

# **Laboratorio di Sistemi Operativi**

**Corso di Laurea in Informatica**

*A.A. 2013-2014*

Alberto Finzi

# Introduzione

Il Sistema Operativo (SO) è lo strato software che controlla le risorse Hardware e fornisce l'ambiente nel quale vengono eseguiti i programmi (kernel).

In senso più largo si considera parte dell'SO anche altro software: utility, shell, applicazioni, librerie etc.

Tutti i SO forniscono servizi. Eseguire un nuovo programma, aprire un file, allocare la memoria sono esempi di servizi forniti da un sistema operativo.

Descriveremo servizi forniti da varie versioni del sistema operativo Unix.

# Il Sistema Operativo Unix

1965 Bell Laboratory (con MIT e General Eletrics) lavora ad un nuovo sistema operativo: Multics. Principali caratteristiche: capacità multi-utente (multi-user), multi-processo (multi-processor) e un file system multi-livello (gerarchico) (multi-level file system).

1969 AT&T abbandona Multics. Ken Thompson, Dennis Ritchie, Rudd Canaday e Doug McIlroy, progettano e implementano la prima versione del file system Unix insieme ad alcune utility. Il nome Unix è di Brian Kernighan: gioco di parole su Multics.

1 Gennaio 1970 Inizio di Unix.

Nel 1973 Unix riscritto in C (Dennis Ritchie)

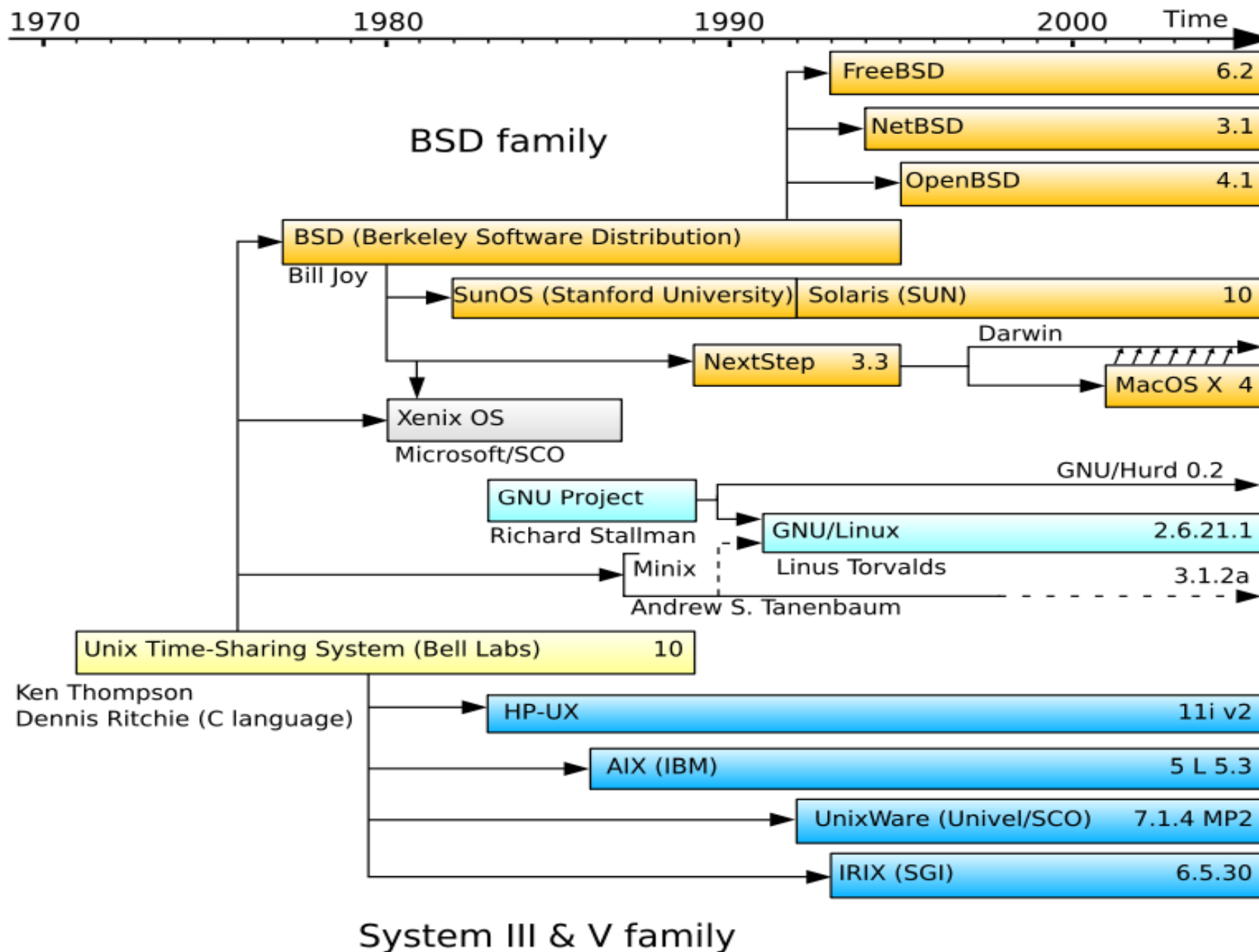
Caratteristiche:

- Architettura semplice ed elegante,
- Ambiente di programmazione (Implementato in C),
- Indipendenza dall'Hardware.

In particolare, SO basato su **kernel**

- il nucleo del SO è l'unica porzione che deve essere adattata all'Hw;
- tutte le altre funzioni poggiano sul nucleo (indipendenti Hw).

# II Sistema Operativo Unix



# Architettura Sistema Unix

- Il **nucleo** del sistema (**kernel**) gestisce le risorse essenziali: la CPU, la memoria, le periferiche
- Tutto il resto, anche l'interazione con l'utente, è ottenuto tramite programmi eseguiti dal kernel, che accedono alle risorse hardware tramite delle richieste a quest'ultimo.

Il kernel è il solo programma ad essere eseguito in modalità privilegiata, con il completo accesso all'hardware, gli altri programmi vengono eseguiti in modalità protetta.

# Architettura di Sistema

Il kernel si occupa di:

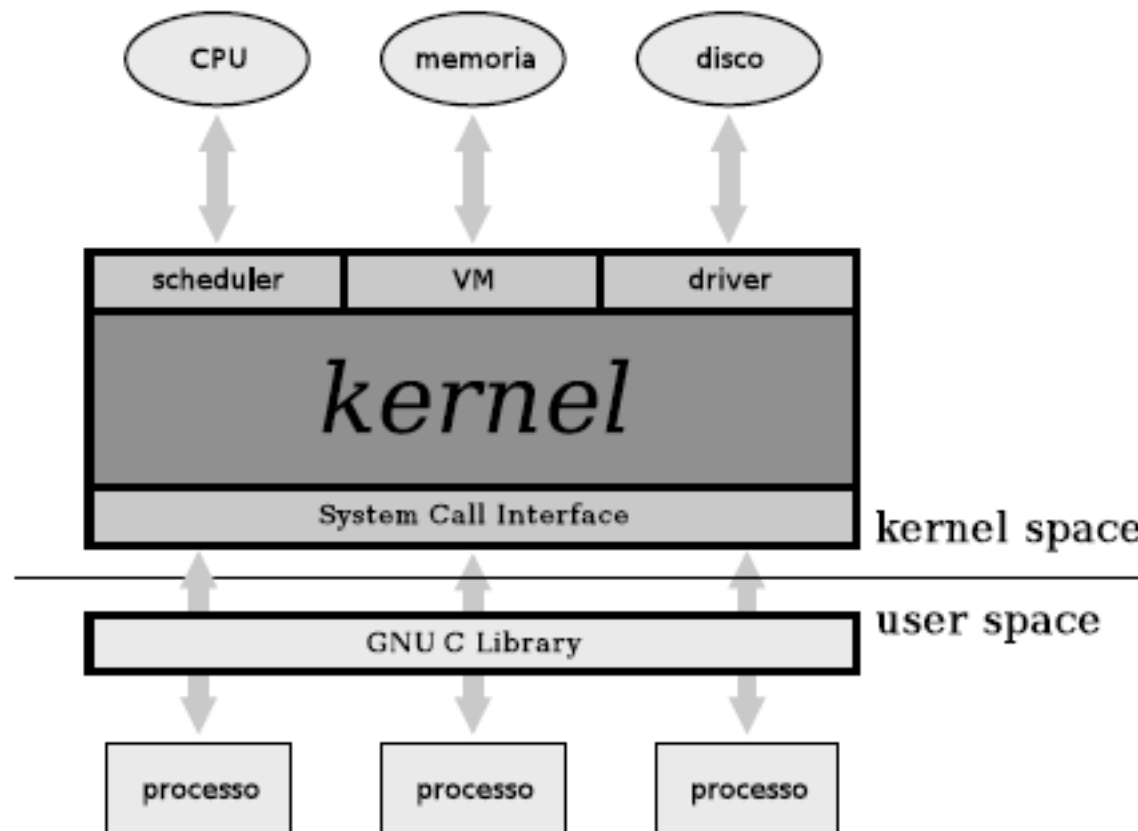
**CPU:** Una parte del kernel, lo scheduler , si occupa di stabilire, ad intervalli fissi e sulla base priorità, quale “processo” deve essere mandato in esecuzione

**Memoria:** La memoria viene gestita dal kernel attraverso il meccanismo della memoria virtuale, che consente di assegnare a ciascun processo uno spazio di indirizzi “virtuale” che il kernel, con l’ausilio della unità di gestione della memoria, rimappa automaticamente sulla memoria disponibile.

**Periferiche:** Le periferiche vengono viste attraverso un’interfaccia astratta che permette di trattarle come fossero file, secondo il concetto per cui “everything is a file” (le interfacce di rete fanno eccezione).

# Architettura di Sistema

Le interfacce con cui i programmi possono accedere all'hardware vanno sotto il nome di chiamate al sistema (system call ): un insieme di funzioni che un programma può chiamare, per le quali viene generata un'interruzione del processo passando il controllo dal programma al kernel.



Queste chiamate al sistema vengono rimappata in funzioni definite dentro opportune librerie (Libreria Standard del C).

# Sistema Multi-utente

- Il kernel Unix/Linux nasce fin dall'inizio come sistema multiutente.
- Il concetto base è quello di utente (user) del sistema, sono previsti *meccanismi* per identificare gli utenti, *permessi* e *protezioni* per impedire che utenti diversi possano danneggiarsi a vicenda o danneggiare il sistema.
- in ogni Unix è presente un utente speciale privilegiato, superuser, il cui username è di norma root, ed il cui uid è zero. Esso identifica l'amministratore del sistema.



# Unix Shell

- La shell è un interprete di comandi
- Si interpone tra l'utente ed il sistema operativo
- In sistemi Unix qualsiasi operazione può essere eseguita da una sequenza di comandi shell

Tra le Shell più utilizzate ci sono:

- |                       |           |             |
|-----------------------|-----------|-------------|
| • Bourne shell,       | /bin/sh   | (Bell Labs) |
| • Bourne-again shell, | /bin/bash | (Linux)     |
| • Cshell,             | /bin/csh  | (Berkeley)  |
| • Korn shell,         | /bin/ksh  | (Bell Labs) |
| • TENEX C shell       | /bin/tcsh | (BBN tech)  |

# Classi di Comandi

- reperire informazioni su comandi, programmi, file....
- gestire filesystem
- operare su file e directory
- elaborare testi
- sviluppare software
- operare in remoto
- ....(comandi “di utilità” vari)
- amministrare il sistema
  - utenti e gruppi
  - dispositivi
  - software

# Formato dei comandi

**comando [ argomento ...]**

Gli argomenti possono essere:

- opzioni o flag (-)
- parametri

separati da almeno un separatore (di default il carattere spazio)

L'ordine delle **opzioni e'**, in genere, irrilevante

L'ordine dei **parametri e'**, in genere, rilevante

**Attenzione: Unix e' CASE SENSITIVE**

# Formato dei comandi

## Comandi Equivalenti:

ls -l -F file1 file2 file3

ls -lF file1 file2 file3

ls -F -l file1 file2 file3

## Comandi NON equivalenti:

cat file1 > file2

cat file2 > file1

ls -l -F file1

ls -f -L file1

## Comandi ERRATI:

LS -F -L

CAT file1 > file2

# Esempi di Comandi

Comando	Significato
<b>ls</b>	visualizza la lista di file nella directory, come il comando <b>dir</b> in DOS
<b>cd directory</b>	cambia directory
<b>passwd</b>	cambia password
<b>file filename</b>	visualizza il tipo di file o il tipo di file con nome filename
<b>cat textfile</b>	riversa il contenuto di textfile sullo screen
<b>pwd</b>	visualizza la directory di lavoro corrente
<b>exit</b> or <b>logout</b>	lascia la sessione
<b>man <i>command</i></b>	leggi pagine manuale su <b>command</b>
<b>info <i>command</i></b>	leggi pagine Info su <b>command</b>
...	

# Gestione delle Directory

`mkdir`

Crea una directory

`rmdir`

Elimina una directory vuota

`pwd`

Emette il percorso della directory corrente

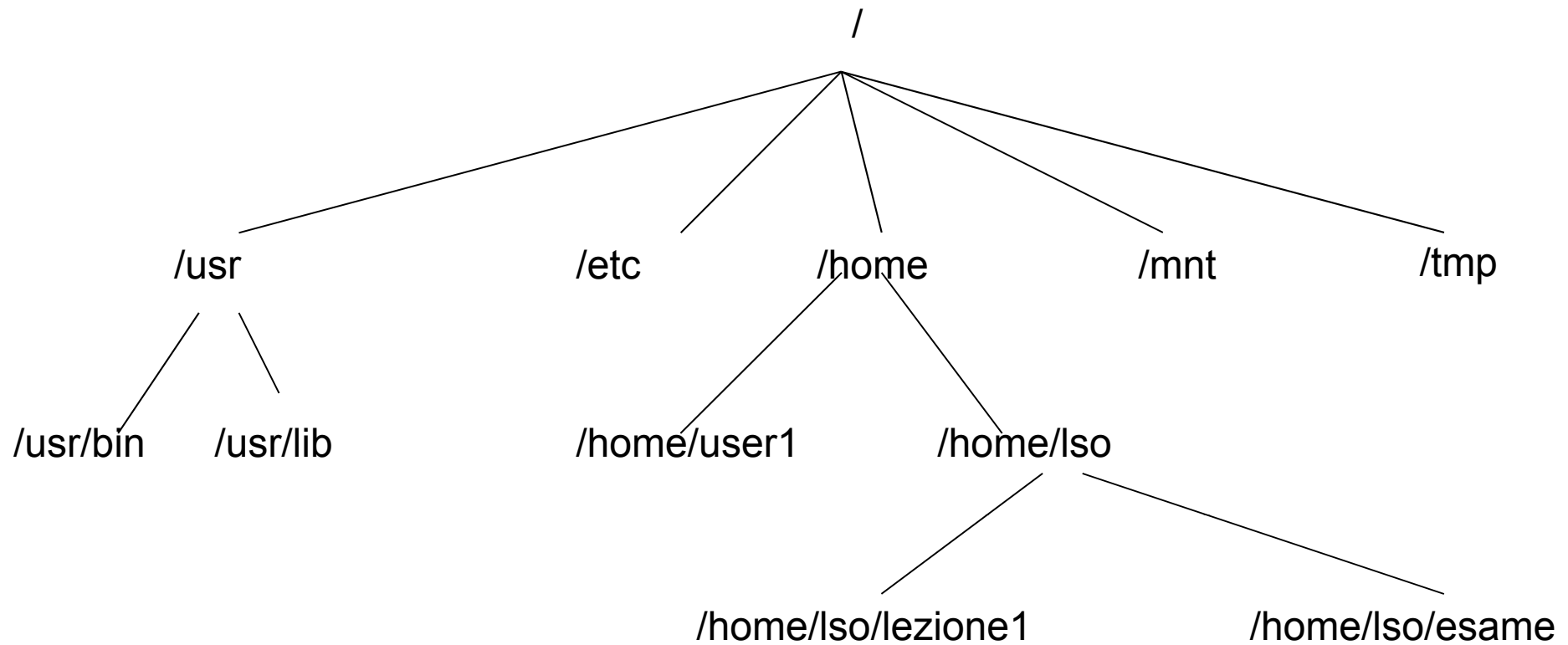
`ls`

Elenca il contenuto di una o più directory

`du`

Calcola lo spazio utilizzato da una serie di directory e subdirectory

# File System di UNIX



# mkdir (make directory)

`mkdir [options] directory`

Crea una directory secondo le opzioni

Alcune opzioni:

`-m mode` Indica le protezioni per la directory da creare

`-p` Crea, se assente, l'intero percorso indicato

Esempi:

```
mkdir -m 600 tracce_esami
```

```
mkdir -p Iso/11-12/lezione1/
```



# rmmdir (remove directory)

`rmmdir [options] directory`

Rimuove la directory indicata.

Le directory possono essere rimosse se

(a) sono vuote e

(b) possibile scrivere nella directory padre.

Opzioni:            -p     Rimuove l'intero percorso indicato

Esempi:

Crea la directory lezione1 nella directory corrente

```
rmmdir lezione1
```

Rimuove, dalla directory corrente, il path indicato

```
rmmdir -p Iso/11-12/lezione1/
```

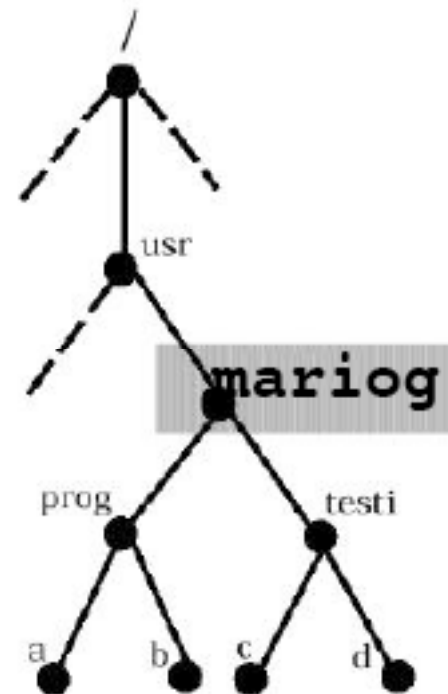
# pwd (print working directory)

**pwd** "printworking directory"

stampa il path della directory corrente

Esempio:

```
% pwd  
/usr/mariog  
%
```



# cd (change directory)

`cd [directory]`

Cambia la directory corrente a quella indicata.

Se non viene passato nessun argomento, la directory corrente diventa la home directory.

Esempio:

```
lso:~>cd
```

```
lso:~>pwd
```

```
/home/lso
```

```
lso:~>cd esempio/esempiocd
```

```
lso:~>pwd
```

```
/home/lso/esempio/esempiocd
```

# ls (list)

```
ls [options] [directory]
```

Elenca i file contenuti nella directory specificata.

Se la directory non e' indicata, viene elencato il contenuto della directory corrente.

Alcune opzioni:

- a Elenca anche i file nascosti
- l Formato esteso con informazioni su modo, proprietario, dimensione, etc dei file
- s fornisce la dimensione in blocchi dei file
- t lista i file nell'ordine di modifica (prima il file modificato per ultimo)
- 1 elenca i file in una singola colonna
- F aggiunge / al nome delle directory e \* al nome dei file eseguibili
- R si chiama ricorsivamente su ogni subdirectory

# ls – campi del formato esteso

Totale dimensione occupata (in blocchi)

Riferimenti al file

Dimensione (byte)

Nome

```

ls:~>ls -l
total 12
-rw-rw-r-- 1 lso lso 10 Mar 4 13:29 a
-rw-rw-r-- 1 lso lso 10 Mar 4 14:12 b
drwxrwxr-x 2 lso lso 4096 Mar 4 14:29 c
  
```

Proprietario Gruppo primario Data ultima modifica

Tipo

Permessi

(r)ead, (w)rite, e(x)ecute, (s)et uid bit

(d)irectory, (l)ink, (c)haracter special file, (b)lock special file, (-) ordinary file

# du (disk usage)

`du [options] file ...`

visualizza informazioni sull'utilizzo dello spazio (blocchi) da parte del/i file in esame. Il file puo' essere ordinario o un directory.

Se il file e' una directory, viene visualizzato, ricorsivamente, lo spazio utilizzato dalle sotto-directory

Alcune opzioni:

- s Visualizza lo spazio complessivamente occupato dal file in esame.
- k Visualizza lo spazio utilizzato espresso in Kb.

# Gestione dei File

cp	Copia un file in un altro
ln	Crea un collegamento
mv	Rinomina/sposta un file
rm	Cancella un file
chown	Cambia proprietario e gruppo primario di un file
chmod	Cambia i permessi di un file
touch	Cambia il tempo di ultimo accesso ad un file
file	Determina il tipo di file
find	Ricerca file in base ad opportuni criteri, permettendo di eseguirvi (opzionalmente) delle operazioni
tar	archivia un file
gzip	Comprime (Decomprime) un file

# cp (copy)

```
cp [options] source... target
```

copia un file in un altro file oppure uno o più file in una directory

Se vengono specificati solo i nomi di due file, il primo viene copiato sul secondo

Se vengono specificati solo due nomi, e se il secondo nome indicato è una directory, source viene copiato con lo stesso nome nella directory target. Se source è una directory, la copia avviene solo con opzioni particolari.

Se vengono indicati più di due nomi, il file target deve essere una directory e vengono generate le copie dei source in target. In mancanza di opzioni particolari, le directory non vengono copiate.

Alcune opzioni

- r se source e target sono directory, copia ricorsivamente source, i suoi file e le sue subdirectory in target
- i opera in modo interattivo, chiedendo una conferma se la copia comporta la cancellazione di un target preesistente



# Esempi

- Copia il file pippo nella directory corrente nel file /tmp/pippo.back

```
cp pippo /tmp/pippo.back
```

- Copia il file /tmp/pippo.back e la directory dir nella directory nuovadir

```
cp -r /tmp/pippo.back dir nuovadir
```

- Copia il file pippo nel file pippo2

```
cp pippo pippo2
```

# mv (move)

```
mv [options] source... destination
```

rinomina (sposta) file o directory.

Se vengono specificati solo i nomi di due elementi, source viene rinominato in destination, oppure in destination/source, a seconda che destination indichi un file o una directory. Qualora destination denoti un file preesistente, questo non sarà più accessibile come tale, e non sarà più accessibile in alcun modo se destination era il suo unico nome.

Se vengono indicati più di due elementi, destination deve essere una directory, e source\_1...source\_n vengono rinominati come destination/source\_1...destination/source\_n.

Nel caso che source e destination appartengono a due diversi file system, il comando effettua un vero e proprio spostamento dati tra i due file system. In tal caso vengono spostati solo i file ordinari, quindi: né collegamenti, né directory.

Alcune opzioni:

- i il comando chiede conferma all'utente qualora destination è un file preesistente

# rm (remove)

```
rm [options] file...
```

elimina i file o le directory indicati come argomento.

Alcune opzioni:

-i chiede conferma prima di rimuovere ogni file

-R rimuove ricorsivamente i file e le sottodirectory

Esempi d'uso

Elimina i file pippo nella directory corrente e /tmp/pippo.back

```
rm pippo /tmp/pippo.back
```

Elimina la directory nuovadir e tutto il suo contenuto

```
rm -R nuovadiru
```

Elimina tutti i file nella directory corrente

```
rm *
```

# touch

---

```
touch [options] file...
```

Cambia sia il tempo di ultimo accesso che di ultimo aggiornamento dei file. Per default, il tempo viene aggiornato alla data/ora di sistema al momento dell'invocazione del comando.

Se si specificano file che non esistono, questi vengono creati vuoti.

Alcune opzioni:

- a cambia solo il tempo di ultimo accesso
- c se file non esiste non viene creato
- m cambia solo il tempo di ultima modifica

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

# chown (change owner)

```
chown [options] [user][:[group]] file...
```

Cambia proprietario e/o gruppo primario per uno o più file.

Se dopo “:” non segue il nome del gruppo, viene attribuito il gruppo principale cui appartiene user.

Se prima di :group non viene indicato il nome dell'utente, viene cambiato solo il gruppo primario (chgrp)

Alcune opzioni:

- R opera ricorsivamente su directory e subdirectory
- h se file è un link simbolico, opera su di esso piuttosto che sul file da esso referenziato

# chmod (Change mode)

```
chmod [options] mode file...
```

cambia i permessi sui file indicati come argomento

Il parametro mode può essere espresso sia in forma ottale che simbolica. In quest'ultimo caso, la stringa dei beneficiari ((u)ser,(g)roup,(o)ther) è separata da quella dei permessi ((r)ead,(w)rite,e(x)ecute) tramite i segni + o -, a seconda che i permessi vadano aggiunti o sottratti; mentre stringhe relative a beneficiari/permessi diversi sono separate da virgole, come in: u+rwx,go+rx

Alcune opzioni:

-R opera ricorsivamente su directory e subdirectory

Esempi d'uso:

Attribuisce al file pippo il modo di permessi rwxr-xr-x (111-101-101)

```
chmod 755 pippo
```

Elimina i permessi x per il gruppo ed rx per other per pippo

```
chmod g-x,o-rx pippo
```

# file

---

```
file [options] file...
```

**analizza i file indicati come argomento e cerca di determinarne il tipo**

## Esempi

```
lso:> file test.tar
```

```
test.tar: GNU tar archive
```

```
lso:> file *gif
```

```
logo1.gif: GIF image data, version 89a, 200 x 200
```

```
logo2.gif: GIF image data, version 89a, 241 x 243
```

# find

**find [pathname...] [expression]**

discende ricorsivamente le directory specificate (pathname...), cercando tutti i file che rendono vera expression.

Molto flessibile:

- ricerca file di specificati attributi (nome, tipo, permessi, proprietario, gruppo, numero di link, dimensione, data di ultima modifica/accesso ...)
- and, or, not di attributi
- può eseguire automaticamente, o previa conferma, uno o più comandi sui file individuati
- le espressioni si ottengono combinando flag, parametri e gli operatori booleani;
- le espressioni costituite solo da un flag e da un parametro (opzionale) si dicono espressioni elementari;



# Esempi

Ricerca in /home/lso di file la cui dimensione è maggiore di 100 blocchi

```
find /home/lso -size +100
```

Ricerca dei file modificati meno di due giorni fa e che non sono di root

```
find / -mtime -2 -a ! -user root
```

Rimozione di file il cui ultimo accesso risale a più di 7 giorni

```
find /home/lso -atime +7 -exec rm {} \;
```

Rimozione di file dell'utente lso e che hanno estensione .old oppure .back

```
find . \( '*.old' -o '*.back' \) -user lso -ok rm {} \;
```

# Comandi di utilità

cat

Concatena file

wc

Conta caratteri, parole e linee in un file

cut

Taglia delle colonne opportune da un file di testo

paste

Compone più file di testo

sort

Ordina le linee di un file

diff

Mostra le differenze di contenuto tra due file

grep

Cerca espressioni regolari all'interno di un file

# cat (concatenate)

---

`cat [options] [file...]`

concatena i file indicati come argomento, visualizzandoli attraverso lo standard output

## *Alcune opzioni:*

- n            fa precedere ogni linea di output dal numero progressivo che identifica la            posizione della linea nel file concatenato
- b            come l'opzione precedente, ma omette la numerazione delle linee bianche
- v            mostra anche i caratteri non stampabili, ad eccezione dei caratteri di            tabulazione, nuova linea e ritorno a capo

# wc (Word count)

`wc [options] [file...]`

fornisce il numero dei codici di interruzione di riga (in pratica il numero delle righe), delle parole o dei caratteri contenuti in *file*. Senza opzioni fornisce, nell'ordine suddetto, ciascuna delle precedenti informazioni.

## *Alcune opzioni:*

- `-c` emette solo il numero complessivo di caratteri di *file*.
- `-w` emette solo il numero complessivo di parole in *file*.
- `-l` emette solo il numero di righe in *file*.

## *Esempi di esecuzione*

```
gio$ wc which_manpage
132          239          2083  which_manpage
gio$ wc -c which_manpage
2083  which_manpage
gio$
```

# cut

---

`cut [options] [file...]`

estrae delle colonne specifiche dalle linee di testo che compongono *file*.

## *Alcune opzioni:*

- |                            |  |
|----------------------------|--|
| <code>-c char_list</code>  | definisce gli intervalli da estrarre espressi in caratteri.  |
| <code>-f field_list</code> | definisce gli intervalli da estrarre espressi in campi.<br>I campi sono distinti in base a un certo carattere usato<br>come delimitatore. Quello predefinito è il carattere di<br>tabulazione. |
| <code>-d delimiter</code>  | definisce un delimitatore alternativo al carattere di<br>tabulazione.  |

## *Esempi d'uso*

*Estrae la prima e la quinta colonna del file /etc/passwd*

```
cut -d: -f1,5 /etc/passwd
```

*Estrae i primi dieci caratteri da ogni riga del file /etc/passwd*

```
cut -c1-10 /etc/passwd
```

# paste

`paste [options] file...`

concatena le righe corrispondenti dei file inseriti come argomento, inserendo tra di esse un delimitatore. Il delimitatore di default è il carattere di tabulazione.

## *Alcune opzioni:*

`-d delimiter` definisce un delimitatore alternativo al carattere di tabulazione.

### *Esempi d'uso*

*Concatena ciascuna riga di pippo con la corrispondente di prova*

`paste pippo prova`

*Come prima, ma in luogo di <TAB> è inserito ":" come delimitatore*

`paste -d: pippo prova`

# sort

---

`sort [options] [file...]`

permette di (ri)ordinare o fondere insieme il contenuto dei file passati come parametri, oppure di (ri)ordinare le linee passategli in input. L'ordinamento è la modalità predefinita, ed è effettuato tenendo conto delle opzioni inserite dall'utente. Il comando tratta ogni linea come un insieme ordinato di campi, separati da opportuni caratteri (di default i caratteri di separazione sono la tabulazione e lo spazio). In assenza di opzioni che definiscano diversi criteri di ordinamento, quest'ultimo avviene in base al primo campo ed è alfabetico.

## *Alcune opzioni:*

- `-f` ignora le differenze tra lettere minuscole e maiuscole
- `-n` considera numerica anziché testuale la chiave di ordinamento
- `-r` ordina in senso decrescente anziché crescente
- `-o fileout` invia l'output a fileout anziché sull'output standard
- `-t s` usa s come separatore di campo
- `-k s1,s2` usa i campi da s1 a s2-1 come chiavi di ordinamento

# sort – Esempi d'uso

*Ordina le linee del file /etc/passwd in base al valore del terzo campo (UID)*

```
sort -t: -k3,4 /etc/passwd
```

*Come prima, solo che ora l'ordinamento è numerico anzichè alfabetico*

```
sort -t: -n -k3,4 /etc/passwd
```

*Come prima, ma seguendo l'ordinamento inverso (prima l'UID maggiore)*

```
sort -t: -n -k3,4 -r /etc/passwd
```

*Come prima, ma ora l'output è memorizzato in passwd\_reordered*

```
sort -t: -n -k3,4 -r /etc/passwd -o passwd_reordered
```



# diff (differences)

```
diff [options] file_1 file_2
```

può funzionare con diverse modalità, stabilite in base alle opzioni, per determinare semplicemente i due file passati come parametri sono identici o meno, oppure per indicare le differenze che ci sono tra i due, con maggiore o minore dettaglio di informazioni al riguardo. Il risultato del confronto dei file viene emesso attraverso lo standard output, e di default mostra la lista di cambiamenti da apportare a *file\_1* per renderlo uguale a *file\_2*.

## *Alcune opzioni:*

- l produce l'output in formato esteso
- s segnala se i due file sono identici (per default non è emesso output in tal caso)
- b ignora i "caratteri bianchi", quali TAB e spazi, e considera stringhe di tali caratteri come equivalenti
- i considera equivalenti le lettere maiuscole e minuscole
- r opera ricorsivamente su directory e subdirectory

# grep (get regular expression)

---

```
grep [options] modello [file...]  
grep [options] -e modello [file...]  
grep [options] -f file_modello [file...]
```

esegue una ricerca all'interno dei file indicati come argomento oppure, nel caso che non sia specificato alcuno di tali file, all'interno dello standard input.

Il modello di ricerca può essere semplicemente il primo degli argomenti che seguono le opzioni (prima versione del comando), oppure può essere indicato precisamente come argomento dell'opzione **-e**, oppure ancora può essere contenuto in un file che viene indicato attraverso l'opzione **-f**.

Le opzioni *options* sono relative a modalità di visualizzazione dell'output e degli eventuali messaggi di errore.

## *Alcune opzioni*

<b>-v</b>	restituisce le linee che non contengono <i>modello</i>
<b>-w</b>	restituisce solo le linee che hanno <i>modello</i> come parola completa
<b>-x</b>	restituisce solo le linee che coincidono con <i>modello</i>

*Nota:* Per evitare interpretazioni improprie da parte della shell è opportuno racchiudere l'espressione che definisce il modello di ricerca tra gli apici ' ' qualora esso contenga uno dei metacaratteri mostrati in tabella.

# grep – Esempi d'uso

---

*Fornisce le linee del file `/etc/passwd` che contengono la parola `gio`*

```
grep -w gio /etc/passwd
```

*Come prima, ma ora sono restituite tutte le linee che hanno `gio` come (sub)stringa*

```
grep gio /etc/passwd
```

*Restituisce le linee che finiscono con la lettera `f`*

```
grep /home/gio/pippo f$
```

*Restituisce tutti i file nella directory corrente che non sono directory e che hanno il permesso in esecuzione per il proprietario*

```
ls -al | grep ^-..x
```