# Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2022/2023

Corso di Laurea in Matematica

# Comandi Linux - Parte 1

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

# Argomenti trattati

1. Filesystem e file
   - Il filesystem
   - File in Linux e comandi sui file

## Filesystem e file

# Il filesystem - ricapitolazione

# Filesystem in Linux

- Un **filesystem** è un'organizzazione di un'area di memoria (tipicamente di massa, come il disco), basata sul concetto di *file* e di *directory*
  - una directory serve a contenere al suo interno altre directory oppure file
  - induce naturalmente una struttura gerarchica, ad albero, dove ogni nodo è una directory o un file
  - solo le directory possono avere figli

# Filesystem in Linux

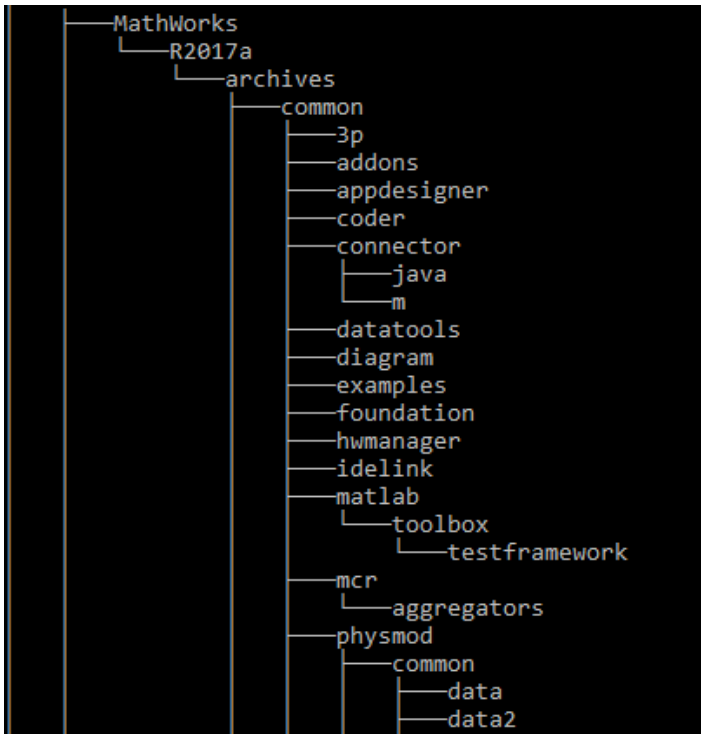
- Linux ha un solo filesystem principale, che ha come radice la directory /, cioè la directory **root**
  - tutti i file e le directory sono contenuti, direttamente o indirettamente, in tale directory
  - le **foglie** dell'albero possono essere **directory vuote** oppure **file**
  - all'interno della stessa directory non ci possono essere due file, due directory oppure un file e una directory con lo stesso nome
  - cambiare le maiuscole/minuscole è sufficiente a distinguere tra due files o directory: `nomeFile` è diverso da `nomefile`

# Comando `ls`

- Il comando `ls [-la] [-R] [nomedir]` mostra il contenuto della directory `nomedir` o della **cwd**, se `nomedir` non è dato)
  - `ls -l` mostra informazioni relative al contenuto della directory
  - `ls -a` mostra i file considerati *nascosti* (*hidden*), cioè i file con nomi che cominciano con il punto
  - `ls -R` mostra tutto il sottoalbero con radice in `nomedir`
  - `ls -s` mostra la dimensione dei file

# Comando `tree`

- Il comando `tree [-a] [-L maxdepth] [-d] [-x] [nomedir]` permette di visualizzare l'albero di directory
  - usare l'opzione `-L` per limitare la profondità dell'albero mostrato
  - potrebbe non essere installato: `sudo apt-get install tree`

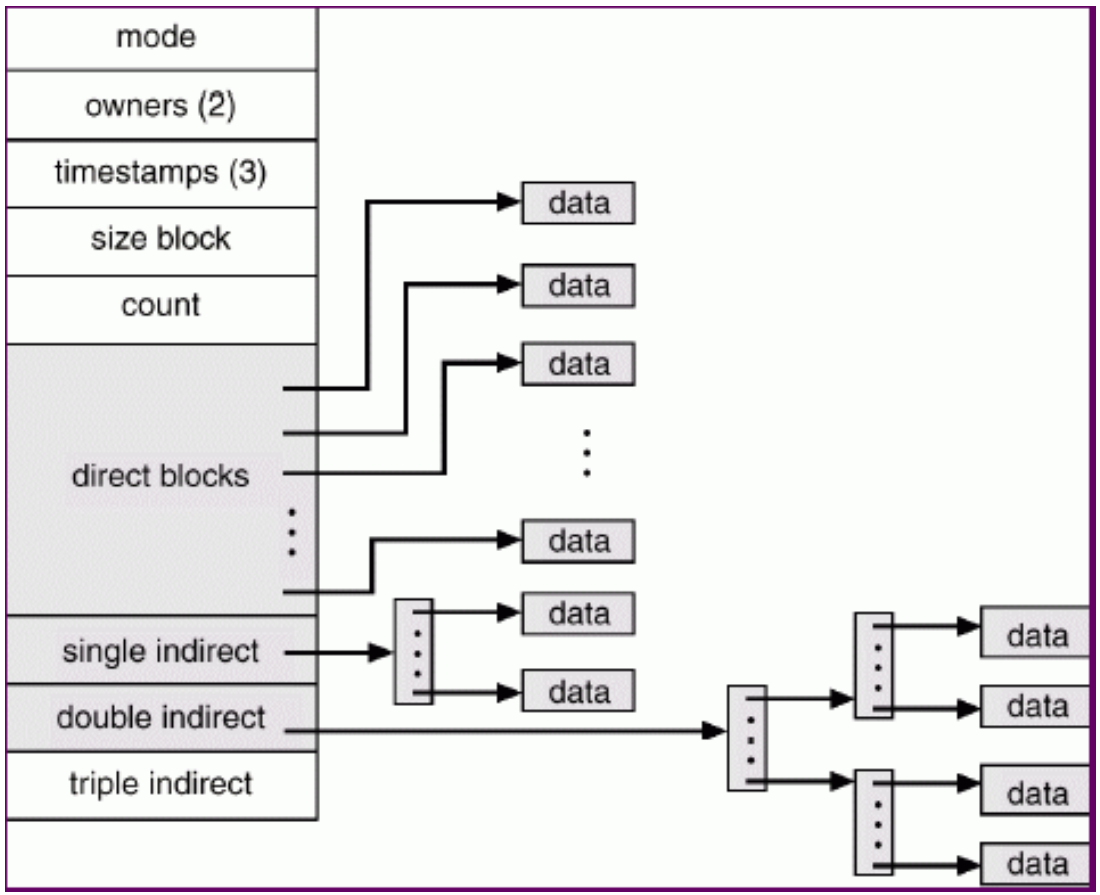


## Utenti, filesystem e file

# File in Linux e comandi sui file

- Linux usa gli **i-node** per memorizzare file su disco
- Un **i-node** (o inode, abbreviazione di *index node*) è una struttura dati sul filesystem che contiene gli *attributi* base di un *filesystem object*, cioè file, directory o altri oggetti
- Le informazioni includono:
  - la dimensione del file e la sua locazione fisica (se risiede su un dispositivo a blocchi, come ad esempio un hard disk)
  - il proprietario e il gruppo di appartenenza
  - le informazioni temporali di modifica (mtime), ultimo accesso (atime) e di cambio di stato (ctime)
  - il numero di collegamenti fisici che referenziano l'inode
  - i permessi d'accesso
  - un puntatore allo spazio del disco che contiene i file veri e propri

- Ogni file del filesystem (directory comprese) è univocamente determinato da un `inode number` ed è rappresentato da una struttura dati `inode`



- Non ci sono mai contemporaneamente 2 file con lo stesso inode number
- Gli inode number *liberati* dalla cancellazione di un file verranno riusati alla prima occasione
- Esiste una **tabella di tutti gli inode**, che si trova solitamente all'inizio del disco

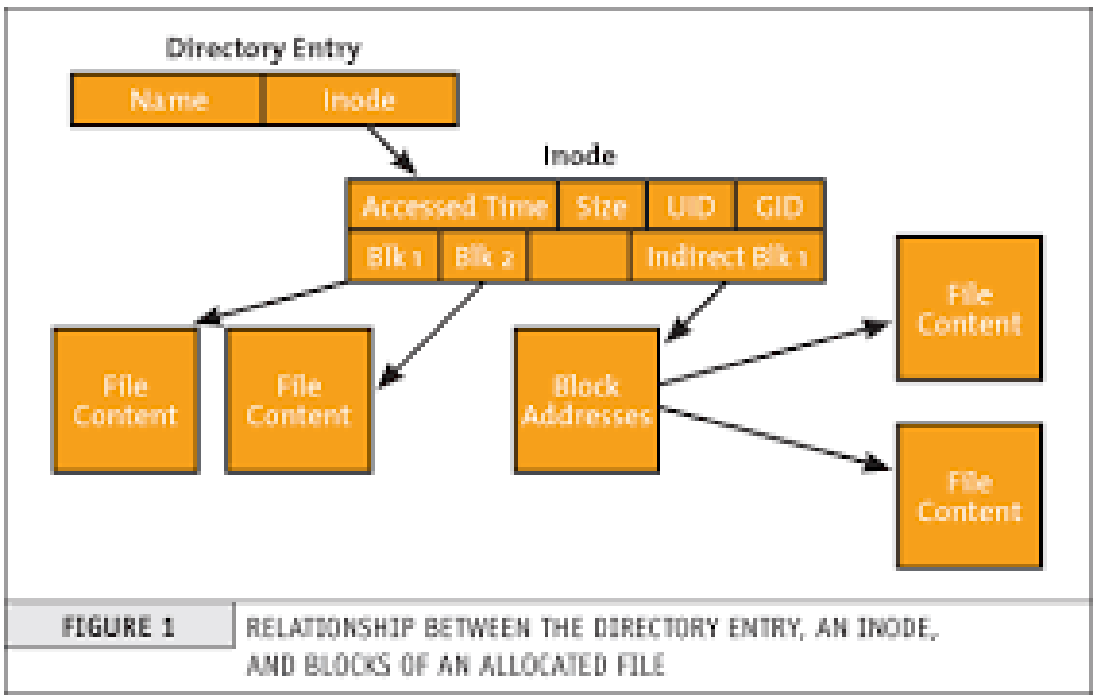



FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

- Tabella dei principali attributi della struttura dati `inode`

| **Attributo** | **Spiegazione** |
|---|---|
| Type | Tipo di file (regular, block, fifo ...) |
| User ID | Id dell'utente proprietario del file |
| Group ID | Id del gruppo a cui è associato il file |
| Mode | Permessi (read, write, exec) di accesso per il proprietario, il gruppo e tutti gli altri |
| Size | Dimensione in byte del file |
| Timestamps | ctime (inode changing time: cambiamento di un attributo), mtime (content modification time: solo scrittura), atime (content access time: solo lettura) |
| Link Count | Numero di hard links |
| Data Pointers | Puntatore alla lista dei blocchi che compongono il file; se si tratta di una directory, il contenuto su disco è costituito di una tabella con 2 colonne: nome del file/directory e suo inode number |

# Il comando `ls`

- Per vedere il valore degli attributi, occorre usare in modo più esteso il comando `ls`:
  `ls [-a] [-c] [-u] [-R] [-l] [-i] [-n] [-S] [-h] [-1] [nomedir1 ... nomedirn] [nomefile1 ... nomefilen]`
- Si possono passare *indifferentemente directory e files*: si otterrà il contenuto delle directory indicate e i files indicati
- Per vedere l'inode number dei file e delle directory, usare l'opzione `-i`
- Per vedere l'ID (numerico) di utente e gruppo, anzichè il corrispettivo nome, usare l'opzione `-n`
- Per vedere i timestamp, si usa l'opzione `-l` in combinazione con `-c` (per `ctime`) o con `-u` (per `atime`) e senza niente (per `mtime`)

# Il comando `ls`

- Per vedere user name, group name, dimensione e permessi usare l'opzione `-l`
  - **per i file** la dimensione è quanto effettivamente occupato da quel file su disco, in bytes
  - **per le directory** viene mostrata la dimensione del file speciale corrispondente, che consiste in una lista di coppie (nomefile, numero di inode), quindi non conta quanto siano grandi i file contenuti, ma *quanti* sono (e quanto sono lunghi i loro nomi)
  - solitamente, le directory hanno una dimensione minima di 4kB, anche se sono vuote

# Il comando `ls`

- L'opzione `-l` mostra anche una riga iniziale *total* per ogni directory di cui mostra il contenuto: si tratta delle dimensione dei file contenuti in quella directory, come numero di *blocchi* su disco (normalmente, 1 blocco ha dimensione di un 1kB)
  - si tratta solo di una comodità per l'utente, non dei blocchi con cui il kernel comunica con il disco
  - per vedere l'output in Mega, cioè $10^6$, si può scrivere `ls --block-size=M`, per vederlo in MegaByte, cioè $2^{20}$, si può scrivere `ls --block-size=MB`, mentre con `ls --block-size=k` e `ls --block-size=kB` si hanno le dimensioni rispetto a $1000 = 10^3$ e a $1024 = 2^{10}$ (kbyte) rispettivamente
  - N.B. le dimensioni sono tutte approssimate per eccesso

# Il comando `ls`

- Notare che queste ed altre informazioni non sono sul `man`: vanno cercate nelle informazioni estese, reperibili con il comando `info ls`

- **Esercizio** verificare le dimensioni con `ls --block-size=M`, `ls --block-size=MB`, `ls --block-size=k` e `ls --block-size=kB` posizionandosi in diverse directory

- **Esercizio** creare un file, modificarlo, poi leggerlo e verificare che i valori per `atime` ed `mtime` cambino di conseguenza

# Il comando `touch`

- Qualche dettaglio in più su `touch [-a] [-m] [-t timestamp] {file}`
  - Il suo *vero* uso è quello di cambiare i timestamps negli attributi dei file dati
  - Li cambia normalmente tutti e tre, a meno che non si sia data l'opzione `-a` (solo atime) o `-m` (solo mtime)
  - Come *effetto collaterale*, se un file dato non esiste lo crea
  - Può essere usato anche su una directory (ma solo se esiste)
  - Con `-t timestamp` setta i timestamp del file al timestamp dato, anziché al timestamp del tempo attuale

# Il comando cp

- Comando `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`: permette di copiare file e directory
  - 2 modalità basilari:
    - con 2 argomenti, `filedestinazione` può essere una directory od un file (in questo caso, ovviamente, il sorgente dev'essere a sua volta un file...)
    - con 2 argomenti, la destinazione può non esistere, e verrà creata (un file se la sorgente è un file, una directory se la sorgente è una directory)
    - con più di 2 argomenti, `filedestinazione` dev'essere una directory (esistente)
    - se si vuole copiare un file che si trova in un'altra directory nella cwd (mantenendone il nome), il secondo argomento sarà `.`

# Il comando cp

- Comando `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`
  - Tra i `filesorgenti` ci possono essere file e/o directory
  - Ovviamente, se la destinazione è una directory esistente, i sorgenti vengono copiati *dentro* la directory destinazione (anche se il sorgente è una directory)
  - Se ci sono directory tra i sorgenti, allora occorre dare l'opzione `-r`, altrimenti quelle directory non verranno copiate (verranno copiati solo i file)
  - Se la copia avviene su file esistenti, verranno sovrascritti; con l'opzione `-i`, prima di sovrascrivere viene chiesta conferma

# Il comando cp

- Comando `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`
  - Con l'opzione `-u`, la sovrascrittura avviene solo se l'mtime del sorgente è più recente di quello della destinazione (o quello di destinazione non esiste)
    - **Esercizio** Vedere il comando `rsync`, più completo
  - I permessi del file sorgente potrebbero non venire preservati: sono soggetti alle regole del comando `umask` che vedremo
  - Per forzare a mantenere i permessi, c'è l'opzione `-a` (che serve anche per altri motivi)

# Il comando cp

- Comando `cp [-r] [-i] [-a] [-u] {filesorgenti} filedestinazione`
  - **Esercizio** verificare il funzionamento di ciascuna delle opzioni mostrate:.
    - per l'opzione `-i`, provare a copiare un file sorgente in un file destinazione esistente e vedere che effettivamente viene chiesta conferma
    - per l'opzione `-a`, è sufficiente creare un file ed aggiungere qualche permesso, poi fare la copia con e senza `-a`
  - **Esercizio** verificare il funzionamento del comando cp con diverse tipologie di argomenti: i) 2 file (tutti e 4 i casi tra esistenti e non); ii) 2 directory (tutti e 4 i casi tra esistenti e non); iii) 3 file (l'ultimo sia esistente che non); iv) 3 directory (idem); v) 2 file e 2 directory...

# Il comando `mv`

- Comando: `mv [-i] [-u] [-f] {filesorgenti} filedestinazione`
  - Come cp, ma serve a *spostare* anziché *copiare*: quindi, i sorgenti risulteranno *cancellati* dopo l'esecuzione del comando, ed esisteranno solo nella destinazione
  - Con 2 argomenti omologhi (2 file o 2 directory) effettua in pratica una *ridenominazione*
  - Le opzioni `-i` e `-u` hanno lo stesso significato di cp; `-f` è il contrario di `-i` (ed è l'opzione di default)
  - **Esercizio** rifare entrambi gli esercizi visti per cp

# Il comando `rm`

- Comando `rm [-f] [-i] [-r] {file}`: per cancellare
  - Cancella definitivamente i file e le directory indicati (non completamente vero nel caso di hard link, che vedremo)
  - Le opzioni `-f` e `-i` sono come quelle della `mv` (ma stavolta il default è `-i`); l'opzione `-r` è come quella della `cp`
  - **Esercizio** creare una directory, poi crearci dentro un file, togliere il permesso di scrittura a quest'ultimo e cancellarlo. Lo lascia fare?
  - **Esercizio** creare un file, guardare il suo inode, cancellarlo e poi creare subito dopo un altro file: qual è l'inode del nuovo file (supponendo che nessun altro utente stia usando il computer)?