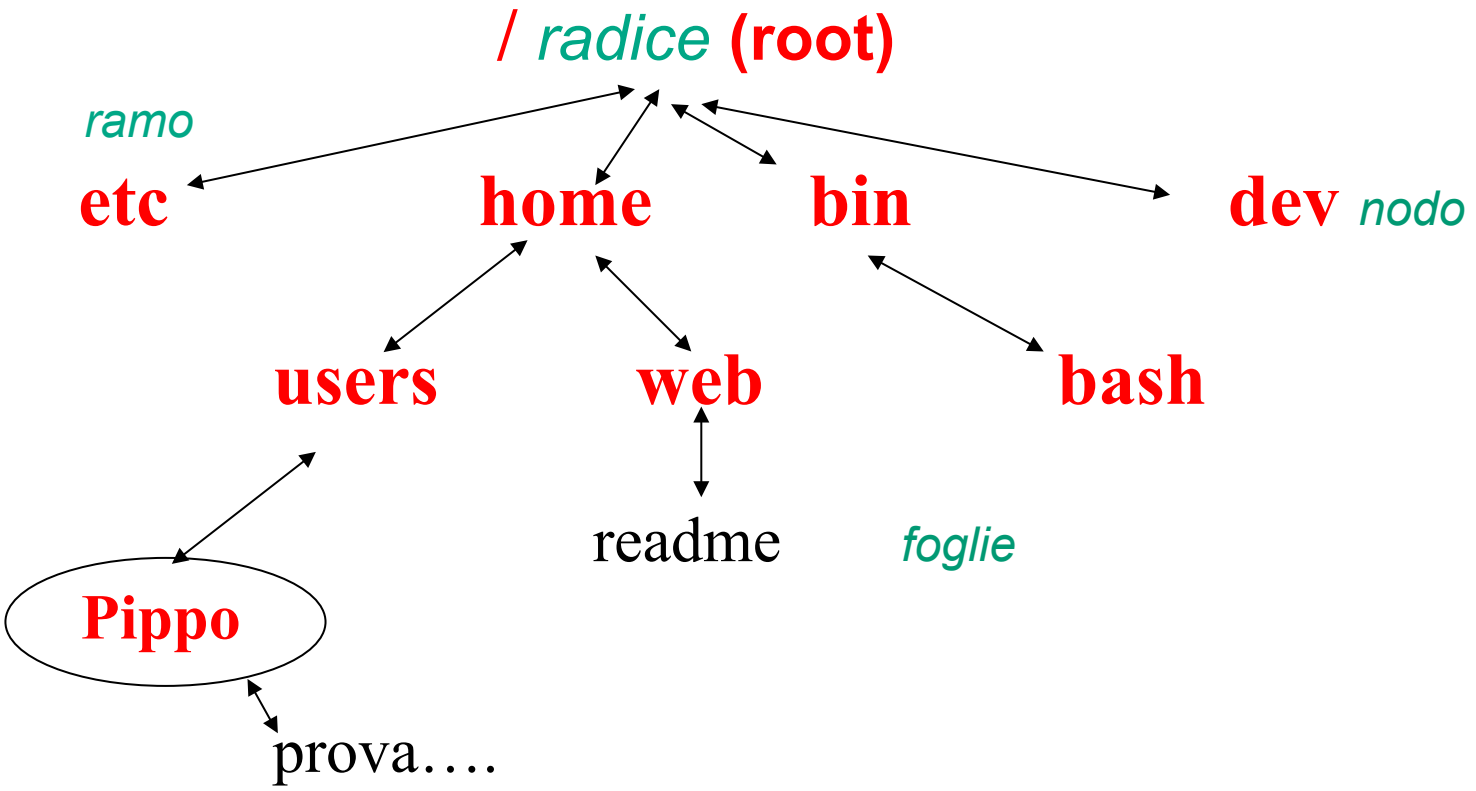


Ogni directory ha almeno 2 entry:
> directory stessa
> directory padre

Tutte le altre informazioni sul file non hanno niente a che fare con la directory in cui esso è contenuto e sono pertanto memorizzate nel suo ***inode.***
La directory serve solo a collegare il nome del file col suo inode.

Struttura del FS Linux

dir: etc, home, ecc…
file ordinari: readme, prova



Tipiche dir di Linux

/bin programmi di base
/dev file speciali (I/O devices)
/etc file per l'amministrazione del sistema
/lib librerie dei programmi
/tmp area temporanea
/home home directory degli utenti

Ad ogni utente viene assegnata, da parte del system administrator, una directory di sua proprietà (**home directory**) che ha come nome lo username dell'utente stesso.

In essa l'utente potrà creare tutti i file (o subdirectory) che desidera.

Spesso le home directory sono sotto la directory di sistema /home.

Per denotare la propria home directory si può usare l'abbreviazione "~".

I programmi di servizio

Editor: vi e **emacs**
Filtri: programmi che leggono l'input utente da file e producono in output una versione modificata del file
Web browser
Email
Compilatori
….

Contrariamente ai sistemi DOS/Windows Linux - Unix è ***CASE SENSITIVE*** ossia differenzia i caratteri maiuscoli da quelli minuscoli.

Nei comandi impartiti ai sistemi Linux- Unix i caratteri minuscoli sono quindi diversi dai caratteri maiuscoli.

Ad es. il comando "*ls*" sarà diverso da "*LS*", "*Ls*" è diverso da "*lS*" ecc….

Livelli di esecuzione (runlevels)…

*Un **runlevel** è una particolare configurazione del kernel in cui certe cose sono permesse ed altre no:*

•0 : halt - avvia la sequenza di arresto del sistema (shutdown)
•1 : single-user mode - un solo utente connesso, servizi di rete disabilitati
•2 : multi-user mode - multiutente, servizi di rete attivo ma file sharing disabilitato
•3 : multi-user con servizi di rete(es. nfs)
•*4 : non usato*
•5 : multi-user con interfaccia grafica(X11)
•6 : reboot

• "**/etc/inittab**" è il file di configurazione dei *runlevel*
• "**/etc/rc**x**.d/**" è la cartella contenente gli script di inizializzazione per ciascun livello

…runlevels…

- Per default l'installazione definisce il ***livello 3*** come **initdefault**:
  - alla partenza il sistema operativo e' in ***multiuser mode***, e tutti i servizi di rete previsti sono attivi;
  - **non** e' attivo l'X-server (evocabile con il comando **'*startx*'** previsto con il ***livello 5)***.

- Per definire un diverso **initdefault** e' necessario modificare il file **/etc/inittab**.
  Per esempio, modificando il record
  id:3:initdefault: in ***id:5:initdefault:***
  al reboot successivo il sistema attivera' automaticamente anche l'X-server.

- Il processo di **BOOT** carica e inizializza tutti i servizi (***daemon***)
  - Uno stesso daemon puo' essere presente in più runlevel (tipicamente 2 e 3)

- **INIT** legge quali cose fare da ***/etc/inittab*** dove e' segnato il ***runlevel*** iniziale (solitamente il 3), e cosa fare per ogni runlevel.

…runlevels

Il **runlevel 3** e' quello che vi permette di avviare il sistema in modalita' testo (ovvero vi verra' offerta una semplice shell testuale).

Il **runlevel 5** invece e' quello grafico, che avvia il server X subito.

Le due modalità non sono esclusive; si può sempre avviare il server grafico dalla modalita' ***console*** *(come già detto)* oppure passare ad una semplice console da un ambiente desktop come ***Gnome***.

Se entrate in modalita' grafica potete:

a) passare alle console di login premendo CTRL+ALT+F1 (fino a F6); per tornare alla grafica, CTRL+ALT+F7.

b) aprire una shell all'interno dell'ambiente grafico

basta cliccare sull'icona che assomiglia a uno schermo nero presente nel Panel ; a questo punto avrete una shell con i privilegi dell'utente con cui siete entrati nel sistema.

Accesso al sistema

- **Linux: sistema multiutente**
  - utenti possono avere accesso al sistema avendo i propri dati, i propri programmi e impostazioni completamente separate da quelle di altri utenti.
  - possibilità di accedere alla risorse del sistema simultaneamente (sia direttamente tramite console (tastiera) sia da remoto via rete)

==> necessità di proteggere o nascondere le informazioni

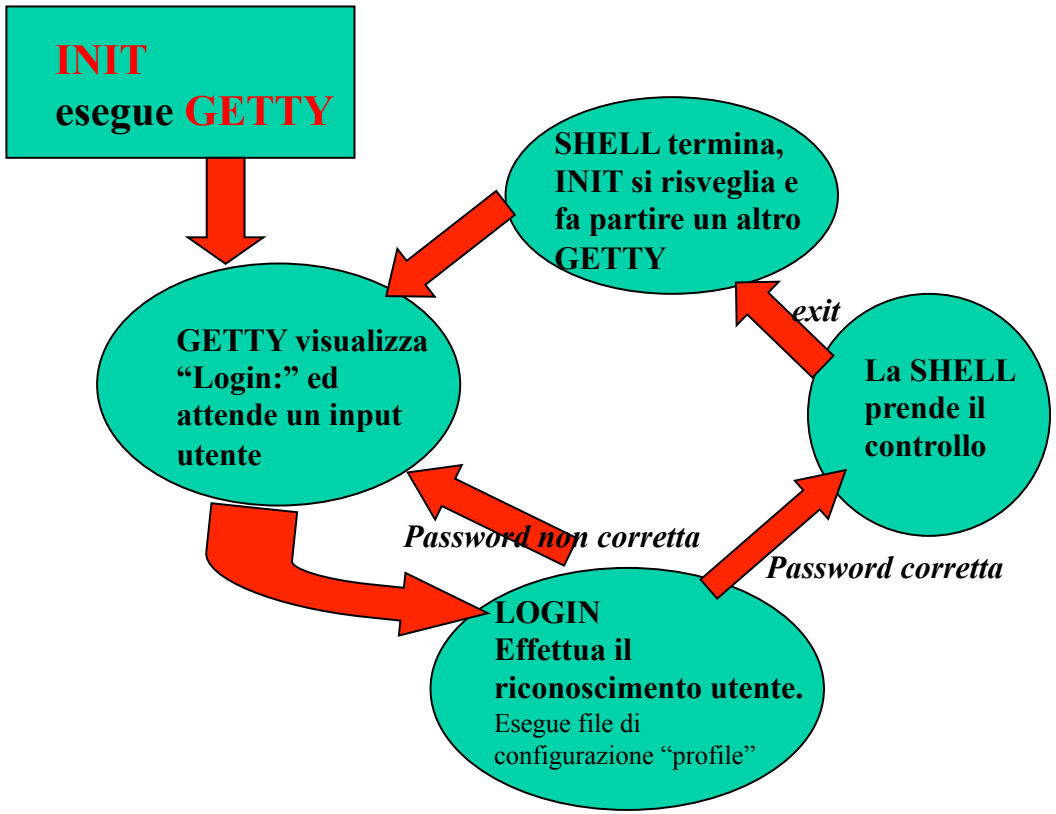
- Concetto di gruppo (es. staff, users, root,...): possibilità di lavorare sugli stessi documenti;

- Ogni utente appartiene a un gruppo, ma per determinate esigenze puo' far parte anche di altri gruppi: configurazioni

- Un gruppo può essere composto anche da un solo utente

login

Il programma **login** controlla che esista un utente con il nome **username,** e che la **password** corrisponda a quella codificata nel file **/etc/passwd.** A questo punto avete una ”identità” **username** a cui corrisponde uno **userid** numerico, e appartenete ad alcuni **gruppi (**ai quali pure corrispondono **groupid** numerici).
Nota: Questa identità viene chiamata "**account**".




```
<login>
do
{
<ricevi comando dal file di input>
<interpreta il comando>
<esegui comando>
}
while (!EOF);
<logout>
```

shell di login

• La shell di login viene attivata automaticamente all'atto di **login**.

• Interpreta prima di tutto uno ***script*** uguale per tutti gli utenti e scritto dal sistemista: ***/etc/profile***

• Successivamente esegue uno ***script, definito dall'utente*** nella propria home directory.

Il nome di questo script varia a seconda del tipo di shell. Per la ***bash***, esegue solo il primo script fra: ***~/.bash_profile, ~/.bash_login, ~/.profile***, oltre allo script ***~/.bashrc*** **.**

• L'utente può quindi personalizzare il suo ambiente di lavoro usando gli script di login.

• All'uscita della sessione viene **eseguito lo script** ***~/.bash_logout***.

una volta effettuato, il **logout** chiude la shell aperta con il precedente login e tutti i successivi programmi lanciati dall'utente.

Le pagine man

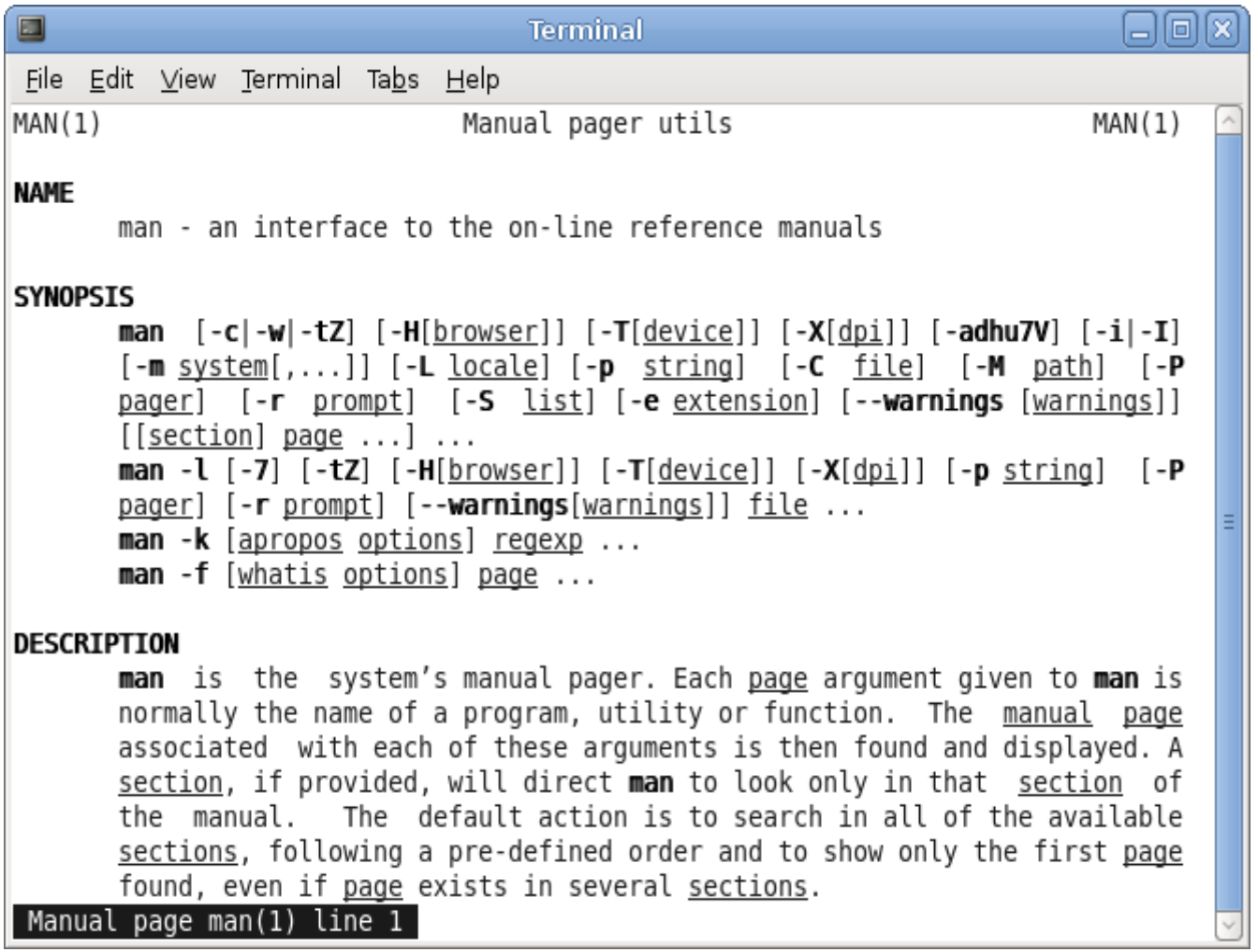
Le pagine *man* forniscono in un formato estremamente condensato il riassunto delle funzionalità del comando, le opzioni disponibili e la sintassi utilizzata per eseguire il comando.

man *nome_comando*

Es: ***man man*** apre la pagina **man** del comando **man** .

Per scorrere le pagine man usare le freccette;

q per uscire



Shell commands

Sintassi standard: ***Nome_comando argomenti -opzioni***

➔Per indicare che un'opzione o un argomento può essere omesso, si usano le parentesi quadre: **[*opzione*]**

➔Se due opzioni o argomenti sono mutuamente esclusivi vengono separati da una barra | : **arg1 | arg2**

➔Quando un argomento può essere ripetuto ***n*** volte gli si pospongono i puntini: ***arg...***

Esempio: il comando **ls** per visualizzare un file system

Sintassi (sempl): ls [-*opzioni*…] [*file*…]

opzioni:
- l (long format): per ogni file una linea che contiene i diritti, il numero di link, il proprietario del file, il gruppo del proprietario, l'occupazione di disco (blocchi), la data e l'ora dell'ultima modifica o dell'ultimo accesso, e il nome;
- t (time): la lista ordinata per data dell'ultima modifica;
- u: la lista ordinata per data dell'ultimo accesso;
- r (reverse order): inverte l'ordine;
- a (all files): fornisce una lista completa (i file che cominciano con il punto non vengono visualizzati *file nascosti*);
- F (classify): aggiunge al termine del nome del file un carattere che ne indica il tipo *(eseguibile:*, direttorio: /, link simbolico: @, FIFO: |, socket:=, niente per file regolari).*

Vedere : *man ls*

metacaratteri

I metacaratteri specificano un pattern per identificare un insieme di nomi di file già esistenti. La shell provvede a sostituire i metacaratteri con i nomi dei file.

· ***** sta per qualsiasi sequenza (anche vuota) di caratteri: es. *.java, file.*

· **?** sta per un carattere qualsiasi: es. file.do?

· **[ ]** specificano una lista o un intervallo di caratteri (es. [a-c], [A-Za-z]: file.do[ct])

Gli apici possono essere usati per indicare che eventuali metacaratteri non vanno espansi.

Esempi:

$ ls

hw temp vi.txt f1.txt f12.txt f2.txt temp.c

$ **ls f?.txt**

f1.txt f2.txt

$ **ls [ft]*.***

f1.txt f12.txt f2.txt temp.c

*Ad ogni modo, se vi ricordate solo le prime lettere di un comando, il tasto **TAB** vi viene in aiuto. Alla prima pressione se trova solo un comando che inizia con le lettere da voi immesse, completa il comando in automatico.*
Se invece ce n'e' piu' di uno emette un beep, alla seconda pressione vi presentera' la lista di comandi possibili.

utenti e gruppi

username: nome identificativo di un account (massimo 8 caratteri)

userid (**uid**): numero intero univoco: un database fuori dal nucleo(*kernel*) di Linux collega username a uid (file ***etc/passwd***) . *Se due o più utenti hanno lo stesso uid, allora si tratta dello stesso utente ma con nomi diversi.*

gruppo: nome identificativo di un gruppo, a cui corrisponde un **groupid** (**gid**) numerico.

Ogni utente deve risiedere in un gruppo. Se non risiede in un gruppo, lo stesso utente forma un gruppo a sé, di cui egli è il solo membro.

Tramite i gruppi è possibile definire delle azioni che sono concesse a più utenti che vi faranno parte.

/etc/passwd

Contiene informazioni sugli utenti definiti

Schema:

username:password:UID:GID:'comment':home directory:login command

/etc/shadow

Contiene informazioni sulle password crittografate degli utenti definiti

schema:

```
username:encrypted password:last change of  password:minimum days that the
password should exist without changing:maximum days of the password's
existance :days until the user gets a message that his password will
expire:number of days that the account will stay out of order before it can
be totally deleted:exact date of password's expiration
```

/etc/group

Contiene l'elenco di tutti i gruppi di utenti presenti nel sistema

Ad ogni gruppo è associata una riga nella quale si trova l'IDgroup e l'elenco degli utenti che ne fanno parte.

Di fatto i gruppi servono per gestire con maggiore flessibilità l'accesso ai file e di conseguenza l'uso delle risorse.

schema:

group name:password:GID:users of the group

Variabili d'ambiente

Digitando al *prompt* di **bash** il comando **env** viene visualizzato l'elenco delle variabili di ambiente della shell.

Ognuna di queste variabili contribuisce a personalizzare l'ambiente:

- **PATH è assegnata con l'insieme delle directory nelle quali il sistema cerca i programmi**

```
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/
usr/texbin:/usr/X11/bin
```

- **Il comando PWD (present working directory) chiede alla macchina di indicarvi in quale directory vi trovate**

```
$ echo $PWD
/Users/nicole
```

***echo** mostra l'input sullo **stdout**, mentre il referencing operator **$** mappa il nome di una variabile nel suo valore (Il comando echo e il simbolo del "$" davanti alla variabile non fanno altro che visualizzarne il contenuto)*

Comandi di controllo dell'ambiente

- *logout* *per uscire da una login*
- passwd *per cambiare password dell'utente*
- stty *per impostare le opzioni per il terminale*
- finger *per visualizzare le informazioni sugli utenti*
- ps *per visualizzare i processi del sistema*
- *top* *mostra i processi in tempo reale*
- *kill* *termina un processo*
- env *per visualizzare i parametri di configurazione dell'ambiente*
- set *per settare i parametri di configurazione*
- alias *per definire abbreviazioni o nuovi nomi di un comando*
- history *per visualizzare gli ultimi comandi digitati.*

Passwd

È possibile cambiare la propria password di utente, mediante il comando *passwd*

- **Verrà prima chiesta la vecchia password (per motivi di sicurezza)**
- **Se ci si dimentica della password, bisogna chiedere all' amministratore di sistema (utente *root*)**

PS

Per visualizzare i ***processi attivi*** con il loro "proprietario"

Es:

debian:~# ps x

Process ID	*terminale*	*stato del processo*	*tempo di CPU*	*nome comando*
7032	p1	S	0:00	-csh
7120	p1	D	0:00	find/-name test
7231	p1	R	0:00	ps x

History

Quando si vuole ripetere un comando già digitato nella shell è sufficiente premere il ***tasto con la freccetta in su***. In questa maniera si accede alla history dei comandi.
Se l'history è troppo lunga o si ricordano le prime lettere del comando, lo si può richiamare direttamente anteponendogli ***il punto esclamativo***.
Ad esempio, se si è digitato il comando: ***$ ps -eF |sort***
che ordina per nome utente i processi attualmente sul sistema, e lo si vuole richiamare, è sufficiente digitare: ***$!p***
Questa funzionalità é particolarmente utile quando si devono riscrivere comandi molto lunghi.
In pratica, quando si esegue il comando "!nome", la shell lo interpreta come "Esegui l'ultimo comando che nella history inizia con nome".

Se poi c'è un problema di cattiva digitazione di un comando, è possibile correggere l'errore senza dover scrivere di nuovo il comando; ad esempio se ho digitato:
$ ps -eF |srot,
con il comando '^ro^or' viene sostituita la stringa 'ro' con la stringa 'or' e viene eseguito automaticamente il comando risultante.

alias

Crea oppure ***rimuove il pseudonimo*** di un comando.

Attenzione quando si termina la sessione di lavoro vengono eliminati anche gli *alias*, per mantenerli è necessario inserirli nel *profile* della console.

Comandi a livello di sistema

- whatis *cmd* *descrive brevemente il comando cmd*
- which *cmd* *fornisce informazione sull'alias di cmd*
- whereis *cmd* *fornisce il path di cmd*
- date *visualizza la data e l'ora corrente di sistema*
- cal *visualizza il calendario*
- bc *attiva una calcolatrice con le operazioni aritmetiche base*
- wc *file* *conta le linee, le parole ed i caratteri di un file*
- spell *segnala errori di spelling*
- lpr *file* *invia allo spool di stampa un file*
- lpq *mostra lo stato della coda di stampa*
- lprm *job* *rimuove job dalla coda di stampa*

- who mostra gli utenti attualmente connessi
- ***su*** *lancia una shell con i privilegi dell'utente indicato.*
- *last* *mostra l'elenco degli accessi più recenti*
- *lastlog* *mostra l'ultimo accesso per ciascun utente*
- *cat /var/log* *visualizza i log di sistema; cat manda in stdout quanto riceve in stdin*

date

date imposta la data e l'ora del sistema

es: date

Eseguendo semplicemente il comando date, riporta la data e l'ora corrente.

es*: date -s '02 May 2000 10:55:10'*

Questo esempio setta la data al 2 Maggio 2000, e l'ora alle 10:55:10, omettendo l'ora setta automaticamente all'ora 00:00:00

es: *date --date '20 days ago'*

Stampa la data di 20 giorni fa

es: *date --date '3 months ago'*

Stampa la data equivalente al giorno corrente di 3 mesi fa.

es: *date --date '1 month 1 day'*

Stampa la data corrispondente a quella fra un mese ed un giorno.

es: *date --date '1 month 1 day ago'*

Stampa la data di un mese ed un giorno fa.

es: *date +'%T'*

Stampa l'ora corrente nel formato ore:minuti:secondi

Stampa di file

`lpr` [nomifile...]

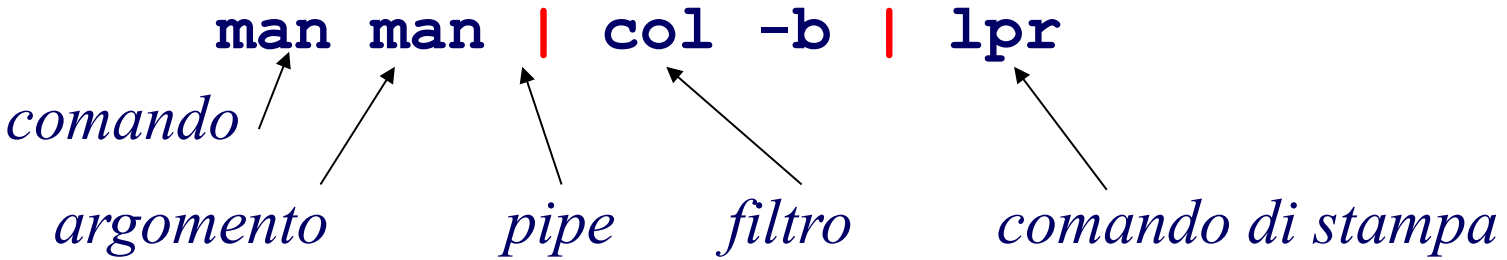
Tecnica di **spool**: i file vengono inseriti in una ***coda*** e la stampa va in *background*

lpr -P *destination filenames* stampa i file con la stampante chiamata

$ lpr -P stampante2 file.odp

stampa delle pagine man

Es: per stampare una pagina man per man:



Il comando **man** invia l'output della pagina man al filtro implementato da col, il quale rimuove i caratteri di formattazione video della pagina e la passa alla stampante predefinita.

Il **segno** | provoca il passaggio dello stdout del comando alla sua sinistra allo stdin del comando a destra.

Comandi di base per il file system

• ls	*elenca file*
• cd	*cambia directory corrente*
• pwd	*mostra la directory corrente*
• mkdir	*crea directory*
• cp	*copia*
• rm	*rimuovi*
• mv	*sposta*
• chown	*cambia il proprietario*
• chmod	*cambia i permessi di accesso*
• ln	*crea un collegamento*
• cat	*mostra il contenuto di un file*
• *find*	*cerca file in una dir*
• *du*	*mostra le dimensioni di un file system*
• *df*	*mostra lo spazio disponibile su disco*

ls: list directory contents

Sintassi: **ls [-*opzioni*…] [*file*…]**

opzioni:

-l (long format): per ogni file una linea che ne mostra il tipo, i diritti, il numero di link, il proprietario, il gruppo cui appartiene il proprietario, l'occupazione del disco (blocchi), data e ora dell'ultima modifica o dell'ultimo accesso, nome del file:

```
-rwxr-xr-x   1 nicole  staff   317599 Dec  7 13:34 200504K002.pdf
-rwxr-xr-x   1 nicole  staff  2228562 Dec  7 13:32 CIT2002ConferenceBook.pdf
-rwxr-xr-x   1 nicole  staff    14848 Dec  7 13:32 Cartel1.xls
-rwxr-xr-x   1 nicole  staff    45192 Dec  7 13:32 HMM_engagement_TITLE.pdf
```

-t (time): lista ordinata per data ultima modifica
-u: lista ordinata per data ultimo accesso
-r (reverse order): inverte l'ordine
-a (all files): visualizza anche i file che cominciano con il punto(*file nascosti*)
-F (classify): aggiunge al termine del nome del file un carattere che ne indica il tipo **(eseguibile: *, direttorio: /, link simbolico: @, FIFO: |, socket: =, niente per file regolari)** :

```
Desktop nicole$ ls –F
cars09.ppt*
corpus italiano/
contesto italiano.rar*
CIT2002ConferenceBook.pdf*
corpus def/
```

Directory

Sono sequenze di byte, come i file ordinari.
A differenza dei file ordinari

- Non contengono dati ma un elenco di nomi di file e relativi riferimenti ad altre strutture dati del file system

- Non possono essere scritte da programmi ordinari

Una directory è un indice contenente i riferimenti (i-number) di tutti i file memorizzati nella directory stessa

path relativi e assoluti

- ogni utente può specificare un file attraverso:
  - **nome relativo**: è riferito alla posizione dell'utente nel file system (direttorio corrente)
  - **nome assoluto**: è riferito alla radice della gerarchia (/)
- nomi particolari
  - . è il direttorio corrente (visualizzato da pwd)
  - .. è il direttorio 'padre'

Ad ogni utente viene assegnata, da parte del system administrator, una directory di sua proprietà (**home directory**) che ha come nome lo *username* dell'utente stesso. In essa, l'utente potrà creare tutti i file (o subdirectory) che desidera.

Spesso le home directory sono sotto la directory di sistema **/home**

Per denotare la propria home directory si può usare l'abbreviazione "~" (tilde)

Working directory

Ogni utente opera, ad ogni istante, su una directory corrente, detta **working directory (pwd)** (dopo la login e' la sua home)
L'utente può cambiare la working directory con il comando **cd**

CD serve per muoversi attraverso le directory.

Esempio: albero delle directory: /home/user1/doc/notes.txt

Se la posizione corrente è /home/user1, per portarsi nella directory dove si trova *notes.txt* basta digitare il comando: **cd doc/**

Per portarsi nella home directory personale digitare il comando ***cd ~*** (o anche ***cd /username/***), mentre per portarsi nel primo livello (root) dell'albero delle directory digitare ***cd /*** ; col comando ***cd ..*** si va alla directory superiore.

pwd:

determinare la directory in cui mi trovo

La stringa indica che vi trovate nella directory */newuser*, che è la vostra directory */ home*.

Il comando ***pwd*** significa ***print working directory***.

Quando digitate ***pwd***, chiedete al vostro sistema Linux la vostra posizione. Il sistema risponde "mostrandovi" sul monitor la directory in cui vi trovate, conosciuto anche come *standard output*.

find files

Quando una directory contiene molti file puo' servire cercarseli per nome, dimensione, tipo, ...

Sintassi

find elenco_directory opzioni

Opzioni

-name *nome_file* = ricerca in base al nome

-print = visualizza sullo stdout i risultati della ricerca

Esempio

$ find report -name lunedi -print

report/lunedi

Link

I link sono particolari file che puntano ad altri file o directory;

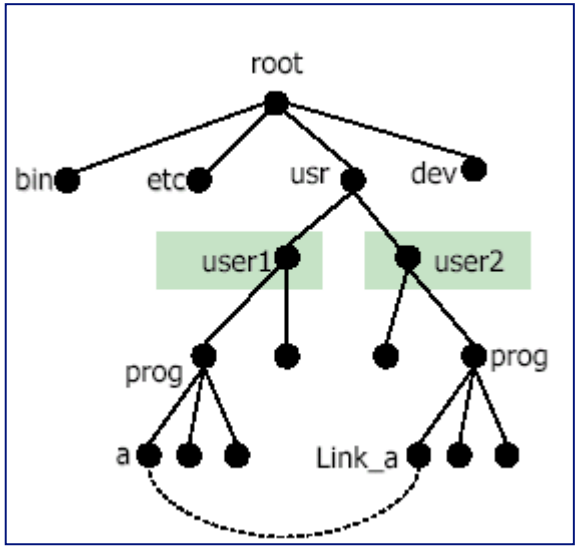
- **danno più flessibilità alla struttura gerarchica del File System**
- **consentono condivisione file - non duplicazione - tra dir diverse**

Sintassi: ***ln [OPTION]... [-T] TARGET LINK_NAME***

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

Ad es. gianluca@debian:/usr/user2/prog$ ln /usr/user1/prog/a link_a

Il file a è identificato mediante due cammini differenti:
/usr/user1/prog/a
/usr/user2/prog/link_a



Link

Hard-link (o fisici): nomi per lo stesso file

➔non possono essere usati tra File System diversi, e per le dir

➔Per default **ln** crea un hard link: di fatto un ***altro nome*** per un file esistente, originale e link sono indistinguibili, ossia condividono lo ***stesso inode***

Soft-link (o simbolici): file contenenti il ***percorso*** per trovare il file riferito

➔opzione **-s** del comando **ln**

Che link è?

- **Link fisici**

Esempio:
```
$ ln oggi meteo
$ ls
oggi meteo
$ ls -l oggi meteo
-rw-rw-r-- 2 alice group 563 Nov 15 10:30 oggi
-rw-rw-r-- 2 alice group 563 Nov 15 10:30 meteo
$ ls -i oggi meteo questo_no
1234oggi 1234meteo 2345questo_no
```

-i *prints the i-nodes*

- **Link simbolici**

Esempio: $ ln -s pranzo /home/giorgio/menu

```
$ ls -l pranzo /home/giorgio/menu
lrw-rw-r-- 1 alice group 4 Nov 15 10:30 pranzo
-rw-rw-r-- 1 giorgio group 793 Nov 15 10:30 menu
```

File speciali

Ogni device di I/O viene visto a tutti gli effetti come un file:

– **A blocchi**: associato a dispositivi che presentano blocchi di informazione accessibili direttamente (es. dischi)
– **A caratteri**: associato a dispositivi che presentano un flusso di caratteri in ingresso o in uscita (es. terminali, stampanti)

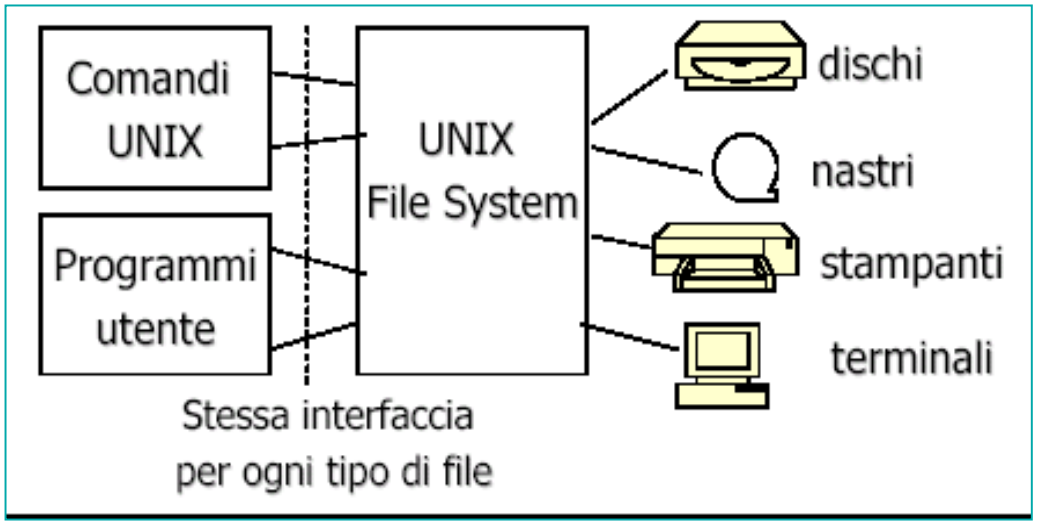
Richieste di lettura/scrittura da/su file speciali causano operazioni di i/o sui device associati
- tutte le operazioni di I/O relative ai dispositivi fisici vengono effettuate applicando le normali primitive definite per i file ordinari

debian:~# echo ciao > /dev/lp0

> provoca una ridirezione nel file /dev/lp0 (la stampante locale) di quanto *echo ciao* manderebbe allo stdout

File speciali: vantaggi

trattamento uniforme di file e device, e indipendenza: programmi portabili e facilmente interfacciabili con ogni tipo di device



File e directory: permessi

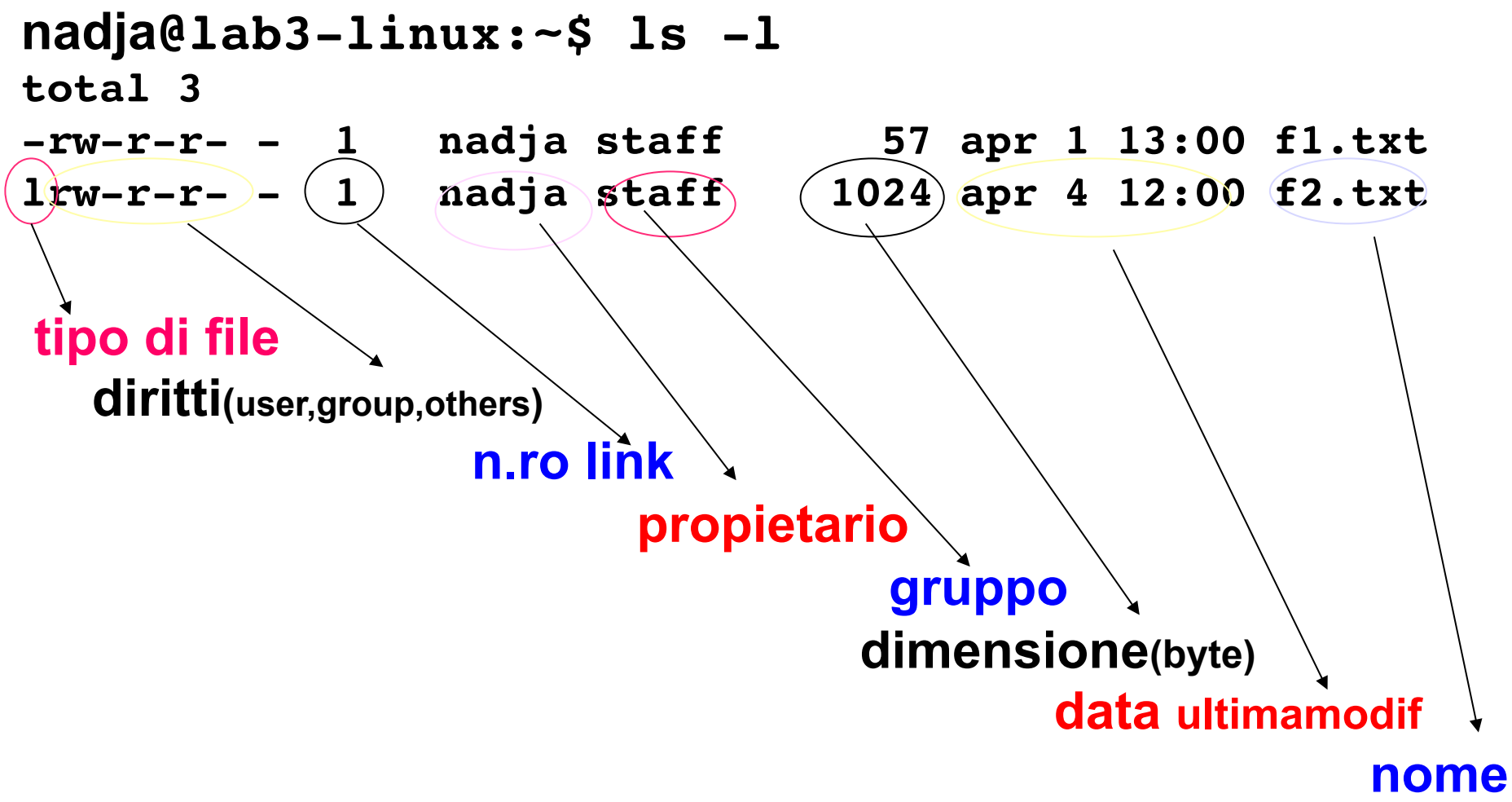
Ad un file possono essere attribuiti i seguenti permessi:

–Lettura (r-ead)
–Scrittura (w-rite)
–Esecuzione (e-x-ecute)

I permessi sono definiti per:

– user proprietario
– group cui appartiene il proprietario
– others

Esempio: ls


nadja@lab3-linux:~$ ls -l
total 3
-rw-r-r- - 1 nadja staff 57 apr 1 13:00 f1.txt
lrw-r-r- - 1 nadja staff 1024 apr 4 12:00 f2.txt
tipo di file
diritti(user,group,others)
n.ro link
propietario
gruppo
dimensione(byte)
data ultimamodif
nome

Al momento della creazione di un file o di una dir vengono assegnati i permessi di lettura e scrittura al proprietario

Es.

```
~ nicole$ ls > ls_home.txt
```

Creo un file con il contenuto della mia home

```
~ nicole$ cat ls_home.txt
    Desktop
    Documents
    Downloads
    ls_home.txt
```

per visualizzarne il contenuto

```
~ nicole$ ls –l
total 5304
drwx------+ 27 nicole  staff      918 Mar 30 09:10 Desktop
drwx------+ 48 nicole  staff     1632 Mar 29 14:58 Documents
drwx------+  4 nicole  staff      136 Mar 29 17:50 Downloads
-rw-r--r--   1 nicole  staff      158 Mar 30 09:25 ls_home.txt
```

l'opzione –l mi fornisce info complete sui file

File e directory: permessi

Per i file ordinari:

r: leggerne il contenuto
w: modificarlo
x: eseguirlo (se è un prog)

Per i file speciali:

r: leggere dal device
w: scrivere sul device
x: non significativo

Per le directory:

r: leggere il contenuto (es: *ls* se x abilitato)
w: modificarla o rimuovere files (es: *mv* se x abilitato)
x: accesso alla directory per leggere, modificare, eseguire un file in essa contenuto

NB: i permessi definiti su un file dipendono dai permessi della directory che lo contiene

chmod

chmod `[ugoa][+-][rwx] filename`

Permette di cambiare i permessi di accesso (lettura, scrittura, esecuzione) su un file; il comando può essere eseguito solo dall'utente proprietario (o da root)

	Permessi			
	Proprietario	Gruppo	Altri utenti	
	rwx	rwx	rwx	
Permessi prima del comando chmod →	rw-	--	--	Simboli dei permessi
Permessi dopo il comando chmod g+rw →	rw-	rw-	--	g+rw
Permessi dopo il comando chmod o+r →	rw-	rw-	r--	o+r

```
$ ls -l dati
-rw------- 1   alice   meteo   207   Feb   20   11:55   dati
$ chmod g+rw dati
$ chmod o+r dati
$ ls -l dati
-rw-rw--r-- 1  alice   meteo   207   Feb   20   11:55   dati
```

chmod

Simboli	*Binario*	*Ottale*
- - -	0 0 0	0
- - x	0 0 1	1
- w -	0 1 0	2
- w x	0 1 1	3
r - -	1 0 0	4
r - x	1 0 1	5
r w -	1 1 0	6
r w x	1 1 1	7

Data la corrispondenza qui affianco, possiamo impostare i permessi utilizzando 3 numeri ottali, risp. per *u*, *g* e *o*

Esempio:

```
$ chmod 544 dati
$ ls -l dati
-r-xr--r— 1 clemente sis2 2 Nov 3 9:30 dati
```

Esempio:

```
$ chmod 750 lettere/grazie
$ ls -ld grazie
drwxr-x-- 1 clemente sis2 2 Nov 3 9:30 grazie
```

chown

chown newuserid file...

change **own**er: cambia l'utente propietario di un file
L'utente identificato da newuserid diventa il nuovo proprietario del file
Il comando può essere eseguito solo dal proprietario "cedente" (o dal superuser)

chgrp

chgrp newgid file...

change **gr**ou**p**: cambia il gruppo propietario di un file
come sopra

touch

touch [options][time] filename…

- aggiorna la data e l'ora dell'ultimo accesso (opzione -a) o
- dell'ultima modifica (opzione -m) di filename (default: -am)
- se time non è specificato, usa la data e l'ora corrente
- se il file non esiste, lo crea

```
%touch 01281738 file1
% ls -l
total 0
----r--r-- 1 user11 usrmail 0 Jan 28 17:38 file1
```

FILTRI

Un filter è un programma che ha stdin e stdout (e magari anche stderr) per default. I filtri si combinano bene con i ***pipe***.

✓ **SED** **editor non interattivo basato sulla individuazione del testo da modificare tramite le espressioni regolari fornite dall'utente.**

```
debian:~# sed 's/ba/ma/' bari
mari
```

✓ **AWK** **linguaggio di programmazione interpretato, nella generazione di report, nella manipolazione di testi, ecc… è nato fondamentalmente per l'analisi e la rielaborazione di file di testo organizzati in una qualche forma tabellare.**

Un programma AWK è composto fondamentalmente da *regole, che stabiliscono il comportamento da prendere nei confronti dei dati in ingresso.*

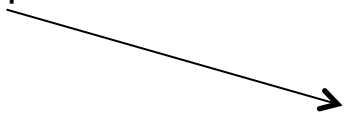
<criterio-di-selezione> { <azione> }

```
Es:   awk 'begin { fs=":"} $7 == "/usr/bin/csh" {csh++}END \
                  {print "utenti con shell:",csh} etc/passwd
```

✓ **GREP** **ricerca sullo standard input l'occorrenza di una espressione regolare fornita dall'utente e, se trova una linea che concorda con questa, la copia sullo standard output**

```
grep "grep" prova_grep.txt
```

del comando grep



file di prova
per verificare
il funzionamento
del comando grep

- **FIND** effettua la scansione ricorsiva della directory alla ricerca di un file che soddisfi l'espressione riportata(già visto)

- **SORT** effettua l'ordinamento di uno o più file di testo sullo standard output

- **VI** editor di testo

Editor vi

Il ***vi*** è l'editor di linee di testo standard per UNIX, è presente in tutte le versioni base e funziona con qualsiasi terminale a caratteri.
Permette di visualizzare una schermata alla volta di un file.
Permette di indirizzare il cursore con appositi comandi al punto (carattere, parola, frase, paragrafo, riga, ...) dove si vuole inserire o aggiornare il testo .
L'uso di ***vi*** è difficoltoso all'inizio, soprattutto per chi è abituato a usare editor grafici
NOTA: i comandi di *vi* sono case sensitive

Prevede diverse modalità operative:

- ***comando,***
- ***testo (inserimento/sostituzione),***
- ***editor di linea***

… Modalità Operative …

Command Mode:

- il cursore è posizionato sul testo,
- la tastiera è utilizzabile solo per richiedere l'esecuzione di comandi, e non per introdurre testo,
- i caratteri digitati non vengono visualizzati.

Input Mode:

- è la modalità che permette di inserire testo
- tutti i caratteri digitati vengono visualizzati ed inseriti nel testo.

Directive Mode:

- è la modalità che permette il controllo del file
- ci si trova posizionati con il cursore nella linea direttive (l'ultima linea del video)
- in questa modalità è possibile invocare comandi della shell digitando `!comando_shell`

vi

*Eseguendo il comando **vi** si entra in un ambiente di editing. L'area di lavoro è completamente dedicata al testo da elaborare, ad esclusione dell'ultima riga in basso che mantiene, di volta in volta, indicazioni sullo **stato** in cui si trova l'editor.*

vi è stato progettato in modo da distinguere anche concettualmente l'***immissione del testo*** dall'***interpretazione di comandi*** che operano sul testo inserito.

In qualsiasi momento, per poter inserire un comando è sufficiente tornare alla modalità comandi premendo il tasto ***Esc***.
Inoltre il tasto ***Esc*** consente di annullare un comando non completato.

Apertura di un file

L'esecuzione del comando ***vi*** senza argomenti crea un nuovo buffer vuoto. Per editare un file esistente possiamo invocare ***vi*** seguito dal nome del file da editare:

vi nomefile

Se il file *nomefile* non esiste, ***vi*** ne crea uno nuovo**.**

Inserimento del testo

Con il comando ***i*** si passa alla modalità di inserimento del testo;

Solitamante questa modalità è identificata dalla presenza della stringa – ***INSERT*** – nella barra di stato.

Il carattere ***~ (tilde)*** rappresenta linee vuote.

Passaggio alla Modalità Testo

Per passare dalla modalità comando alla modalità testo si possono usare i seguenti comandi:

- **`i`** permette di *inserire* il testo a sinistra della posizione corrente del cursore
- **`I`** permette di *inserire* il testo all'inizio della linea in cui si trova il cursore
- **`a`** permette di *appendere* il testo a destra della posizione corrente del cursore
- **`A`** permette di *appendere* il testo alla fine della linea in cui si trova il cursore
- **`R`** permette di inserire testo in modalità di sovrascrittura

Modifica del testo

Il comando ***r*** attiva la modalità modifica del testo, in cui il testo inserito sovrascrive quello esistente.

L'attivazione di tale modalità è identificata dalla presenza della stringa **– REPLACE –** nella barra di stato.

Cancellazione del testo

Per cancellare del testo è possibile utilizzare il comando ***x*** che cancella il carattere identificato dal cursore,

oppure il comando ***dd*** preceduto da un numero ***n*** che cancella ***n*** righe a partire da quella in cui si trova il cursore.

Taglia, copia e incolla

Il comando ***yy***, eventualmente preceduto da un numero, copia in un buffer ***n*** righe a partire da quella in cui si trova il cursore.

Per incollare si usano i comandi ***p*** oppure ***P***. Il primo incolla a partire dalla riga seguente a quella in cui si trova il cursore, il secondo a partire dalla riga stessa in cui si trova il cursore.

La funzione taglia in realtà viene svolta dal comando ***dd*** che non si limita a cancellare una riga, ma a copiarla in un buffer.

Ricerche nel testo

Per cercare una stringa all'interno di un file di testo si usa il comando **/**

(che svolge la stessa funzione anche in programmi quali man, less, more, ecc.).

Dopo la pressione del tasto **/** è possibile digitare la stringa da cercare che apparirà nella barra di stato.

Per ripetere l'ultima ricerca effettuata si usa il comando ***n.***

Salvataggio e chiusura

Per salvare il testo editato si usa il comando ***:w***,

mentre per uscire dal programma si usa il comando ***:q***

*I due comandi possono esseri combinati (**:wq**) per salvare ed uscire nello stesso momento.*

Inoltre e' possibile forzare l'uscita **senza salvare** (con conseguente perdita delle modifiche), con il comando ***:q!***

Movimento del Cursore …

Per muoversi nel file si possono usare i seguenti comandi (in command mode):

- `h` permette di spostarsi di un posto a *sinistra*
- `l` permette di spostarsi di un posto a *destra*
- `k` permette di spostarsi di una riga in *alto*, sulla stessa colonna
- `j` permette di spostarsi di una riga in *basso*, sulla stessa colonna
- talvolta si possono anche usare le frecce della tastiera

- `G` permette di spostarsi sull'ultima linea del testo
- `#G` permette di spostarsi sulla linea identificata dal numero #
- `:#` stesso comportamento di `#G`
- `^D` permette di spostarsi mezza pagina in avanti
- `^U` permette di spostarsi mezza pagina indietro
- `^F` permette di spostarsi sulla pagina successiva
- `^B` permette di spostarsi sulla pagina precedente

È possibile ripetere il generico comando di movimento `<move_cmd>` con il comando

- `#<move_cmd>`

… Movimento del Cursore

- `$` permette di posizionarsi *alla fine* della linea corrente
- `^` permette di posizionarsi sul primo carattere non blank della linea corrente
- `0` permette di posizionarsi *all'inizio* della linea corrente
- `#|` permette di posizionarsi alla colonna #
- `+` di posizionarsi sul primo carattere non blank della linea successiva
- `-` di posizionarsi sul primo carattere non blank della linea precedente

Visualizzazione del Testo

- `^E` permette di visualizzare la schermata successiva senza spostare il cursore
- `^Y` permette di visualizzare la schermata precedente senza spostare il cursore
- `^G` permette di visualizzare informazioni di varia natura relative al file corrente
- `:=` permette di visualizzare il numero di linee del file
- :.= permette di visualizzare il numero della linea corrente
- `^L` realizza il refresh del video

NOTA: per visualizzare sempre il numero di linea si usa il comando

- `:set number`

Gestione delle Linee

- `o` permette di inserire una linea vuota *al di sotto* di quella corrente
- `O` permette di inserire una linea vuota *al di sopra* di quella corrente
- `J` permette di concatenare la linea successiva alla linea corrente

Gestione delle Parole

In ambiente ***vi***, per *parola* si intende una qualsiasi sequenza di caratteri chiusa da un carattere non alfanumerico.

Per gestire le parole sono disponibili i seguenti comandi:

- `w` permette di posizionarsi all'inizio della parola successiva
- `e` permette di posizionarsi alla fine della parola successiva
- `b` permette di posizionarsi all'inizio della parola precedente

Undo

I comandi per l'annullamento sono i seguenti:

- `u` permette di annullare l'*ultima* modifica effettuata
- `U` permette di annullare *tutte* le modifiche effettuate sulla linea corrente a partire dal momento in cui il cursore è stato posizionato su di essa

Buffer …

I comandi per gestire i buffer sono i seguenti:

- `x`, permette di cancellare il carattere su cui è posizionato il cursore e di inserirlo nel buffer
- `X`, permette di cancellare il carattere precedente quello su cui è posizionato il cursore e di inserirlo nel buffer
- `dw`, permette di cancellare tutti i caratteri da quello corrente fino alla fine della parola e di inserirli nel buffer
- `D`, permette di cancellare tutti i caratteri da quello corrente fino alla fine della linea e di inserirli nel buffer
- `dd`, permette di cancellare tutta la linea corrente e di inserirla nel buffer

… Buffer …

L'operazione di copia di un testo nel buffer si chiama *yank*

- `yl` permette lo yank del carattere a sinistra della posizione corrente
- `yh` permette lo yank del carattere a destra della posizione corrente
- `yw` permette lo yank della parola, o parte di parola, a sinistra della posizione corrente
- `yb` permette lo yank della parola, o parte di parola, a destra della posizione corrente
- `Y` permette lo yank di tutta la linea corrente
- `Yf<`*`char`*`>` permette lo yank di tutto il testo fino a quando si trova il carattere `<`*`char`*`>`

… Buffer …

- `P` permette di inserire alla sinistra della posizione corrente il testo più recentemente inserito nel buffer
- `p` permette di inserire alla destra della posizione corrente il testo più recentemente inserito nel buffer

NOTA: se il testo più recentemente inserito nel buffer è un'intera linea `P` *e* `p` *inseriscono rispettivamente una linea prima e dopo la linea corrente*

In ***vi*** vi sono diversi buffer, e precisamente:

- Nove buffer numerati (1, 2, …, 9) gestiti automaticamente
- 26 buffer identificati dai caratteri alfabetici (a, b, …, z) gestiti dall'utente

Il testo che viene cancellato o copiato nel buffer mediante yank viene inserito nel buffer 1, così facendo il precedente contenuto del buffer 1 viene spostato nel buffer 2, e così via. Il precedente contenuto del buffer 9 viene eliminato

Ricerche

Per cercare una stringa si usa uno dei comandi:

- `/<str>` permette di cercare la stringa `<str>` in avanti a partire dalla posizione corrente
- `?<str>` permette di cercare la stringa `<str>` indietro a partire dalla posizione corrente

In entrambi i casi, se la stringa viene trovata il cursore si posiziona sul suo inizio.

Se la stringa non viene trovata viene visualizzato un messaggio di errore

Sostituzione …

Per sostituire una stringa con un'altra sono disponibili i seguenti comandi:

- ~ permette di sostituire il carattere corrente minuscolo con il corrispondente maiuscolo e viceversa
- S<*new_str*><ESC> permette di sostituire l'intera linea corrente con la stringa <*new_str*>
- C<*new_str*><ESC> permette di sostituire la linea corrente a partire dalla posizione attuale sino alla fine con la stringa <*new_str*>
- S<*new_str*><ESC> permette di sostituire l'intera linea corrente con la stringa <*new_str*>
- cw<*new_str*><ESC> permette di sostituire la parte di parola dalla posizione corrente sino alla fine con la stringa <*new_str*>
- cb<*new_str*><ESC> permette di sostituire la parte di parola dalla posizione corrente sino all'inizio con la stringa <*new_str*>
- :s/<old_str>/<*new_str*>/*g* permette di sostituire il testo <*new_str*> al testo <old_str> tutte le volte che questi compare nella linea corrente
- :s/<old_str>//*g* permette di sostituire tutte le occorrenze di <old_str> nella linea corrente con la stringa vuota, cioè permette di cancellare <old_str>
- :<start_line>,<end_line>/<old_str>/<*new_str*>/*g* permette di sostituire il testo <*new_str*> al testo <old_str> tutte le volte che questi compare nella porzione di file compresa tra la linea numero <start_line> e la linea numero <end_line>

Altre Caratteristiche …

Quando il sistema va in crash e eventuali modifiche a un `<nome_file>` non sono state salvate è bene riaprire il file con il comando

- `vi -r <nome_file>`

per recuperare i cambiamenti non memorizzati nella sessione interrotta

È possibile richiedere l'esecuzione di un comando `<cmd>` immediatamente dopo l'apertura del file `<nome_file>` invocando vi nel modo seguente

- `vi -r +<cmd> <nome_file>`

È possibile aprire con un'unica invocazione di vi più file con il comando

- `vi <nome_file1> <nome_file2> …`

Comunque uno e un solo file può essere attivo in un dato istante

Il primo file attivo è il primo indicato sulla linea di comando di invocazione di vi

Per accedere ai file successivi, secondo l'ordine indicato si usa il comando

- `:n`

Uscita da vi

Per uscire da vi si possono usare i seguenti comandi:

- `ZZ` (da command mode) permette di *uscire* da vi *salvando* le modifiche
- `x` (da directive mode) permette di *uscire* da vi *salvando* le modifiche
- `q` (da directive mode) permette di *uscire* se non ci sono state modifiche
- `q!` (da directive mode) permette di *uscire* da vi *senza salvare* le modifiche

Riepilogo vi

- ***vi nomefile***, apre il file in ***vi***
- ***Esc*** torna alla modalità comandi
- ***:e*** (edit) apre un file
- ***i*** o ***Ins*** inserisce del testo
- ***r*** o due volte ***Ins*** sovrascrive il testo
- ***[n] dd*** taglia n righe
- ***[n] yy*** copia n righe
- ***p*** incolla dopo la riga corrente
- ***P*** incolla a partire dalla riga corrente
- ***/*** cerca all'interno del testo
- ***n*** ripete l'ultima ricerca
- ***:w*** salva
- ***:q*** chiude
- ***:q!*** chiude senza salvare

Esercizi

Utilizzare tutti i comandi visti finora per effettuare le seguenti operazioni:

- Creare una cartella dentro la vostra home
- Creare una serie di file nella cartella creata contenenti:
  1. Il contenuto delle sottodirectory della home
  2. Il contenuto della home con i file nascosti
  3. Come in 2 ma in ordine alfabetico inverso
- Creare nella home direcotry un link simbolico ad uno dei file creati
- Elencare tutti i file della cartella creata che iniziano per “h”, e poi togliergli i diritti di scrittura
- Utilizzare il manuale per avere info sui comando “rm” e “ps”, poi
  - Cancellare un file tra quelli creati
  - Visualizzare i processi del sistema e
    - ordinarli per PID;
    - salvare in un file solo quelli del vostro account
    - sovrascrivere il file conl’elenco dei processi ordinati per PID

Soluzioni

- mkdir /home/studente/prova, WD: ~ → mkdir prova è sufficiente
- ls /home > subhomels, WD: ~/prova
  - aggiungere l'opzione -R per renderlo ricorsivo
- ls -la > homela
- ls -lar >homelar
- ln -s ~/prova/homela ~/link, WD: ~
- ls h* | chmod a –w oppure ls h* |find -type f -exec chmod 444 {} \; WD: ~/prova
- rm homela
- ps -eF --sort=pid
- ps -u root u > fileps
- ps -eF --sort=pid > fileps

~ è la dir /home/studente

Un suggerimento

BackTrack

- http://www.backtrack-linux.org/downloads/
- nella sezione ‘how to’ (http://www.backtrack-linux.org/tutorials/) trovate le istruzioni su come installare BackTrack live su USB (opzione consigliata)
- se preferite potete creare una partizione e installare la distribuzione che preferite sulla vostra macchina