

# Concetti di base UNIX

---

Università di Modena e Reggio Emilia

*Prof. Nicola Bicocchi ([nicola.bicocchi@unimore.it](mailto:nicola.bicocchi@unimore.it))*



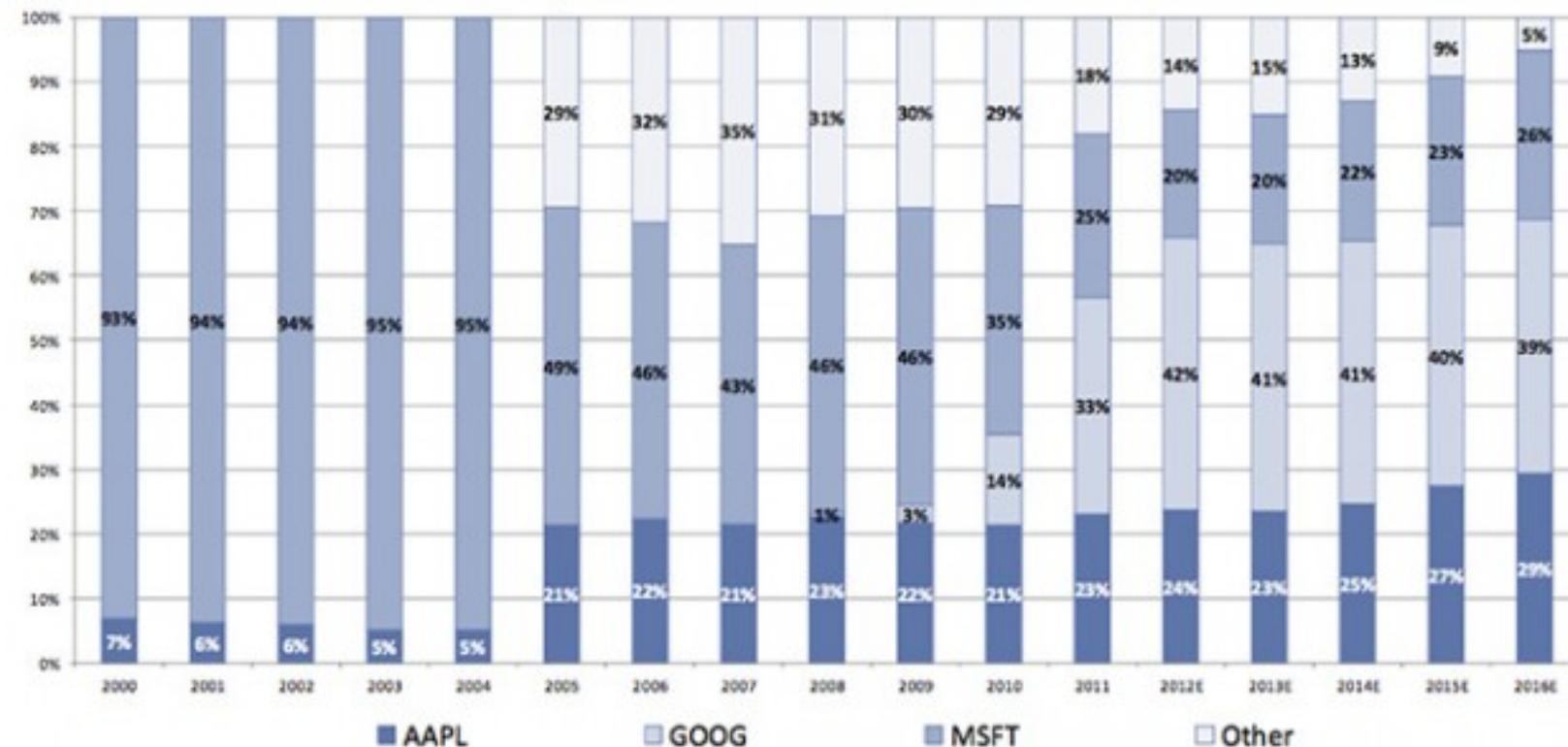
# Introduzione

---



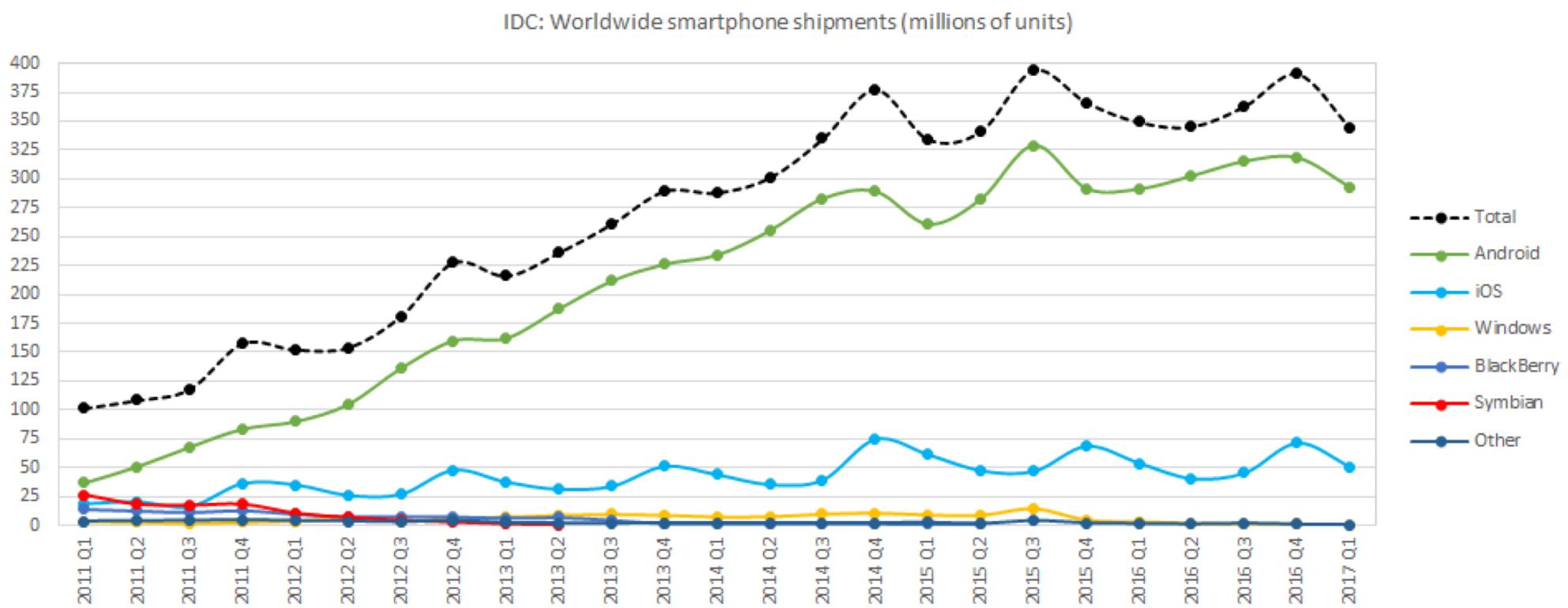
# Total Market Share

Vendor share of consumer compute, 2000-2016E

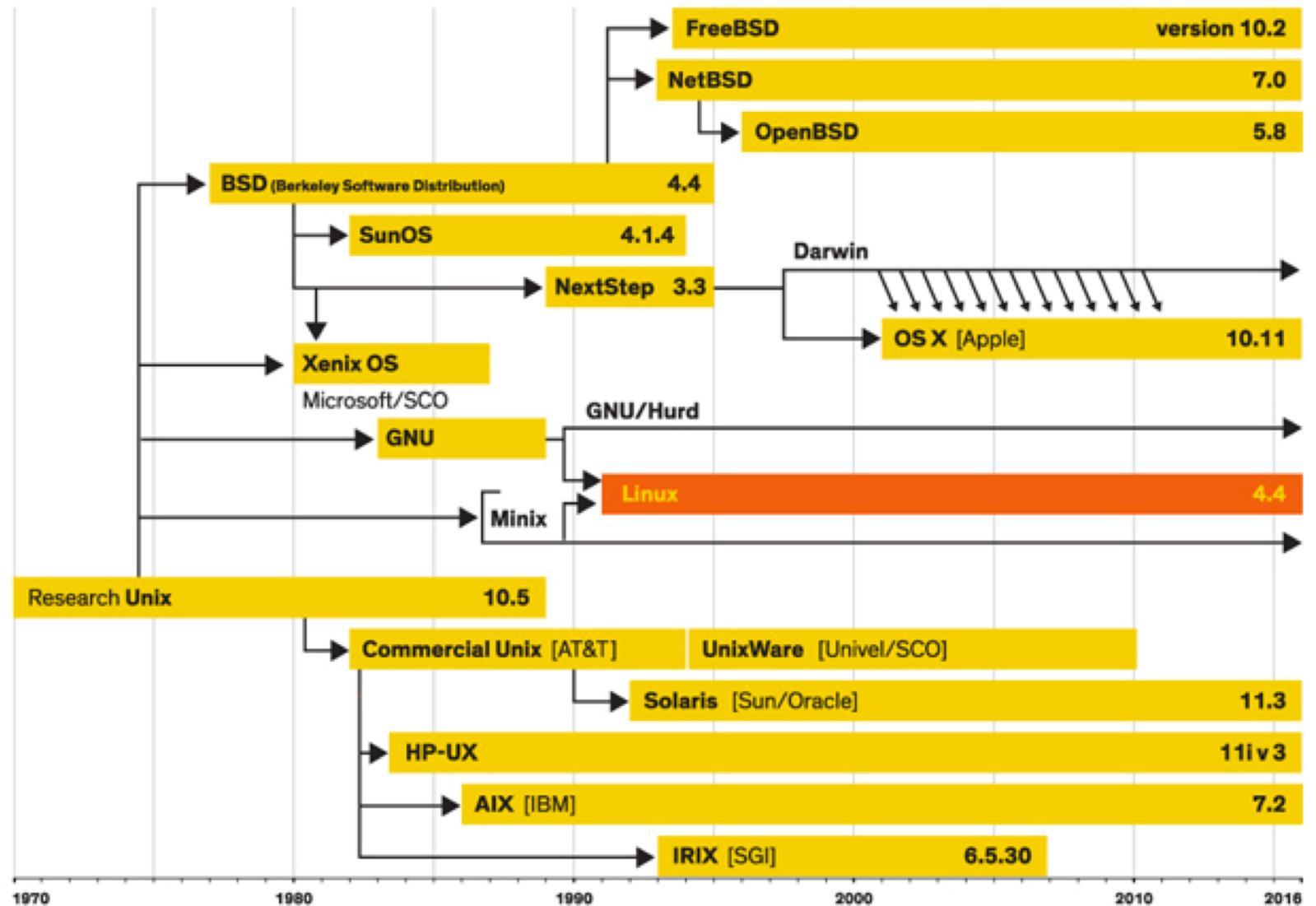


Source: IDC, Goldman Sachs Research.

# Mobile Market Share

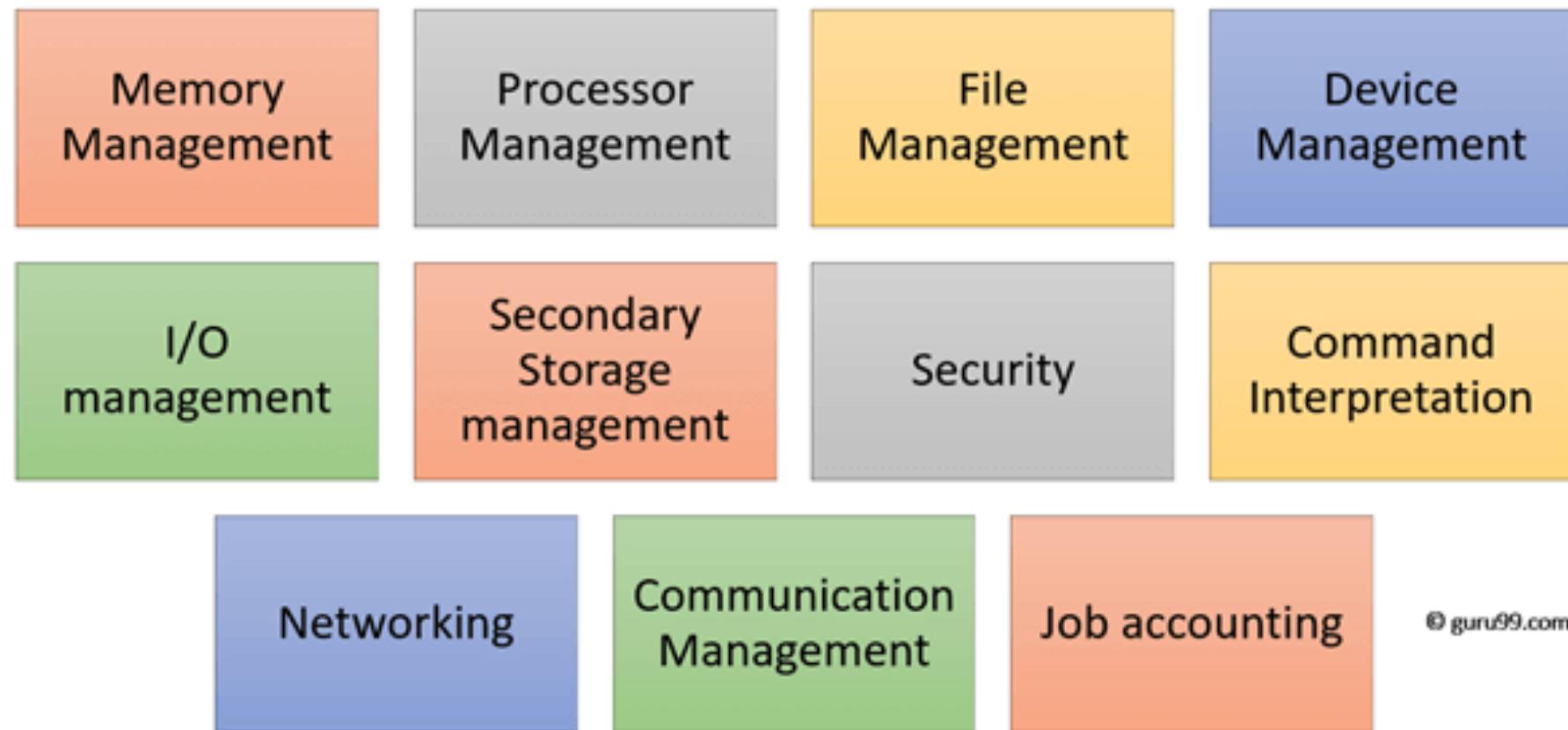


# Genealogia famiglia Unix



# Funzionalità principali OS

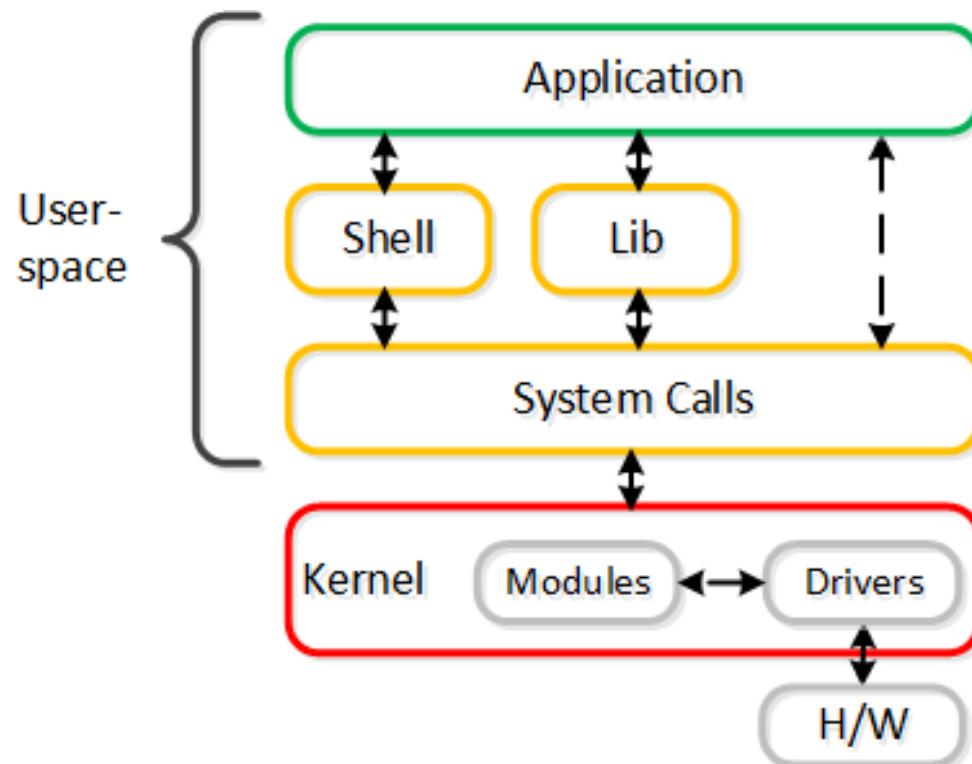
---



© guru99.com



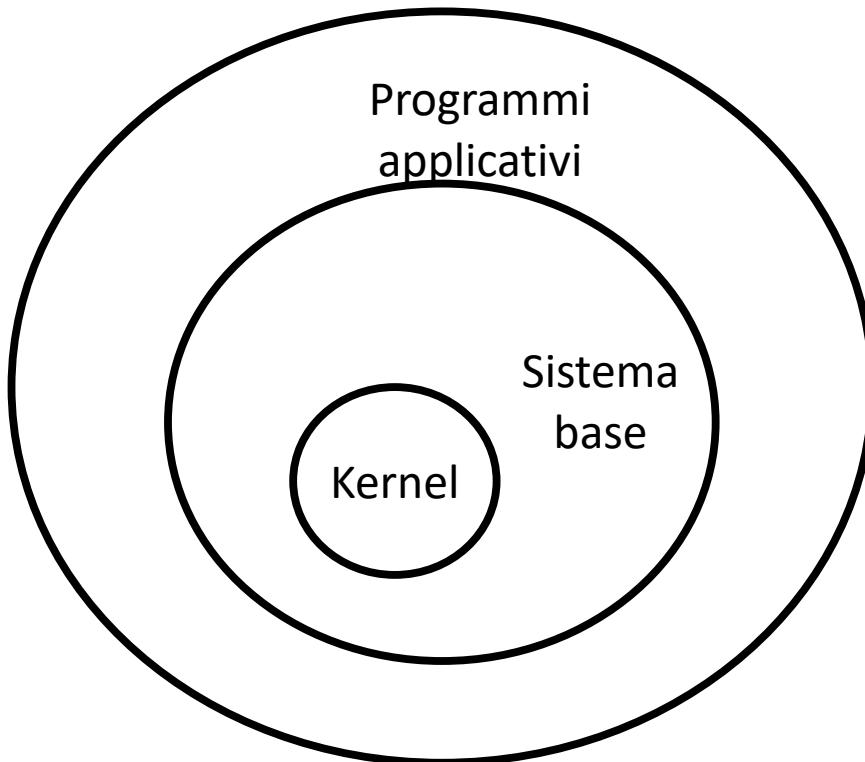
# Architettura interna



- Sistema Operativo
  - Kernel: mediatore fra applicazioni e hardware
  - Sistema base: gestisce la fase di boot ed un insieme di funzionalità minime
- Programmi applicativi

# Struttura a guscio

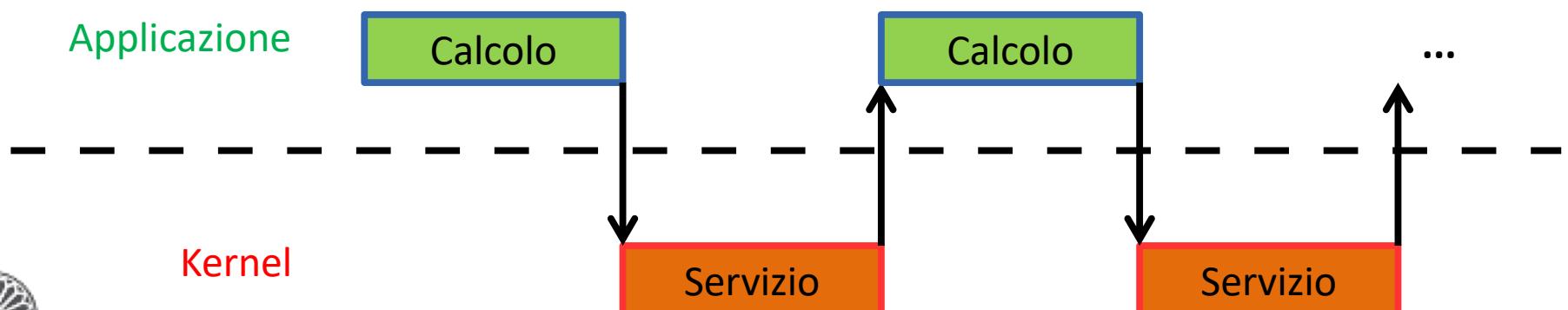
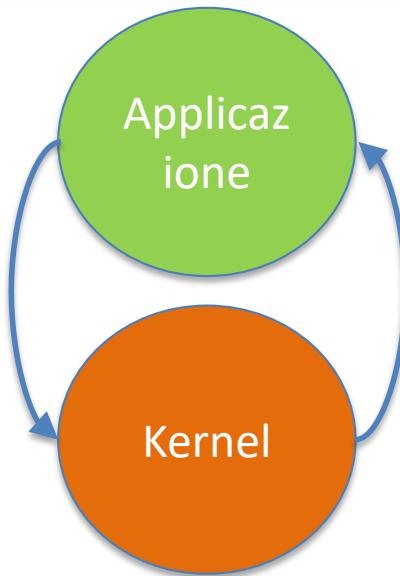
---



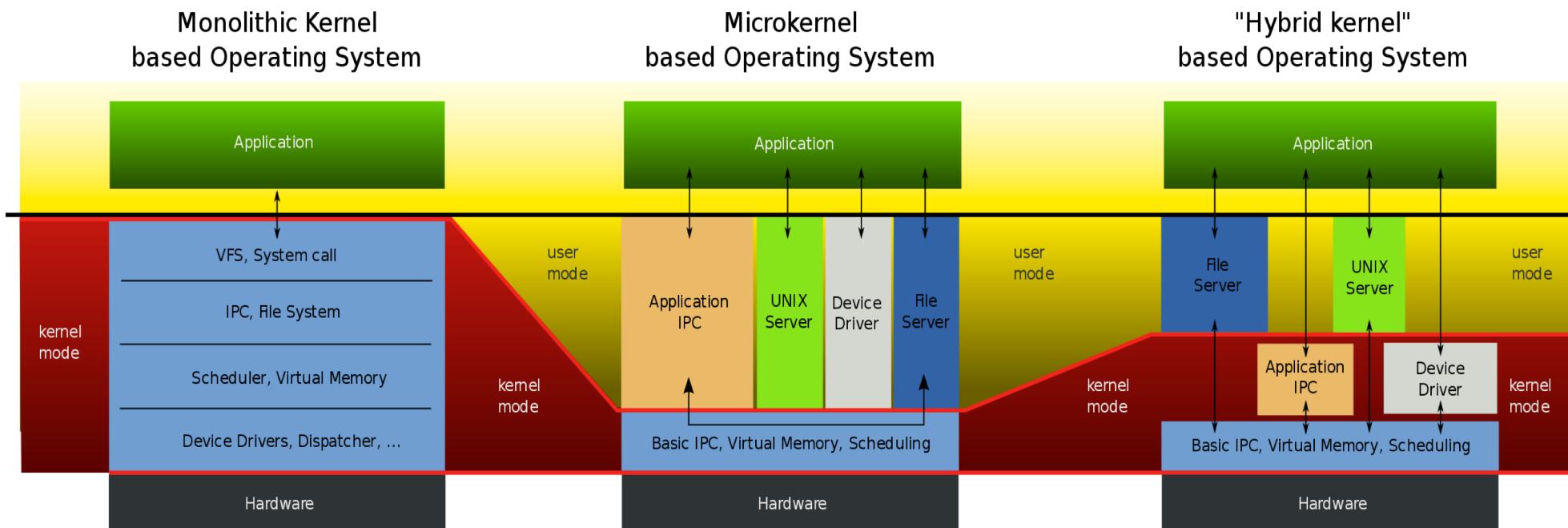
- Applicativi: browser, email, programme ufficio, compilatori
- Sistema base: librerie di sistema, sistema di boot, shell, terminale
- Kernel: gestione processi, filesystem, memoria, IPC

# Interazione Kernel - Applicazioni

- Modello Client-Server
  - Applicazioni richiedono servizi al kernel
  - Kernel elabora la risposta e risponde all'applicazione

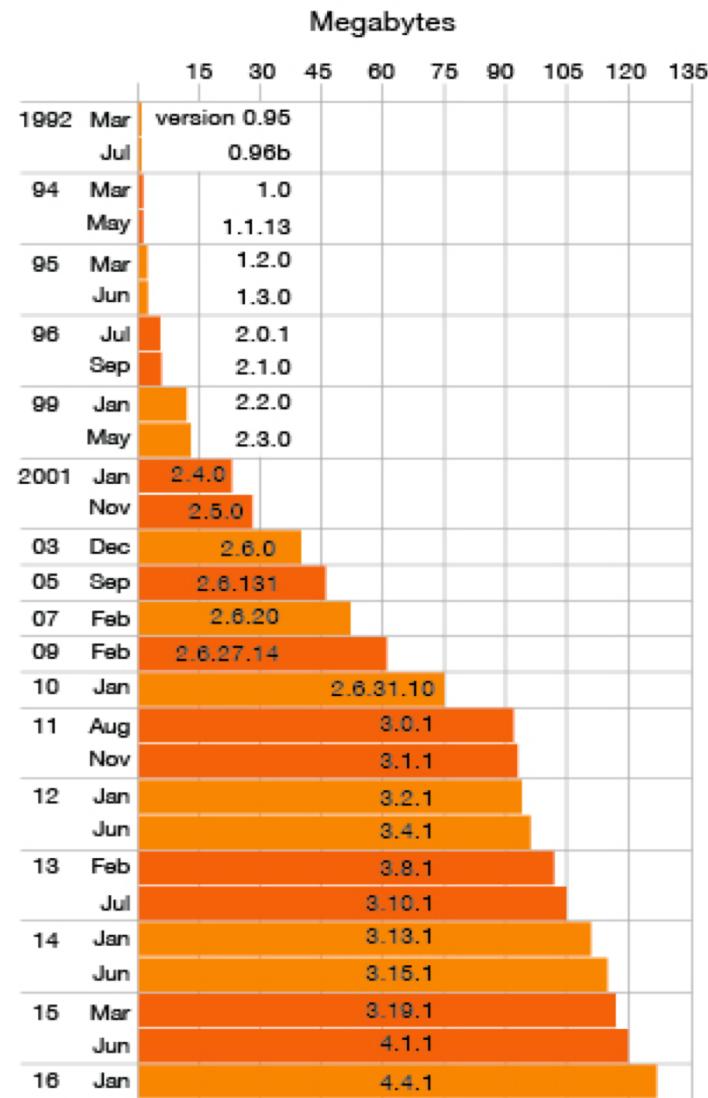


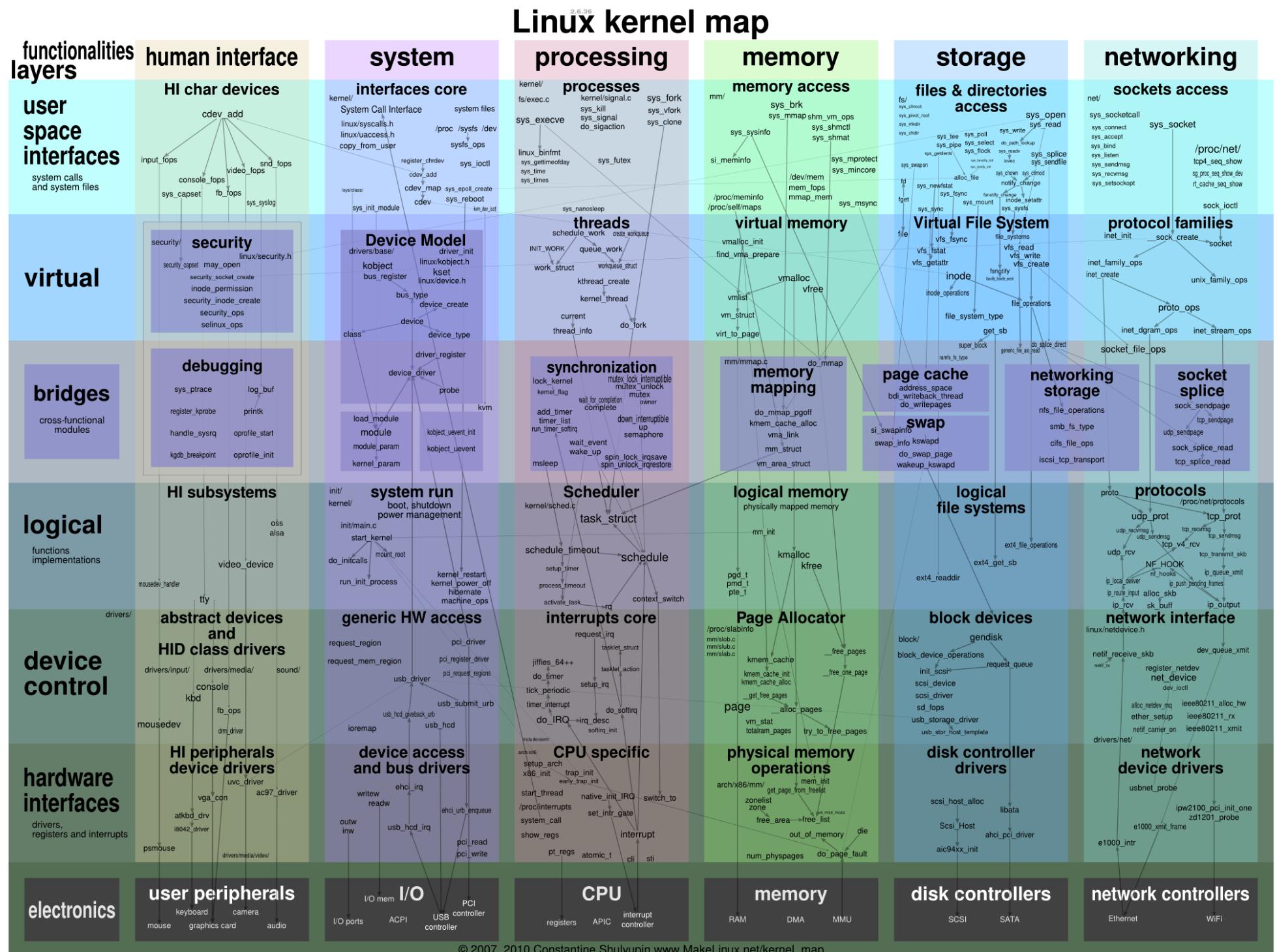
# Monolitici, Micro, Ibridi



# Quanto è complesso un kernel?

- 20K SLOC (XV6)
- <https://github.com/mit-pdos/xv6-public>
- 30M SLOC (Linux Kernel 5)
- <https://www.kernel.org/>





# Filosofia UNIX

---

# KISS principle

---

**Keep It Simple, Stupid.** In riferimento al codice sorgente di un programma, significa mantenere uno stile di progettazione semplice e lineare demandando le ottimizzazioni al compilatore o a successive fasi dello sviluppo.

Richiama in parte il principio filosofico del Rasoi di Occam: A parità di fattori la spiegazione più semplice è da preferire.

[https://en.wikipedia.org/wiki/Unix\\_philosphy](https://en.wikipedia.org/wiki/Unix_philosphy)

# Multiutenza e Multitasking

---



# Multitasking

---

- Un sistema operativo multitasking permette di eseguire più programmi (task) contemporaneamente. Ad esempio, se viene chiesto al sistema di eseguire due processi, A e B, la CPU eseguirà per qualche istante il processo A, poi per qualche istante il processo B, poi tornerà ad eseguire il processo A e così via creando l'illusione che procedano contemporaneamente.
  - Il componente del Kernel delegato a questa funzione viene chiamato *scheduler*

# Multiutenza

---

- Un sistema multiutente può essere utilizzato contemporaneamente da utenti diversi. Ad ogni utente del sistema viene assegnato uno username, una password, e una cartella personale
  - /Users/nomeutente (macOS)
  - /home/nomeutente (Linux)

# Console e terminali

---

# Terminale testuale

---

- Con il termine console o terminale si definisce una coppia tastiera/video collegata alla macchina. Storicamente, per rendere accessibile una macchina da più utenti, era possibile collegare più tastiere e video allo stesso computer. Oggi i terminali sono virtuali.

# Terminale testuale

---

```
chris@ubuntu: ~
chris@ubuntu:~$ bash --version
GNU bash, version 4.3.46(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.

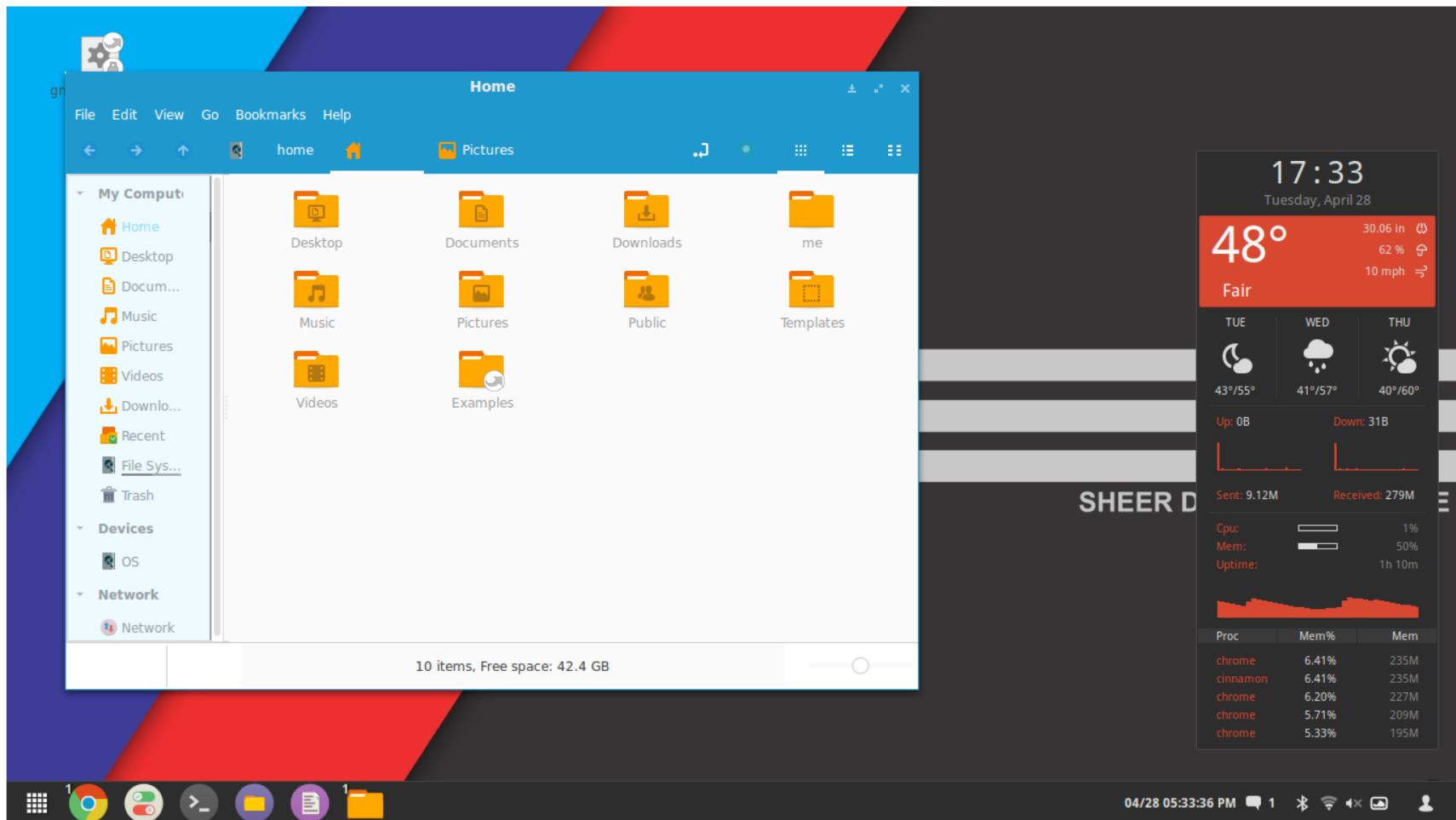
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
chris@ubuntu:~$
```

# Terminale grafico

---

- Esistono terminali più evoluti, i cosiddetti terminali grafici che permettono di utilizzare un'interfaccia grafica (GUI) per eseguire le operazioni di input/output.
- Windows dispone di un solo terminale grafico, i sistemi Unix ne hanno diversi intercambiabili (X Window System).
- <https://www.tecmint.com/best-linux-desktop-environments/>

# Terminale grafico



# Perchè usare il terminale testuale?

---

- Accesso completo alla configurazione del sistema e dei servizi
- Automatizzazione e scripting
- Basso consumo di risorse computazionali
- Esistono applicazioni in cui un terminale grafico non viene installato perchè inutile (non esiste monitor) o per risparmiare risorse (apparati rete/applicazioni IoT)

# Apertura e chiusura sessioni

---



# login

---

woodstock login: nicola

Password: \*\*\*\*\*

Last login: Fri Mar 06 10:27:08 on ttys2

\$\_ ← shell prompt

# shell

---

- Programma che permette di far interagire l'utente con il sistema operativo tramite comandi
  - resta in attesa di un comando...
  - esegue comando alla pressione di <ENTER>
- La shell è un interprete di comandi evoluto
  - potente linguaggio di scripting
  - interpreta ed esegue comandi da standard input o da file comandi

# Ciclo di esecuzione shell

---

```
loop forever
    <LOGIN>
    do
        <ricevi comando da file di input>
        <interpreta comando>
        <esegui comando>
    while (! <EOF>)
    <LOGOUT>
end loop
```

# Quale shell?

---

- La shell non è unica, un sistema può metterne a disposizione varie
  - Bourne shell (/bin/bash)
  - C shell (/bin/csh)
  - Fish shell (/bin/fish)
- Ogni utente può indicare la shell preferita. La scelta viene memorizzata in /etc/passwd, un file contenente le informazioni di tutti gli utenti del sistema
- Dopo il login, per ogni utente viene generato un processo shell dedicato

# passwd

---

Per modificare la password dell'utente in esecuzione è possibile utilizzare il comando **passwd [OPTION] [USER]**

```
$ passwd  
(current) UNIX password:  
Enter new UNIX password:  
Retype new UNIX password:
```

# logout

---

- Per uscire da una shell si può utilizzare il comando **exit** (che invoca la system call `exit()` per quel processo). In alternativa:
  - **logout**
  - **CTRL+D**
- Per rientrare, va effettuato un nuovo login

# shutdown

---

- Varie possibilità
  - \$ sudo shutdown -h now
  - \$ sudo halt
- Trattandosi di modifica al sistema (lo spegnimento coinvolge tutti gli utenti) sono necessari diritti di amministrazione

# Manuale in linea

---



# man

---

- Esiste un manuale on-line (**man**), consultabile per informazioni su ogni comando Linux. Indica:
  - formato del comando (input) e risultato atteso (output)
  - descrizione delle opzioni
  - possibili restrizioni
  - file di sistema interessati dal comando
  - comandi correlati
  - eventuali bug per uscire dal manuale
- Per uscire premere q

# apropos

---

- Per cercare una pagina di manuale di cui non si conosce il nome, è possibile usare il comando **apropos** per cercare tutte le pagine che contengono una keyword specifica

```
$ apropos man
```

```
$ apropos top
```

# Utenti e gruppi

---

# Concetto di gruppo

---

- Sistema multiutente ⇒ problemi di privacy e di possibili interferenze: necessità di proteggere/nascondere informazione
- Concetto di gruppo (staff, utenti, studenti): possibilità di lavorare sugli stessi documenti
- Ogni utente appartiene a un gruppo principale ma può far parte anche di gruppi secondari a seconda delle esigenze e configurazioni

# Utenti

---

- Ogni utente è identificato univocamente all'interno del sistema mediante uno **username**. Gli utenti del sistema sono distribuiti in più gruppi; ogni utente fa parte di almeno un gruppo.
- Esiste un utente privilegiato, il cui username è **root**, che viene assegnato all'amministratore del sistema. **root** può modificare la configurazione dell'intero sistema.

# sudo

---

\$ **sudo apt-get update**

\$ ← Prompt utente normale

Eleva i diritti di esecuzione (da utente a root) per un solo comando. *Per aggiornare il sistema sono necessari diritti di amministrazione*

\$ **sudo -i**

# ← Prompt utente root

Eleva i diritti di esecuzione in modo permanente

# /etc/passwd

---

**Username:** username dell'utente

**Password:** la x indica che la password cifrata è presente nel file /etc/shadow

**User ID (UID):** ID utente

**Group ID (GID):** ID del gruppo (primario) dell'utente

**User ID Info:** Informazioni aggiuntive

**Home directory:** percorso assoluto home directory utente

**Command/shell:** percorso assoluto shell utente

*nicola : x : 1000 : 1000 : Nicola B.,,,: /home/nicola : /bin/bash*

# /etc/group

---

**Group name:** nome del gruppo

**Password:** generalmente non utilizzato. Si possono definire password di gruppo.

**Group ID (GID):** ID del gruppo

**Group List:** lista degli utenti che appartengono al gruppo

*adm : x : 4 : syslog,nicola*

*sudo : x : 27 : nicola*

*nicola : x : 1000 :*

# whoami, id

---

who-am-I mostra il nome utente corrente

```
$ whoami
```

```
nicola
```

id mostra UID, GID, gruppi secondari

```
$ id
```

```
uid=1000(nicola) gid=1000(nicola)
groups=1000(nicola),4(adm),24(cdrom),2
7(sudo),30(dip),46(plugdev),116(lpadmin),
126(sambashare)
```

# Protezione dei file

---

- Multiutenza implica necessità di regolare gli accessi alle informazioni. Per ogni file, esistono 3 tipi di utilizzatori:
  - proprietario, user
  - gruppo del proprietario, group
  - tutti gli altri utenti, others
- Per ogni utilizzatore, si distinguono tre modi di accesso al file:
  - lettura (r)
  - scrittura (w)
  - esecuzione (x) (per una directory significa list del contenuto)
- Ogni file è marcato con UID e GID del proprietario
- 12 bit di protezione

# File e metadati

```
host133-63:~ marco$ ls -l
```

tot. spazio occupato (blocchi)

```
total 8
```

permessi	numero di (hard) link	proprietario	gruppo	dimensione (byte)	data ultima modifica	nome file
drwx-----	3	paolo	prof	102	May 18 22:49	Desktop
drwx-----	3	paolo	prof	102	May 18 22:49	Documents
-rw-r--r--	1	pippo	stud	29	May 19 00:10	f1.txt
-rw-r--r--	1	marco	nerdz	0	May 18 22:53	f2



# Bit di protezione

---

12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	1	1	1	1	0	0	1	0	0
SUID	SGID	Sticky	R	W	X	R	W	X	R	W	X
User				Group				Others			
PERMESSI											

# chmod, chown

---

chmod [opzioni] mode file

Assegna diritti ad un file

\$ chmod 0755 /home/nicola/test

0	0	0	1	1	1	1	0	1	1	0	1
SUID	SGID	Sticky	R	W	X	R	W	X	R	W	X
User				Group				Others			

chown [opzioni] owner:group file

Assegna proprietario e gruppo ad un file

\$ chown nicola:nicola /home/nicola/test

# SUID, SGID, Sticky

---

- SUID (Set User ID)
  - Si applica a un file di programma eseguibile solamente
  - Se attivo, l'utente che esegue il programma viene considerato il proprietario di quel file (solo per la durata della esecuzione)
  - È necessario per consentire lettura/scrittura su file di sistema, che l'utente non avrebbe il diritto di leggere/ modificare.
  - Esempio: passwd (vedi diritti /etc/passwd)
- SGID: come SUID bit, per il gruppo
- Sticky: il sistema cerca di mantenere in memoria l'immagine del programma, anche se non è in esecuzione

# adduser, deluser

---

\$ **sudo adduser** utente

Aggiunge un nuovo utente al sistema

\$ **sudo deluser** utente

Rimuove un utente dal sistema

In alternativa, è sempre possibile modificare manualmente i file `/etc/passwd` e `/etc/group` e usare il comando `passwd` per aggiornare la password



# Processi

---



# Utenti e Processi

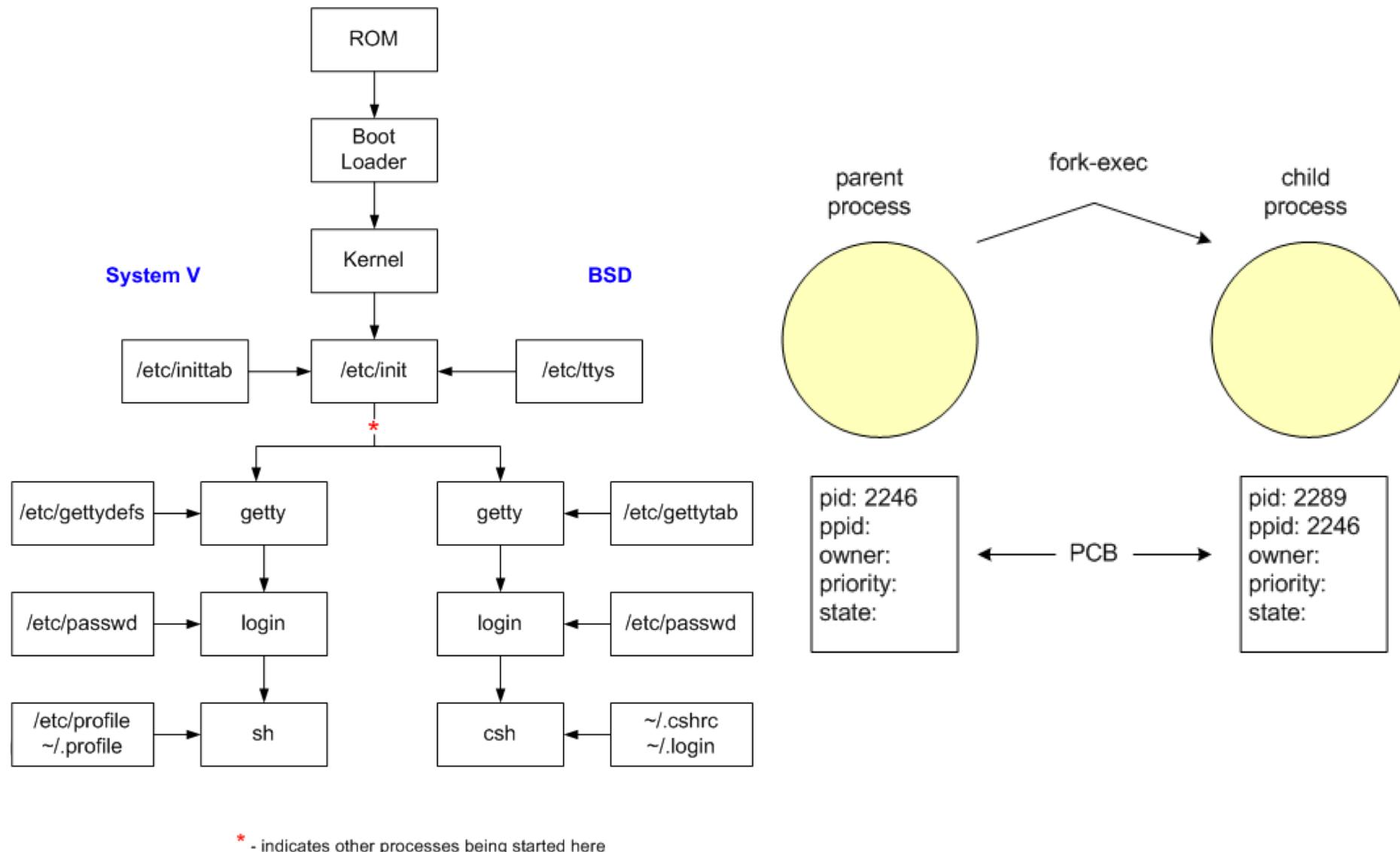
---

- Ogni operazione eseguita su una macchina Unix viene effettuata a nome e per conto di un determinato utente. Non esistono task o programmi funzionanti in modalità anonima!
- Ogni programma viene eseguito per conto di un determinato utente e pertanto ne acquisisce tutti i permessi ed i vincoli.

\$ ps aux



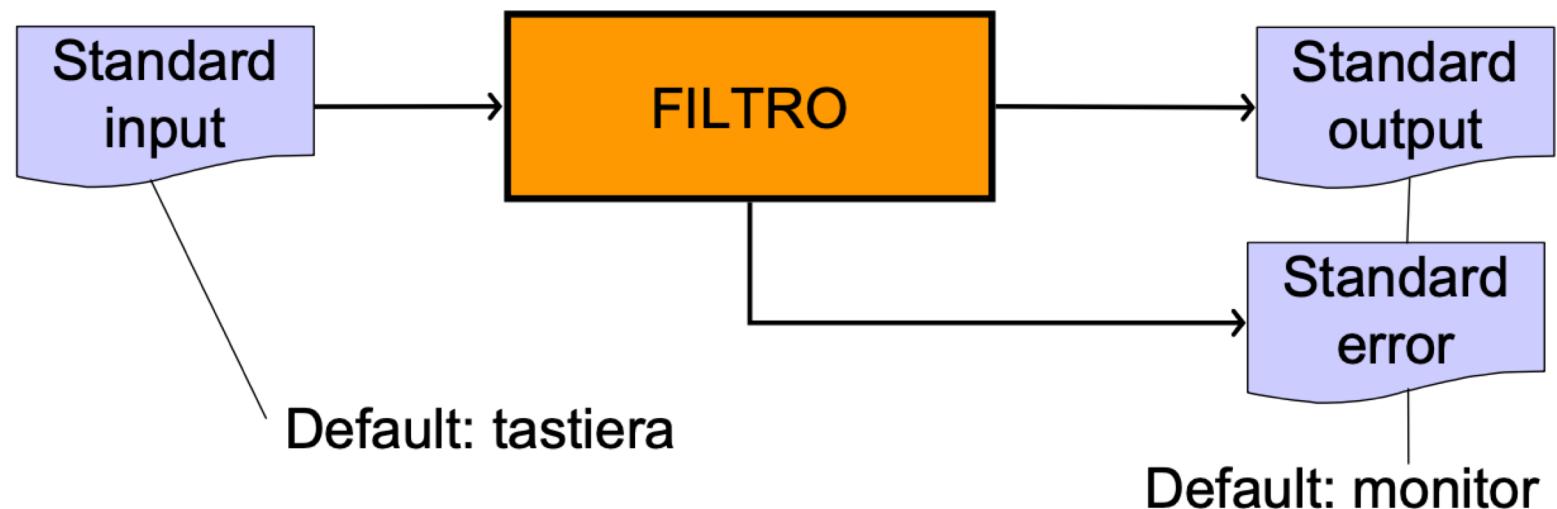
# Avvio del sistema



# Flussi dati standard

---

- I comandi UNIX si comportano come FILTRI
- Un filtro è un programma che riceve un ingresso da un input e produce il risultato su uno o più output



# Esecuzione comando (processo)

---

- \$ ls
- I comandi principali del sistema si trovano nelle directory /bin oppure /usr/bin
- Possibilità di realizzare nuovi comandi (scripting). Per ogni comando, la shell genera un processo figlio dedicato alla sua esecuzione
- Il processo padre attende la terminazione del comando (foreground) o prosegue in parallelo (background)

# Formato invocazione

---

- Di solito: nome comando opzioni argomenti
- \$ ls –l filename
- Convenzione nella rappresentazione della sintassi comandi:
  - se un'opzione o un argomento possono essere omessi, si indicano tra quadre [opzione]
  - se due opzioni/argomenti sono mutuamente esclusivi, vengono separati da |. Ad esempio: arg1 | arg2
  - quando un argomento può essere ripetuto n volte, si aggiungono dei puntini argomento...

# ps

---

Un processo utente in genere viene attivato a partire da un comando (da cui prende il nome). Tramite ps si può vedere (staticamente) la lista dei processi attivi. Per una rappresentazione continua si utilizza top.

```
nicola@ubuntu:~$ ps
```

PID	TTY	TIME	CMD
5527	pts/0	00:00:00	bash
7595	pts/0	00:00:00	ps

# top – linea #1

---

- Ora attuale (21:34:21)
- Uptime della macchina (3:51)
- Utenti attualmente connessi (2 users)
- Media del carico di sistema. i 3 valori si riferiscono a: ultimo minuto, ultimi 5 minuti, ultimi 15 minuti.

```
top - 21:34:21 up  3:51,  2 users,  load average: 1.01, 0.41, 0.25
Tasks: 134 total,   1 running, 133 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  0.3%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem: 508488k total, 466596k used, 41892k free, 59132k buffers
Swap: 407548k total,    2516k used, 405032k free, 234228k cached
```

# top – linea #2

---

- Processi totali in esecuzione (134 total)
- Processi attivi (1 running)
- Processi dormienti (133 sleeping)
- Processi in stop (0 stopped)
- Processi che aspettano di essere gestiti dal processo padre (0 zombie)

```
top - 21:34:21 up  3:51,  2 users,  load average: 1.01, 0.41, 0.25
Tasks: 134 total,   1 running, 133 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  0.3%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem: 508488k total, 466596k used, 41892k free, 59132k buffers
Swap: 407548k total,    2516k used, 405032k free, 234228k cached
```

# top – linea #3

---

- Percentuale del carico dei processi utente (0.7%**us**)
- Percentuale del carico dei processi di sistema (0.3%**sy**)
- Percentuale del carico dei processi con priorità di aggiornamento *nice* (0.0%**ni**)
- Percentuale di inattività della cpu (99.0%**id**)
- Percentuale dei processi in attesa di operazioni I/O (0.0%**wa**)

```
top - 21:34:21 up 3:51, 2 users, load average: 1.01, 0.41, 0.25
Tasks: 134 total, 1 running, 133 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 508488k total, 466596k used, 41892k free, 59132k buffers
Swap: 407548k total, 2516k used, 405032k free, 234228k cached
```

# Filesystem

---

# Tutto è file

---

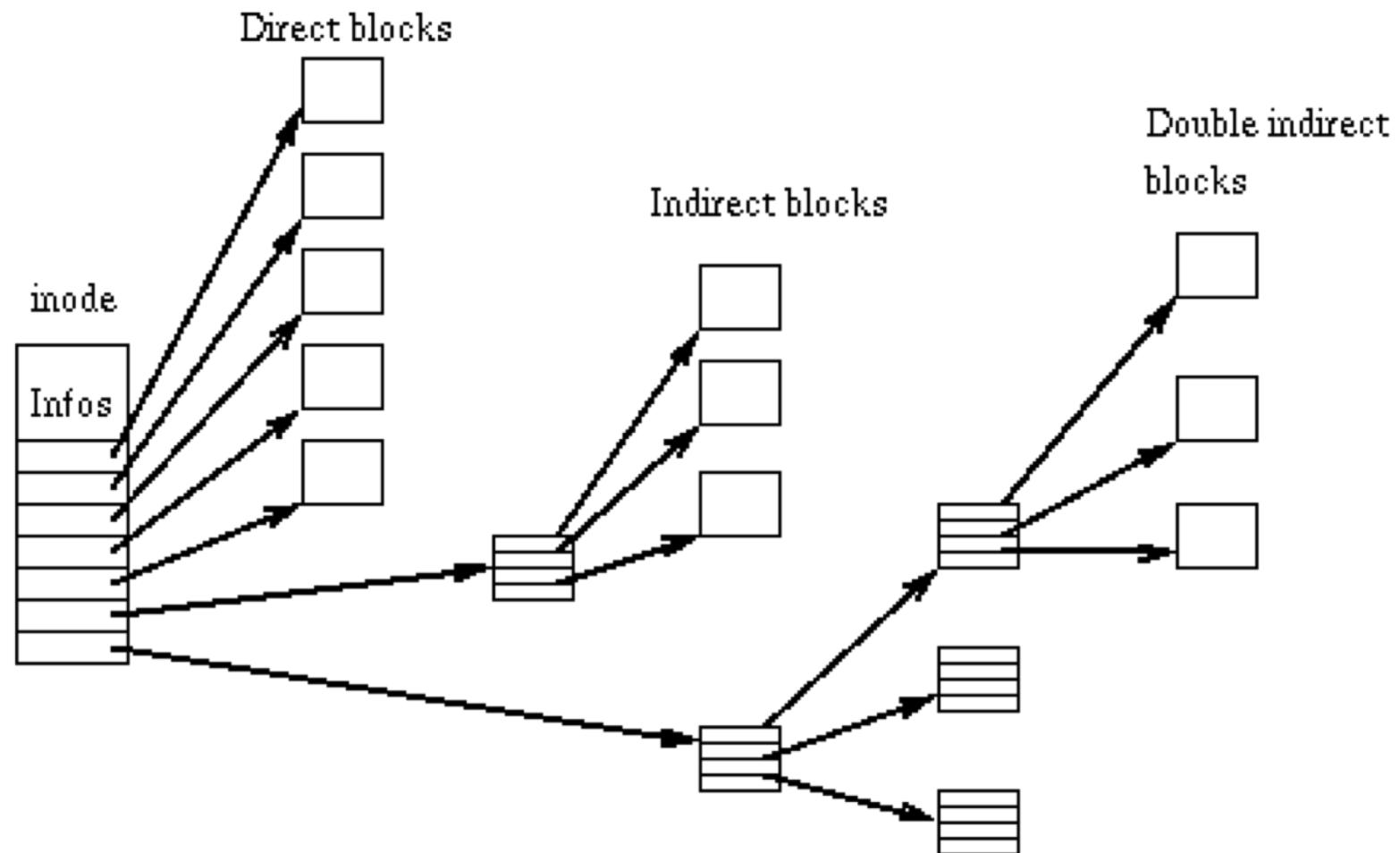
- File come risorsa logica costituita da sequenza di bit, a cui viene dato un nome
- Astrazione che consente di trattare allo stesso modo entità fisicamente diverse come file di testo, dischi rigidi, stampanti, direttori, tastiera, video, ...
  - Ordinari: archivi di dati, comandi, programmi sorgente
  - Directory: contengono riferimenti a file
  - Speciali: dispositivi hardware, FIFO, soft links

# Tutto è file

---

- È possibile nominare un file con una qualsiasi sequenza di caratteri (max 255), a eccezione di '.' e '..'
- È sconsigliabile utilizzare per il nome di file dei caratteri speciali, ad es. metacaratteri e segni di punteggiatura
- Ad ogni file possono essere associati uno o più nomi simbolici (link) ma ad ogni file è associato un solo **i-node**

# i-node



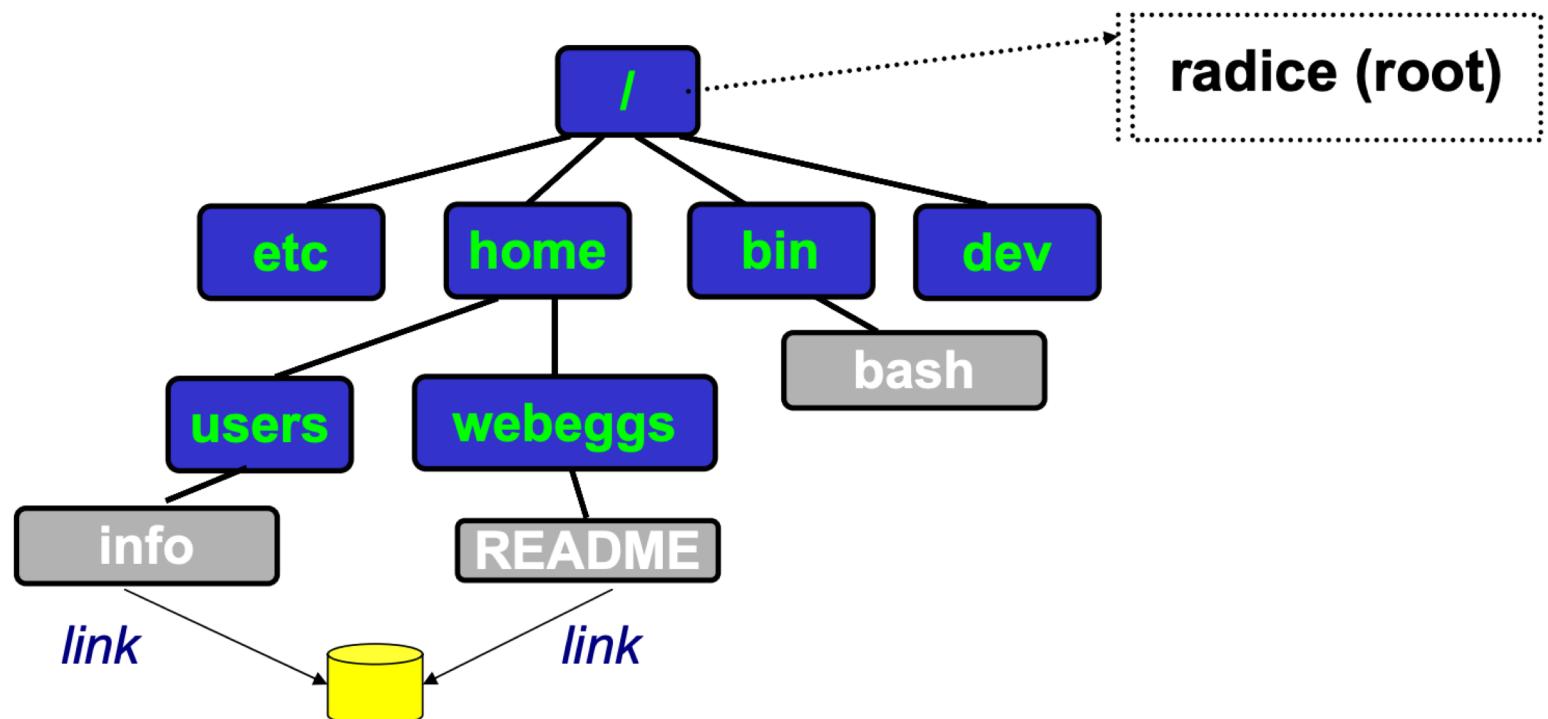
# File di testo, file binario

---

- **File di testo:** leggibile da un essere umano. I dati contenuti rappresentano caratteri (ASCII o Unicode)
- **File binario:** richiede specifica interpretazione di un software per essere letto (mp3, jpg, mp4)
- **Tipo di file:** lo specifico tipo di informazione contenuta nel file (audio, immagini, testo)
- **Estensione:** i caratteri terminali del nome di un file (di solito 3) che su alcuni sistemi, ad esempio Windows ne rappresentano il tipo

# Directory

- File system Linux è organizzato come un grafo diretto aciclico (DAG)



# Gerarchie

---

- All'atto del login, l'utente comincia ad operare all'interno di una specifica directory (*/home/nameofuser*). In seguito è possibile cambiare directory.
- Il sistema operativo mette a disposizione comandi per orientarsi (cd, pwd)

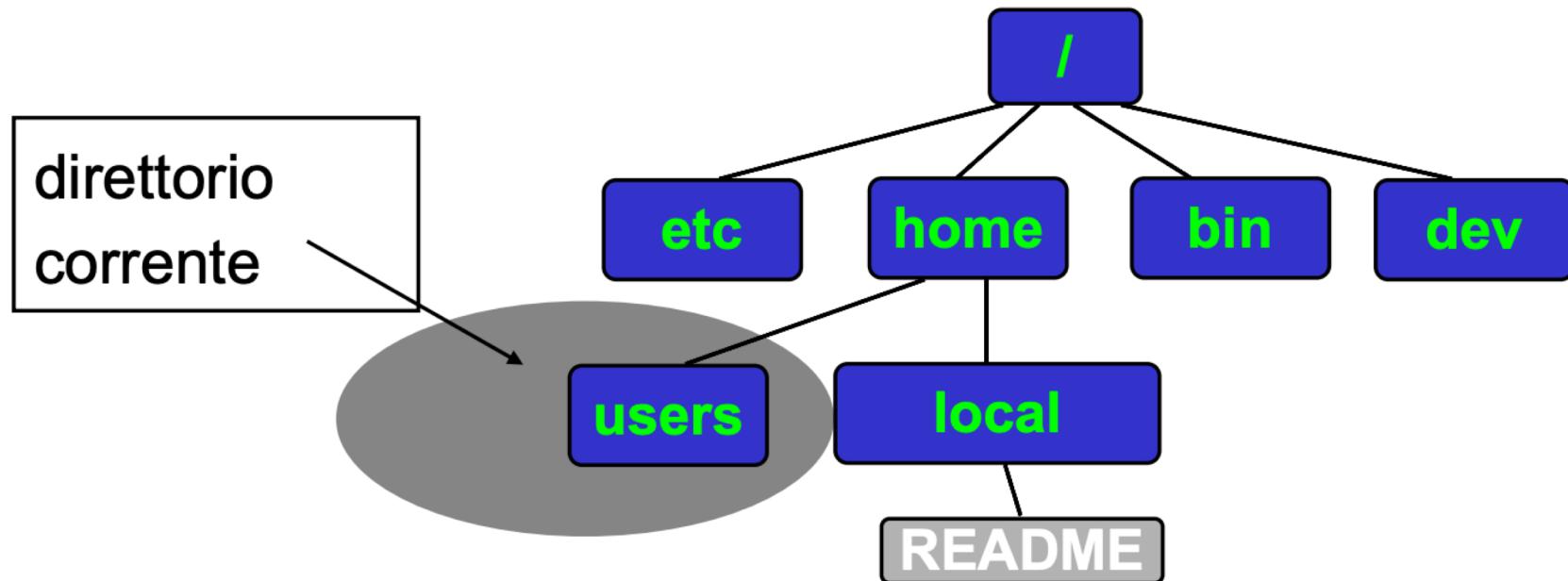
# Nomi assoluti e relativi

---

- 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. Inizia SEMPRE con /
- Nomi particolari:
  - . directory corrente (visualizzato da pwd)
  - .. directory ‘padre’
  - ~ home utente

# Nomi assoluti e relativi

---



nome assoluto: /home/local/README

nome relativo: ../local/README

# Links

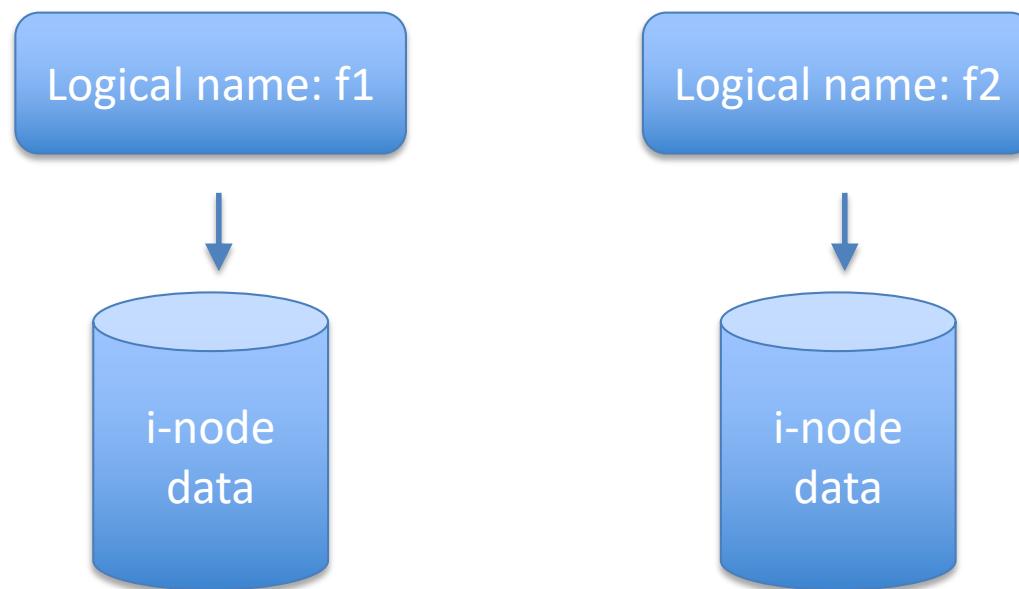
---

- Le informazioni contenute in un file possono essere visibili attraverso nomi diversi, tramite “riferimenti” (link) allo stesso file fisico
- SO considera e gestisce la molteplicità possibile di riferimenti: se un file viene cancellato, le informazioni sono veramente eliminate solo se non ci sono altri link a esso
- Due tipi di link:
  - link fisici (`$ ln src dst`)
  - link simbolici (`$ ln -s src dst`)

# cp vs ln vs ln -s

---

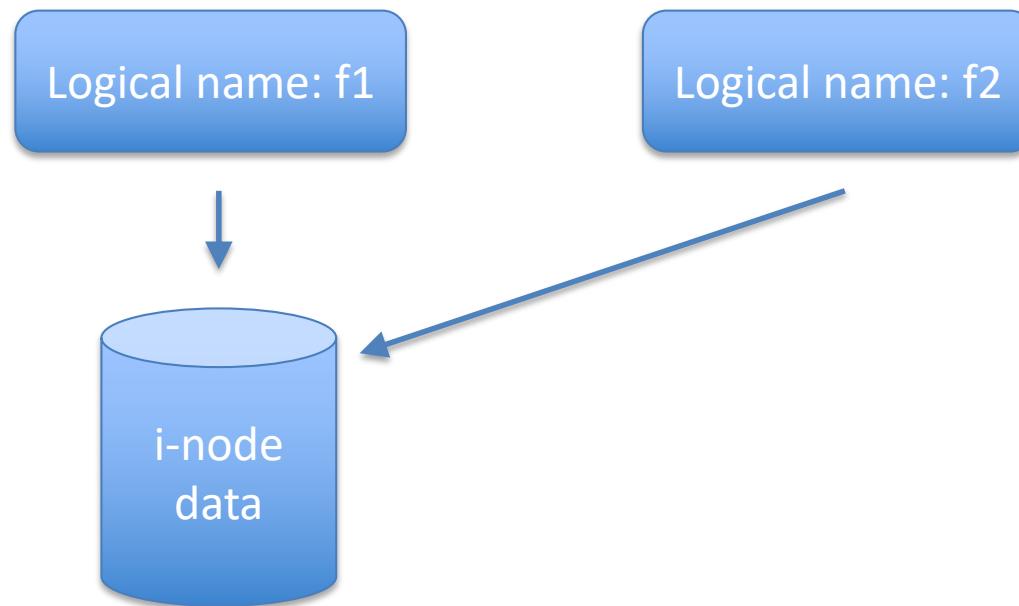
```
$ cp /home/nicola/f1 /home/nicola/f2
```



# cp vs ln vs ln -s

---

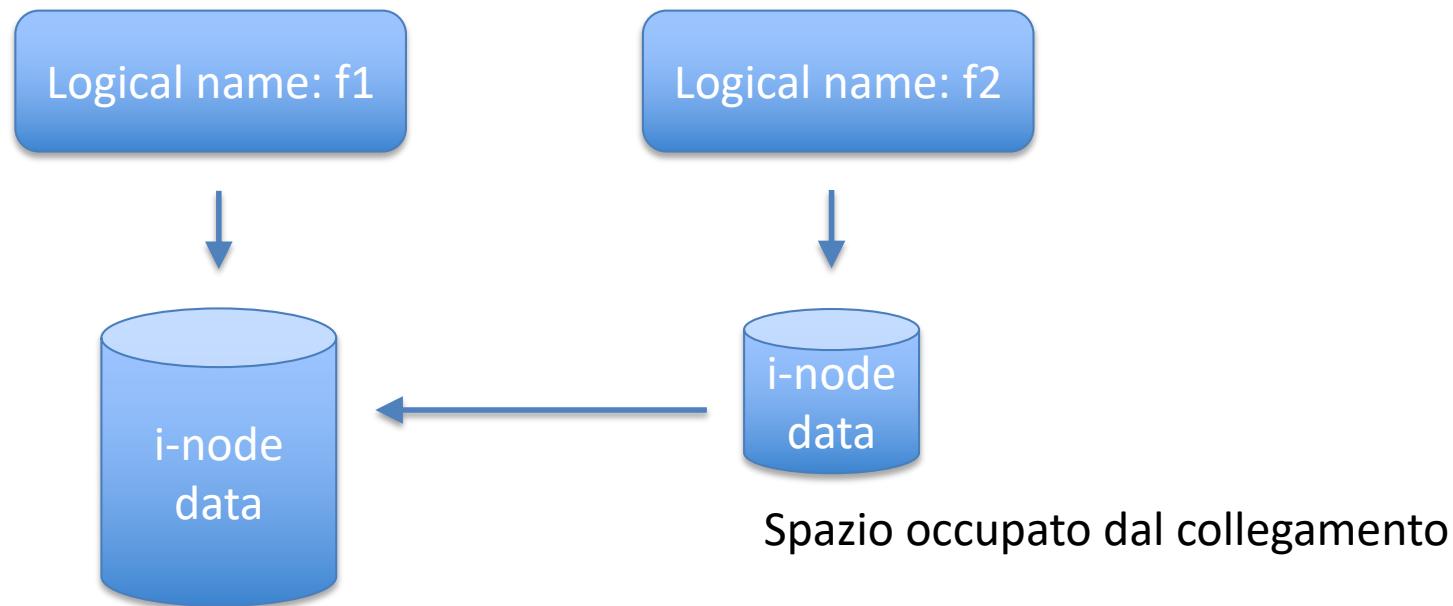
```
$ ln /home/nicola/f1 /home/nicola/f2
```



# cp vs ln vs ln -s

---

```
$ ln -s /home/nicola/f1 /home/nicola/f2
```



# Struttura file system

```
\ ---- bin
      |
      +--- dev
      |
      +--- etc
      |
      +--- home ---- marco
          |
          |           +--- marina
          |
          |           +--- root
      +--- lib
      |
      +--- proc
      |
      +--- sbin
      |
      +--- tmp
      |
      +--- usr ---- X11
          |
          |           +--- bin
          |
          |           +--- include
          |
          |           +--- lib
          |
          |           +--- local ---- bin
          |                   |
          |                   +--- etc
          |                   |
          |                   +--- lib
          |
          |           +--- man
          |
          +--- var ---- log
              |
              +--- mail
              |
              +--- spool
```

- Ogni sottocartella di / rappresenta un gruppo di file con uno scopo preciso
- Varia fra i sistemi. In generale:
  - **/bin** binari essenziali (sistema di base)
  - **/etc** file di configurazione
  - **/home** home degli utenti
  - **/proc** interfaccia verso il kernel
  - **/tmp** file temporanei
  - **/usr** binari non essenziali (applicazioni)
  - **/var** log di sistema

# Composizione filesystem

---

# mount

---

- Un file system (contenuto su qualsiasi dispositivo) per essere utilizzato deve essere montato su un file system esistente, usando una directory come punto di attacco.
  - Ad esempio, per le chiavette USB
- La directory di aggancio prende il nome di mount point.

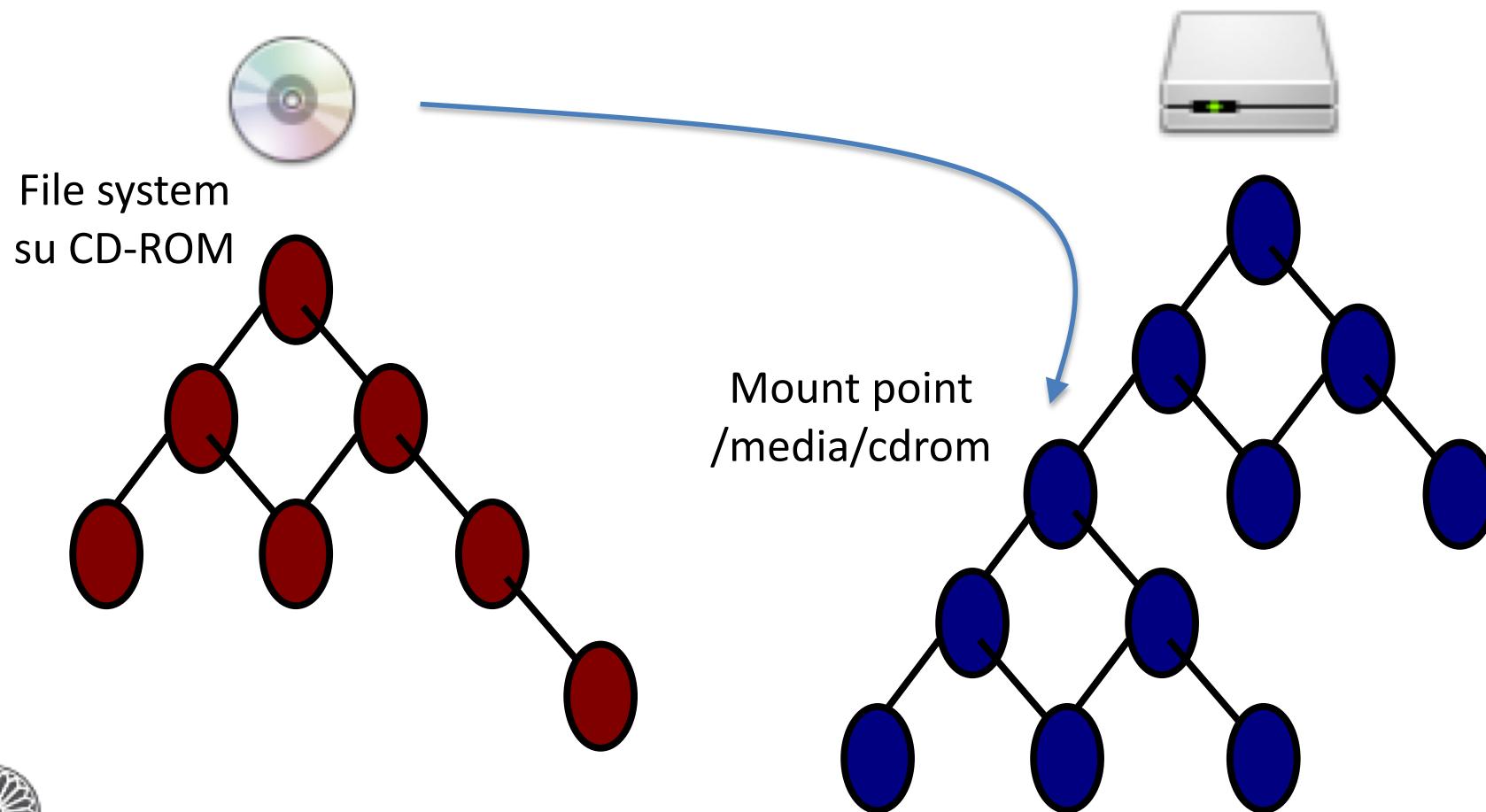
# umount

---

- Il file system può essere staccato dal suo mount point tramite l'operazione di umount (inversa di mount).
- Per motivi di efficienza, le scritture su di un file system sono eseguite in blocco, al momento più favorevole.
- Estrarre fisicamente un dispositivo senza aver smontato il suo file system può portare corruzione dei dati!

# Esempio mount

---



# Installazione pacchetti

---



# apt-get

---

- **apt-get** è il comando per la gestione (**install/remove/update**) pacchetti in distribuzioni derivate da Debian
- **apt-cache (search)** cerca pacchetti in base a parole chiave. Inoltre visualizza le intestazioni delle versioni disponibili del pacchetto (**show**).
- **/etc/apt/sources.list** contiene la lista dei repository attivi
- L'installazione o la rimozione di software di sistema richiede diritti di amministrazione (**sudo**)

# apt-get

---

- **apt-get update** aggiorna lista dei pacchetti disponibili
- **apt-get clean** rimuove tutti i pacchetti scaricati
- **apt-get install *pkgnname*** installa uno specifico pacchetto e le sue dipendenze
- **apt-get remove *pkgnname*** rimuove uno specifico pacchetto
- **apt-get autoremove** rimuove pacchetti inutili (dipendenze di pacchetti rimossi in precedenza)

# apt-get

---

```
$ sudo apt-get update  
$ sudo apt-cache search mc  
$ sudo apt-get install mc  
$ mc  
  
...  
$ sudo apt-get remove mc  
$ sudo apt-get autoremove  
$ sudo apt-get clean
```