Operazione Rif. PA 2023-19410/RER approvata con DGR 1317/2023 del 31/07/2023 finanziata con risorse del Programma Fondo sociale europeo Plus 2021-2027 della Regione Emilia –Romagna.

Progetto n. 1 - Edizione n. 1

TECNICO PER LA PROGETTAZIONE E LO SVILUPPO DI APPLICAZIONI INFORMATICHE

MODULO: N. 5 Titolo: SICUREZZA DEI SISTEMI INFORMATICI E DISPIEGO DELLE APPLICAZIONI
DURATA: 21 ORE DOCENTE: MARCO PRANDINI

FILESYSTEM LINUX
UTENTI, FILE E PERMESSI

Navigazione

- `pwd` mostra la directory corrente di lavoro
- `cd` permette di spostarsi a un'altra directory
  - esplicitamente nominata, oppure
  - la home dell'utente se invocato senza parametri, oppure
  - la directory in cui ci si trovava prima dell'ultimo cd se invocato con `-`
- ricordiamo che in ogni directory D sono sempre presenti due sottodirectory
  - `.` che coincide con la directory D stessa
  - `..` che coincide con la directory superiore (in cui D è contenuta)

Collocazione delle risorse

- FHS (Filesystem Hierarchy Standard) definisce la struttura del filesystem Unix allo scopo di rendere più facile a programmi automatici ed utenti l'individuazione delle risorse, rendere più efficiente la condivisione di parti del filesystem e rendere più sicura la memorizzazione dei dati.

  http://www.pathname.com/fhs/pub/fhs-2.3.html

- Le distinzioni base che guidano alla corretta collocazione dei dati in FHS sono 2:

	condivisibili	non condivisibili
statici	es. /usr /opt	es. /etc /boot
variabili	es. /var/mail /var/spool/news	es. /var/run /var/lock

root directory (/)

- E' l'origine della gerarchia. Deve contenere tutti i dati necessari all'avvio del sistema, ma (storicamente) essere il più compatto possibile per ridurre i rischi di corruzione accidentale e poter essere alloggiato in media di scarsa capacità. Per rispondere a questi requisiti ed inoltre ospitare i punti iniziali di sottogerarchie più flessibili, deve contenere:

```
/ -- the root directory
      +-bin       Essential command binaries
      +-boot      Static files of the boot loader
      +-dev       Device files
      +-etc       Host-specific system configuration
      +-lib       Essential shared libraries and kernel modules
      +-mnt       Mount point for (temp) mounting a filesystem
      +-opt       Add-on application software packages
      +-sbin      Essential system binaries
      +-tmp       Temporary files
      +-usr       Secondary hierarchy
      +-var       Variable data
```

ROOT filesystem – alcuni componenti

`/dev`

La directory /dev contiene i file che costituiscono il punto di accesso, per i programmi utente, agli apparati connessi al sistema. Questi file sono essenziali per il funzionamento del sistema stesso.

`/etc`

La directory /etc è riservata ai file di configurazione locali del sistema. In /etc non devono essere messi eseguibili binari. I binari che in passato erano collocati in /etc devono andare in /sbin o /bin

X11 e skel devono essere subdirectories di /etc/

```
/etc
  |- X11
  +- skel
```

La directory X11 è per i file di configurazione del sistema X Window, come XF86Config. La directoy skel è per i prototipi dei file di configurazione delle aree utente.

ROOT filesystem – alcuni componenti

`/lib`

La directory /lib deve contenere solo le librerie richieste per il funzionamento dei programmi che si trovano in /bin e /sbin.

`/proc`

La directory /proc contiene file speciali che permettono di ottenere informazioni dal kernel o di inviare run-time informazioni al kernel, e merita di essere esplorata con attenzione.

`/sbin`

La directory /sbin è riservata agli eseguibili utilizzati solo dall'amministratore di sistema, possibilmente solo quelli necessari al boot ed al mount dei filesystem. Qualunque cosa eseguita dopo che /usr sia stato montato correttamente dovrebbe risiedere in /usr/sbin o in /usr/local/sbin

Come minimo devono essere presenti in /sbin i seguenti programmi:

clock, getty, init, update, mkswap, swapon, swapoff, halt, reboot, shutdown, fdisk, fsck.*, mkfs.*, lilo, arp, ifconfig, route

/usr

La directory /usr è per i file condivisibili e statici. Risiede di preferenza su di una propria partizione, e dovrebbe essere montata read-only. Subdirs:

```
/usr
  |- X11R6                 X Window System
  |- bin                   eseguibili
  |- dict
  |- doc                   documentazione diversa dalle man pages
  |- etc                   file di configurazione validi per il sito
  |- games
  |- include               C header files
  |- info                  GNU info files
  |- lib                   librerie
  |- local
  |- man                   man pages
  |- sbin                  programmi linkati staticamente
  |- share
  +- src                   codice sorgente
```

VAR filesystem

La directory /var è riservata ai file non statici, sia condivisibili che non, ad esempio i file di log, di spool, di lock, di amministrazione e temporanei. Dovrebbe contenere le seguenti subdirs:

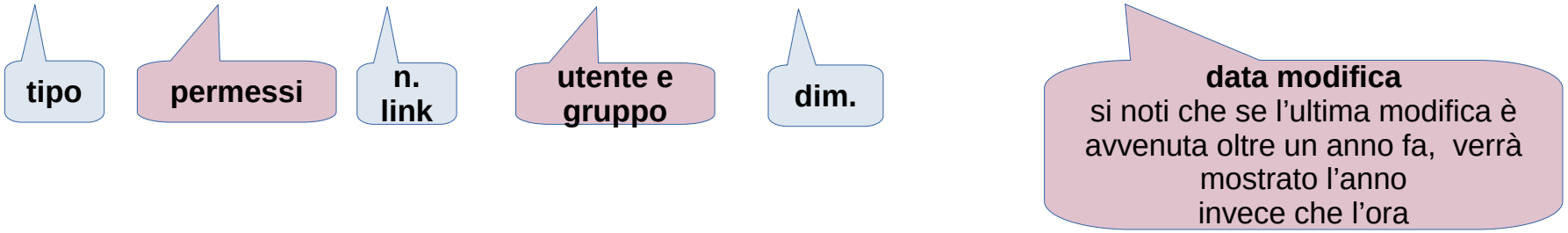
```
/var
  |- spool--------------+- at
  |- log                |- cron
  |- catman             |- lpd
  |- lib                |- mail
  |- local              |- mqueue
  |- named              |- rwho
  |- nis                |- smail
  |- preserve           |- uucp
  |- run                +- news
  |- lock
  +- tmp
```

Opzioni principali di ls

- `-l` abbina al nome le informazioni associate al file
- `-a` non nasconde i nomi dei file che iniziano con .
  - per convenzione i file di configurazione iniziano con un punto, non essendo interessanti per l'utente non sono mostrati di default da ls
- `-A` come -a ma esclude i file particolari `.` e `..`
- `-F` pospone il carattere `*` agli eseguibili e `/` ai direttori
- `-d` lista il nome delle directory senza listarne il contenuto
  - il comportamento di default di ls quando riceve come parametro una directory è di elencarne il contenuto, cosa spesso indesiderabile quando nomi di file e directory vengono espansi dalla shell a partire da wildcard
- `-R` percorre ricorsivamente la gerarchia
- `-i` indica gli i-number dei file oltre al loro nome
- `-r` inverte l'ordine dell'elenco
- `-t` lista i file in ordine di data/ora di modifica (dal più recente)

I metadati principali mostrati da ls -l

```
vagrant@bullseye:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1514 Mar 29 11:07 /etc/passwd
```



- tipi:
  - `-` file standard
  - `d` directory
  - `l` link simbolico
  - `b` block special (device)
  - `c` character special (device)
  - `p` named pipe (FIFO)
  - `s` socket

Le marcature temporali (timestamp)

- Ogni file ha tre (o quattro) timestamp distinti
  - mtime modification time istante dell'ultima **modifica del contenuto**
  - atime access time istante dell'ultimo **accesso al contenuto**
  - ctime change time istante dell'ultima **modifica ai metadati**
  - wtime birth time istante della **creazione** del file, **se supportato dal filesystem**
- Queste informazioni vengono gestite automaticamente dal filesystem, ma possono essere cambiate a mano col comando **touch**
- Tutti i metadati possono essere estratti e visualizzati in un formato arbitrario col comando **stat**

```
stat --format='%U %a %z' /etc/passwd
root 644 2021-03-15 08:33:06.381876582 +0100
```

(U=utente proprietario, a=permessi, z=ctime)

Creazione e rimozione di file

- `rm` cancella un file o, meglio, rimuove il link
  - "garbage collection" - il file viene cancellato quando il link count = 0
  - link count = n. link sul filesystem + n. open file descriptors
- `cp` copia un file o più file in una directory
  - attenzione ai file speciali: copio il "concetto" o il contenuto?
- `mv` sposta un file o più file in una directory
- `ln` crea un link ad un file
  - hardlink di default, solo all'interno dello stesso FS e non verso directory
  - symlink con l'opzione `-s`, nessuna limitazione
- `mkdir` crea una directory
- `rmdir` cancella una directory
  - deve essere vuota
  - `rm -r` cancella ricorsivamente

Ricerca nel filesystem con find

- **find** ricerca in tempo reale
  - quindi esplorando il filesystem → attenzione al carico indotto!

  i file che soddisfano una combinazione di criteri, ad esempio:
  - nome che contenga una espressione data
  - timestamp entro un periodo specificato
  - dimensione compresa tra un minimo e un massimo
  - tipo specifico (file, dir, link simbolici, …)
  - di proprietà di un utente o di un gruppo specificati (o “orfani”)
  - permessi di accesso specificati

  e molti altri
- Esempio:
  - ricercare sotto **/usr/src** tutti i file che finiscono per **.c**, hanno dimensione **maggiore di 100K**, ed elencarli sullo standard output:

```
find /usr/src -name '*.c' -size +100k -print
```

Esecuzione di operazioni sui file trovati

- Una delle opzioni più potenti di find permette, per ciascun oggetto individuato secondo i criteri impostati, di invocare l'esecuzione di un comando:
- Es. mostra il contenuto dei file trovati

```
find /usr/src -name '*.c' -size +100k -exec cat {} \;
```

  - il comando che segue `-exec` viene lanciato per ogni file trovato
  - la sequenza `{}` viene sostituita di volta in volta con il nome del file
  - `\;` è necessario per indicare a find la fine del comando da eseguire

- Es. elenca solo i file regolari "orfani" modificati meno di due giorni (2*24 ore) fa che contengono TXT

```
find / -type f -nouser -mtime -2 -exec grep -l TXT {} \;
```

Ricerca di file con locate

- **locate** effettua la ricerca su di un database indicizzato
  - Il database deve essere aggiornato periodicamente con l'utility updatedb
- Vantaggi su find
  - Carico sul sistema ridotto a una singola esplorazione per ogni periodo, indipendentemente dal numero di query successive
  - Esplorazione pianificabile nei momenti di basso carico
  - Risposta pressoché istantanea
- Svantaggi rispetto a find
  - Unico criterio di ricerca: pattern nel nome
  - Risposte potenzialmente obsolete
    - file creati dopo l'esplorazione non vengono riportati
    - file cancellati dopo l'esplorazione sembrano ancora esistere

Identificazione del contenuto di file

- In Linux, le estensioni dei nomi hanno come unico utilizzo quello di renderli più leggibili all'utente
- Si può ottenere manualmente l'identificazione con **file**
  - test 1: usa stat per capire se il file è vuoto o speciale
  - test 2: usa il database dei **magic number** per identificare il file
  - test 3: usa metodi empirici per capire se è un file di testo, e in tal caso quale sia la lingua naturale o linguaggio di programmazione

I due formati dei file di testo

- nei sistemi UNIX le linee sono terminate da un carattere:
  - **line feed** o **LF** o **\n** o **0x0A**
- nei sistemi DOS/Windows le linee sono terminate da due caratteri:
  - **carriage return line feed** o **CRLF** o **\r\n** o **0x0D0A**

- senza conversione opportuna
  - file di origine DOS, su sistemi UNIX hanno caratteri extra a fine linea
    - comunemente visualizzati dagli editor come **^M**
    - possono causare errori negli script e nei file di configurazione
  - file di origine UNIX, su sistemi DOS confondono le linee

- strumenti Linux
  - alcuni protocolli di rete convertono automaticamente
  - command line Ubuntu: pacchetto **tofrodos** → comandi `todos` / `fromdos`
  - nomi alternativi su altre distro, es. `unix2dos` / `dos2unix`

Archiviazione di file

- Per poter agevolmente memorizzare e trasferire una molteplicità di file, eventualmente senza perdere le proprietà associate a ciascuno (ownership, permessi, timestamps…) è comune avvalersi di **tar**. La sintassi prevede che debba essere specificato esattamente uno dei seguenti comandi:

`-A`	concatena più archivi
`-c`	crea un nuovo archivio
`-d`	trova le differenze tra archivio e filesystem
`-r`	aggiunge file ad un archivio
`-t`	elenca il contenuto di un archivio
`-u`	aggiorna file in un archivio
`-x`	estrae file da un archivio
`--delete`	cancella file da un archivio

Archiviazione di file

- Le origini di **tar** risalgono ai tempi dei nastri magnetici (il nome è acronimo di Tape ARchiver) quindi di default assume che l'archivio sia su **/dev/tape**.
- L'opzione `-f <FILENAME>` viene quindi sempre usata per specificare un file di archiviazione.
  - Dove sensato, **FILENAME** può essere **-** in per indicare
    - lo standard input da cui leggere un archivio con `d`, `t`, `x`
    - lo standard output su cui scrivere l'archivio con `c`
- Altre opzioni comunemente usate sono:

  `-p` (preserve) conserva tutte le informazioni di protezione
    - (funziona pienamente solo per root, un utente standard quando ricrea i file estraendoli da un archivio è forzato a dargli la sua ownership)

  `-v` stampa i dettagli durante l'esecuzione

  `-T <ELENCO>` prende i nomi dei file da archiviare da ELENCO invece che come parametri sulla riga di comando

  `-C <DIR>` svolge tutte le operazioni come dopo `cd DIR`

Archiviazione di file

- Esempi (si noti che il trattino per indicare le opzioni può essere omesso fintanto che non è necessario utilizzare più di un'opzione che richiede parametri)
- creazione

```
tar cvpf users.tar /home/*
```

  - la barra iniziale verrà rimossa in modo da rendere relativi tutti i path

- estrazione

```
tar -C /newdisk -xvpf users.tar
```

  - poiché i path nell'archivio sono relativi, la directory home viene ricreata dentro /newdisk e tutta la gerarchia sottostante viene ricostruita

- pipeline

```
tar cvpf - /home/* | tar -C /newdisk -xvpf -
```

Compressione di file

- tar non comprime
- esistono moltissimi formati di compressione
  - https://linuxhint.com/top_10_file_compression_utilities_on_linux/
  - https://en.wikipedia.org/wiki/List_of_archive_formats
- I più comuni nei sistemi Linux sono
  - estensione `.gz` comando base: `gzip`
  - estensione `.bz2` comando base: `bzip2`
  - estensione `.xz` comando base: `xz`
- Il comando base prende come argomento un file e lo comprime aggiungendo l'estensione
  - con l'opzione `-d` decomprime ricreando il file e rimuovendo l'estensione
  - con l'opzione `-c` riversa il risultato su STDOUT invece che su file
    - filtro!
    - es: `tar cf - * | xz -c > archive.tar.xz`

Compressione di file - scorciatoie

- Esiste tipicamente un comando di decompressione equivalente al comando base invocato con -d
  - es. `gunzip`, `bunzip2`, `unxz`
- Esistono alias per le combinazioni più comuni di filtro di decompressione e comandi di trattamento testo
  - `zcat file.gz` == `gzip -dc file.gz`
  - `zegrep <REGEX> file.gz` == `gzip -dc file.gz | egrep <REGEX>`
  - (idem per i decompressori `bz*`, `xz*`, e per i comandi `*diff`, `*less`, `*cmp`)
- **tar** in particolare supporta opzioni per invocare direttamente la (de)compressione di un archivio

`-z`	usa `gzip`	estensione `.tar.gz` o `.tgz`
`-j`	usa `bzip2`	estensione `.tar.bz2` o `.tbz2`
`-J`	usa `xz`	estensione `.tar.xz` o `.txz`

  - esempio precedente == `tar cJf archive.tar.xz *`

Copia massiva di file (anche remota)

- Il trasferimento di gerarchie di file e cartelle, contenenti file non standard non è gestito correttamente da tutte le versioni di `cp -a` e `scp -R`
- `tar` archivia correttamente tutti i metadati
  - prima possibilità:
    - creare un archivio
    - (eventualmente trasferirlo con scp su un altro host)
    - estrarlo nella cartella di destinazione
- alternativa più evoluta: `rsync`
  - Possibilità di non trasferire file già presenti a destinazione
  - Possibilità di trasferire solo le differenze tra un file sorgente e il corrispondente file a destinazione
  - Comportamento con file speciali configurabile
  - Criteri flessibili di inclusione ed esclusione

Copia massiva di file con rsync

- Sintassi base del comando client

  `rsync [OPZIONI] SORGENTE DESTINAZIONE`
- Copia locale (e remota dove usati negli esempi seguenti)
  - SORGENTE = elenco di file e cartelle
  - DESTINAZIONE = cartella
- Copia via rete con protocollo nativo
  - da / verso host su cui gira il demone rsyncd

    `rsync [USER@]HOST::SRCDIR DESTINAZIONE`

    `rsync SORGENTE [USER@]HOST::DESTDIR`
- Copia via rete via SSH
  - no demone rsyncd richiesto

    `rsync [USER@]HOST:SRCDIR DESTINAZIONE`

    `rsync SORGENTE [USER@]HOST:DESTDIR`

Alcune opzioni di rsync

- Come copiare

`-l` / `-L`	copia i link come link / come file puntato
`-p` / `-o` / `-g`	preserva i permessi, il proprietario, il gruppo
`-t` / `-a` / `-N`	preserva i ts di modifica / accesso / creazione
`-D`	preserva i file speciali
`-a`	= `-rlptgoD`
`-b`	backup (come? `--backup-dir` / `--suffix`)

- Cosa copiare

`-r`	ricorsivo
`-u`	salta i file che sono più nuovi a destinazione o che a parità di età hanno la stessa dimensione
`-c`	salta i file che a destinazione hanno lo stesso checksum
`--exclude`	specifica path da non includere nella copia

Laboratorio

- (anticipiamo un semplice costrutto shell: la ridirezione)
- cerchiamo file con caratteristiche specifiche
- creiamo archivi contenenti tali file, in diversi formati
- estraiamo gli archivi in posizioni controllate

Dischi e filesystem

Device files di uso comune

- Alcuni device files notevoli che rappresentano vere periferiche:

`/dev/tty*`	terminali fisici del sistema
`/dev/pts/*`	pseudo-terminali (dentro finestre del sistema grafico)
`/dev/sd*`	dischi e partizioni

Il sistema di storage

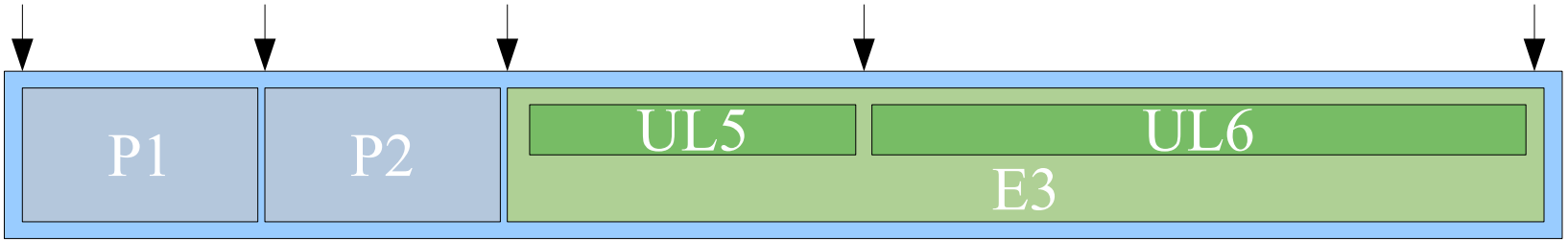
- I dispositivi a blocchi rappresentano tipicamente supporti di storage
  - Reali: hard disk, SSD, usb drive, dischi ottici, ...
  - Logici: sistemi RAID, componenti LVM, …
- Un dispositivo a blocchi è utilizzabile come un semplice elenco di blocchi dati di dimensione fissa, numerati
- Per renderlo fruibile servono tre operazioni
  - Partizionamento
    - Opzionale ma sempre utilizzato
  - Formattazione
    - Creazione dei metadati per organizzare lo spazio in modo comprensibile (filesystem)
  - Mount
    - Associazione dei singoli filesystem alla gerarchia di directory

Partizionamento

- Suddivisione di un disco in sottoinsiemi di blocchi
  - Ogni partizione si presenta come un dispositivo indipendente
  - Utile per separare spazi con esigenze diverse
    - Di organizzazione (filesystem linux vs. Microsoft vs. ...)
    - Di persistenza (reinstallazione sistema vs. dati utente)
    - Di politiche di accesso (dati vs. programmi, …)
- Vari standard
  - Più diffusi su PC: Master Boot Record, GUID Partition Table
  - Più rari o specifici di altre architetture
    - Extended Boot Record
    - Boot Engineering Extension Record
    - Apple Partition Map
    - Rigid Disk Block
    - BSD disklabel

MBR-based

- Tabella principale: max 4 entry (*partizioni primarie*)
  - Semplicemente delimitate da blocco inizio – blocco fine (+ tipo)
- Una di queste può essere contrassegnata *partizione estesa*
  - Lo spazio occupato dalla partizione estesa contiene un'ulteriore tabella
  - Nella tabella possono essere elencate fino a 12 *unità logiche*

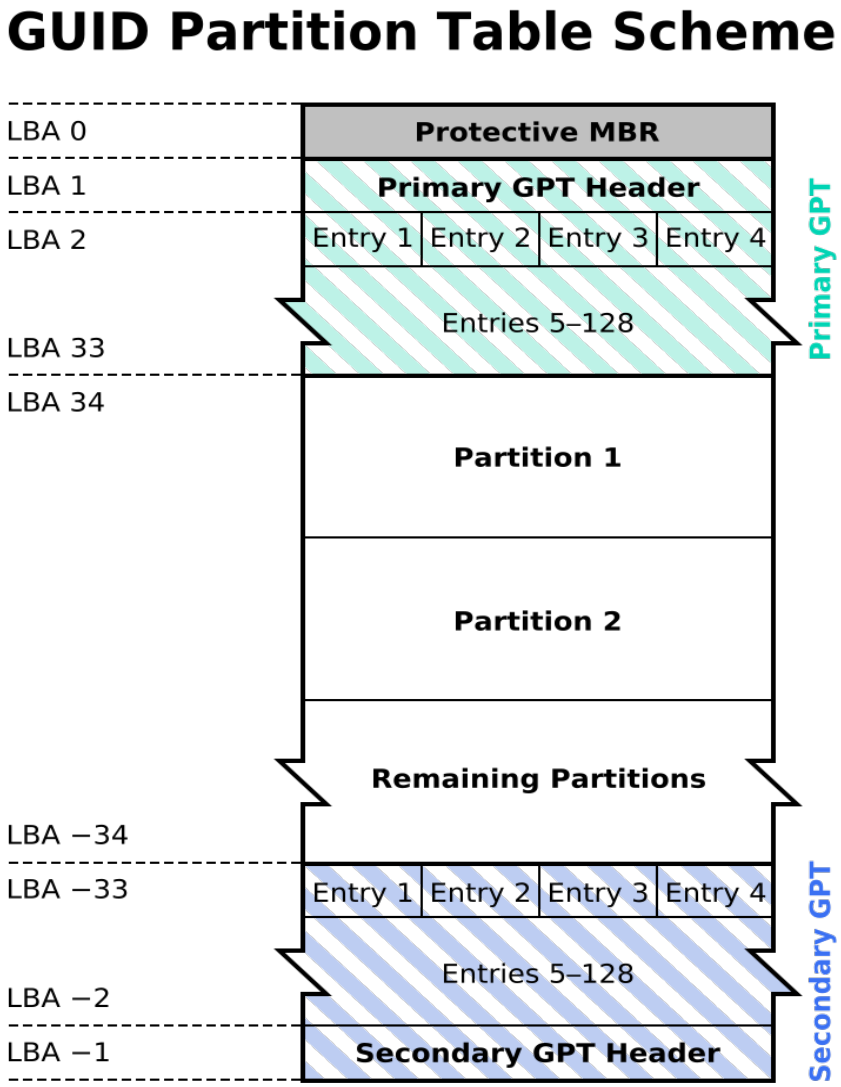


P1: primaria – inizio 0 – fine 50
P2: primaria – inizio 51 – fine 100
E3: estesa – inizio 101 – fine 300
UL5: unità logica – inizio 101 – fine 180
UL6: unità logica – inizio 181 – fine 300

- Dal punto di vista dell'utilizzo, partizioni primarie e unità logiche sono equivalenti

GPT

- Utilizzato specialmente con UEFI, ma volendo anche con BIOS
- MBR max 15 partizioni di 2TiB
- GPT max 128 partizioni di 8 ZiB
- Oltre a limiti e tipo
  - Nome
  - GUID
  - Attributi



https://en.wikipedia.org/wiki/GUID_Partition_Table#/media/File:GUID_Partition_Table_Scheme.svg

Device file per dischi e partizioni

- Comunemente unificati sotto il framework SCSI

  `/dev/sdXXNN` es. `/dev/sda1`

  - XX = una o più lettere che identificano il “disco”
  - NN = numero della partizione
- Tipicamente i nuovi dischi NVMe/M.2 compaiono come

  `/dev/nvmeXnYpZ` es. `/dev/nvme0n1p2`

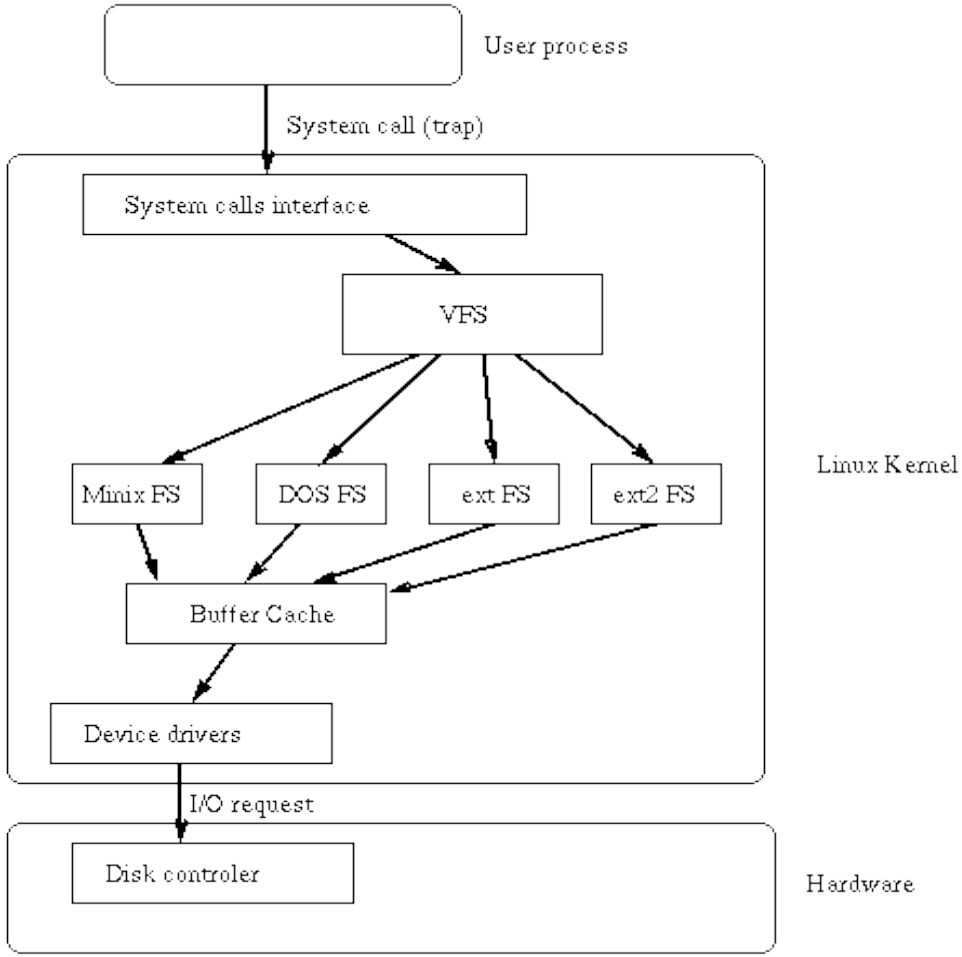
  - X = identificatore del “disco”
  - Y = identificatore del namespace
    - una sorta di macro-partizione hardware
      https://nvmexpress.org/resources/specifications/
  - Z = numero della partizione
- Gli identificatori di disco possono cambiare a seconda dell’ordine in cui i dischi vengono rilevati al boot!

Criteri di partizionamento

- Partizione di *swap* – memoria virtuale
  - In origine consiglio 2xRAM, ma ora la penalità di prestazioni è inaccettabile per usarla davvero come memoria virtuale
  - Uso comune sui laptop: partizione dedicata >>RAM per ibernazione
  - Altri casi: allocare un po' di spazio solo per evitare che un piccolo esubero di uso di memoria mandi in crash il sistema
    - Possibilità sia di partizione separata che di file in partizione dati → file ***sparse***
- Collocazione dati – approccio minimale
  - unica partizione con spazio sufficiente per tutto
- Collocazione dati per funzioni
  - una partizione per ogni "tipo di accesso", ad esempio:
    - una partizione per / (obbligatoria)
    - una partizione per i file di boot
    - una partizione per librerie e applicazioni → sola lettura
    - una partizione per code e log → alto traffico in lettura e scrittura
    - una partizione per aree utente → alto traffico e necessità di quota

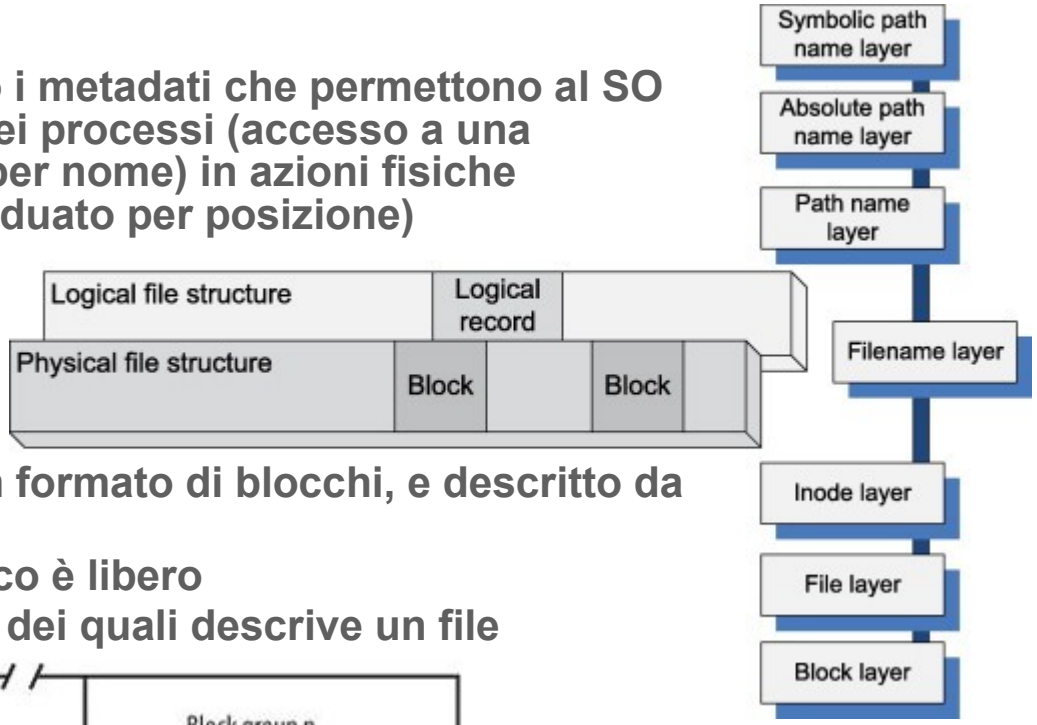
Formattazione

- Crea il filesystem in una partizione
  - Richiede spazio contiguo, può crescere o ridursi ma non può avere “buchi”
- Permette l’accesso ai file secondo il modello del Virtual File System di Linux
  - VFS astrae dall’organizzazione dei dati specifica di diversi FS

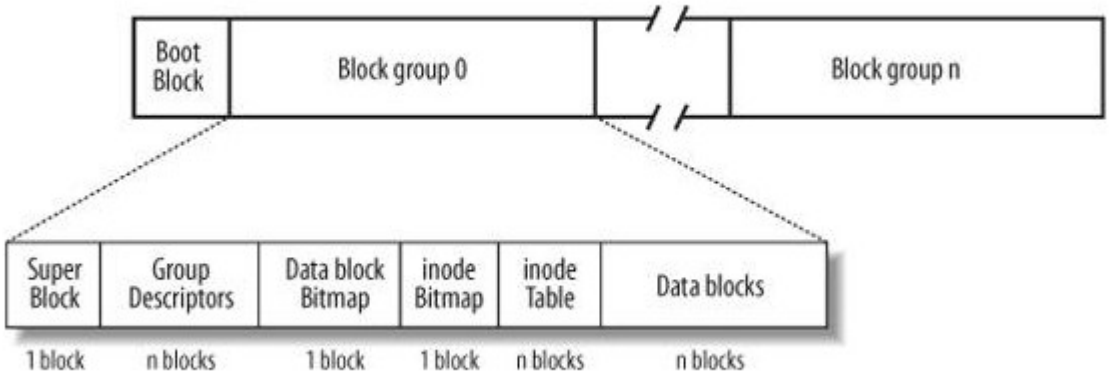


Formattazione

- Con la formattazione si inizializzano i metadati che permettono al SO di trasformare le richieste logiche dei processi (accesso a una porzione logica di un file chiamato per nome) in azioni fisiche (accesso a un blocco di disco individuato per posizione)

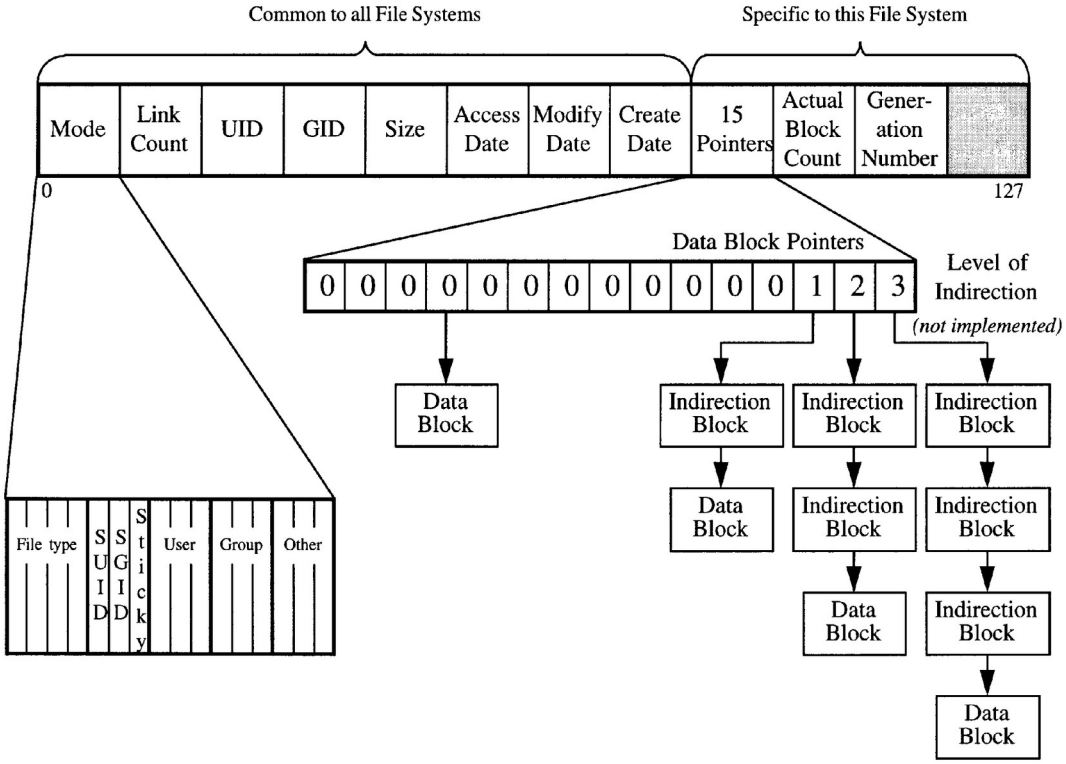


- Ogni partizione ospita un filesystem formato di blocchi, e descritto da un *superblock* che contiene
  - bitmap che indicano se un blocco è libero
  - blocchi speciali (inode) ognuno dei quali descrive un file



Inode e directory

- Un inode contiene tutte le proprietà di un file **tranne il nome** e gli indirizzi dei blocchi che contengono i dati veri e propri



- Una directory è un file speciale che memorizza un indice, che associa **il nome** che vogliamo dare a un file in essa contenuto **all'inode** che lo rappresenta

inode	len	name
51	6	passwd
671	9	wireshark
49	11	thunderbird
1120	3	rpc
...	...	...

`directory /etc`

Filesystem ext2

- Il second extended filesystem è il più tradizionale in Linux.
- Non è journaled, e quindi anche se piuttosto robusto (e molto veloce) non è adatto a realizzare FS di grandi dimensioni, poichè il minimo guasto richiederebbe ore per la rilevazione e riparazione.
- Con blocchi di 4KB, le dimensioni massime sono
  - 2TiB per i file
  - 16TiB per l'intero filesystem
- La struttura è quella del FS Unix, con inodes che supportano fino a due livelli di indirettezza.
- Poichè le directory non sono indicizzate (la ricerca dei file avviene sequenzialmente) è bene non collocare più di 10-15000 file in una directory per non rallentare troppo le operazioni

Filesystem ext3

- ext3 è nato per essere essenzialmente ext2 più un journal, mantenendo la compatibilità col predecessore. Per questo motivo non esibisce le prestazioni dei FS nativamente journaled, ma rispetto a ext2 ha due ulteriori vantaggi significativi:
  - la possibilità di crescere, anche a caldo nelle ultime versioni
  - la possibilità di indicizzare le directory con htree, e quindi di gestire directory contenenti un maggior numero di file
- Il journaling di ext3 può essere regolato su tre livelli:
  - journal - registra sia i dati che i metadati nel journal prima del commit sul filesystem
  - ordered - registra solo i metadati, ma garantisce che ne sia fatto il commit solo dopo che i dati corrispondenti sono stati scritti sul FS
  - writeback - registra solo i metadati senza alcuna garanzia sui dati

Filesystem ext4

- L'ultima evoluzione del filone “ext”, come sempre nata per offrire nuove feature pur consentendo un certo grado di retrocompatibilità, è stata introdotta in forma stabile nel kernel 2.6.28.
- Rispetto ad ext3, aumenta i limiti
  - di dimensione dei singoli file da 2 TiB a 16 TiB
  - di dimensione del filesystem da 16 TiB a 1 EiB
  - di numero di subdirectory da 32.000 a infinito
- Introduce i concetti di:
  - extent al posto del sistema di allocazione indiretta;
    - i file grandi sono allocati in modo contiguo anzichè essere suddivisi in moltissimi blocchi indirizzati in modo indiretto
  - allocazione dei blocchi a gruppi;
    - l'allocatore dei blocchi disco può essere usato per riservarne più di uno per volta
  - allocazione ritardata;
    - l'allocatore è invocato solo quando c'è l'effettiva necessità di scrivere su disco
  - journal checksumming
    - la consistenza del journal è ottenuta con un checksum invece che con una procedura di commit a due fasi, migliorando affidabilità e velocità

Filesystem ext4 (continua)

- Caratteristiche particolarmente interessanti per i sistemi embedded e real-time sono:
  - journal disattivabile
    - elimina la causa delle maggiori lentezze di accesso ai dischi SSD e l'eccesso di scritture concentrate in pochi blocchi (usura);
  - inode più grandi
    - come sopra, poichè molti attributi estesi potranno essere ospitati direttamente nell'inode
  - inode reservation
    - aumenta il determinismo dei tempi di creazione dei file nelle directory
  - persistent preallocation
    - aumenta il determinismo dei tempi di scrittura dei dati in un file
- Le caratteristiche di preallocation e reservation sono particolarmente importanti. Per contro, l'allocazione ritardata (di default) riduce la resistenza agli spegnimenti bruschi
- Dettaglio pratico: di default la formattazione ext4 riserva il 5% dei blocchi a root – è una strategia utilissima per evitare situazioni critiche, ma uno spreco per partizioni dati pure → disattivabile

Altri Journaled FS

- **ReiserFS - Il primo journaled FS ad essere inserito nel kernel di linux, è teoricamente molto performante su sistemi che gestiscono grandi quantità di file di piccole dimensioni, ma ha attratto critiche in merito alla sua stabilità. Inoltre lo sviluppatore capo ha imposto un modello molto conflittuale nei confronti della comunità, ed ha perso definitivamente credibilità per i suoi guai giudiziari.**
- **XFS - Sviluppato da Silicon Graphics per IRIX e JFS - Sviluppato da IBM per AIX, sono stabili, maturi, nativamente a 64bit (quindi su S.O. altrettanto a 64bit possono reggere partizioni da 8 ExaByte) ed offrono diverse ottimizzazioni molto vantaggiose; la loro scarsa diffusione è probabilmente dovuta solo alla consuetudine della maggior parte degli utenti Linux verso ext2/3**

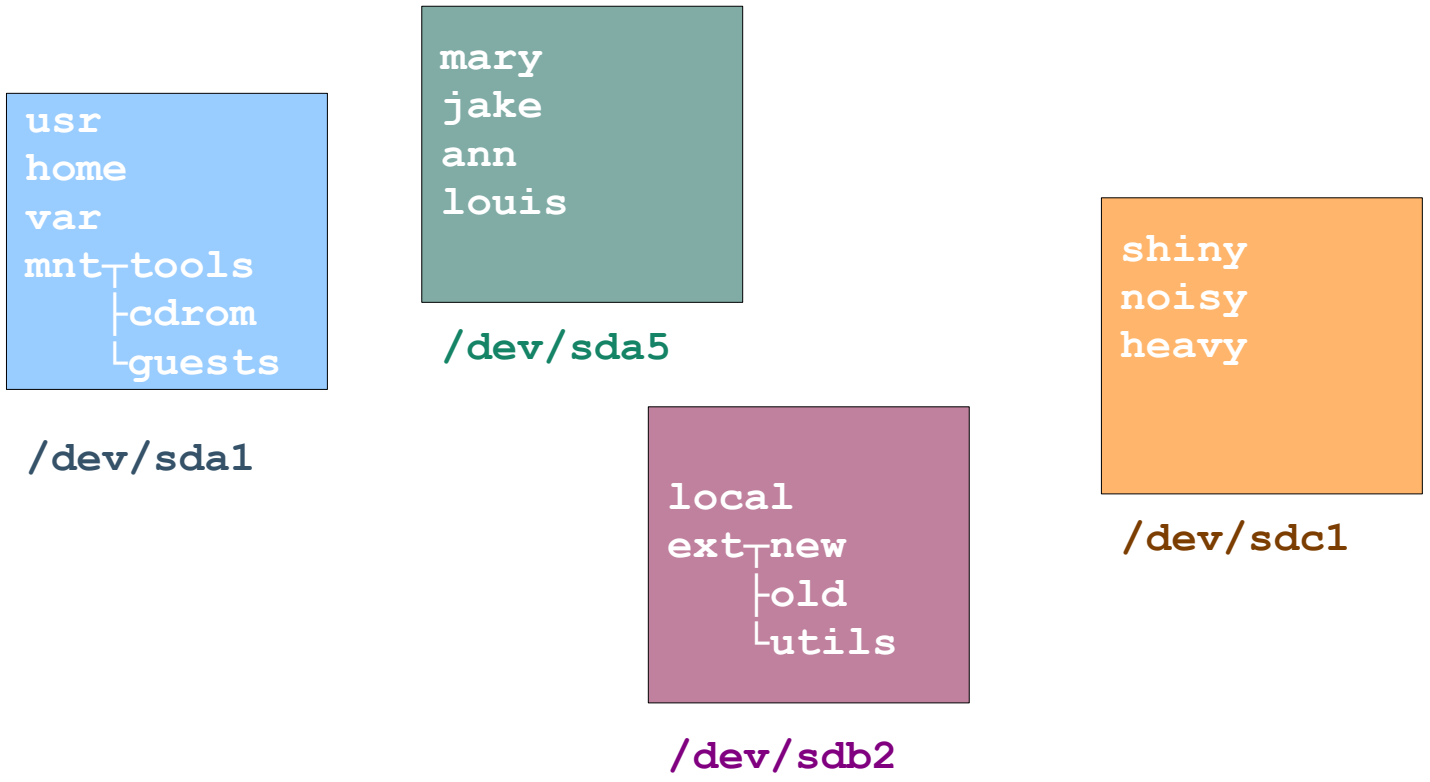
Il futuro: btrfs?

- **B-tree FS, abbreviato in btrfs e pronunciato “Butter F S”, è un progetto sviluppato da Oracle sotto licenza GPL. http://btrfs.wiki.kernel.org/**

- **Gli obiettivi sono quelli di integrare funzionalità per la scalabilità enterprise, simili a quelle offerte da ZFS di Sun:**
  - **pooling di risorse HW, multi-device spanning**
  - **copy-on-write (versioning, writable snapshots, uso di supporti RO)**
  - **checksum su dati e metadati**
  - **compressione/deframmentazione/checking on line**

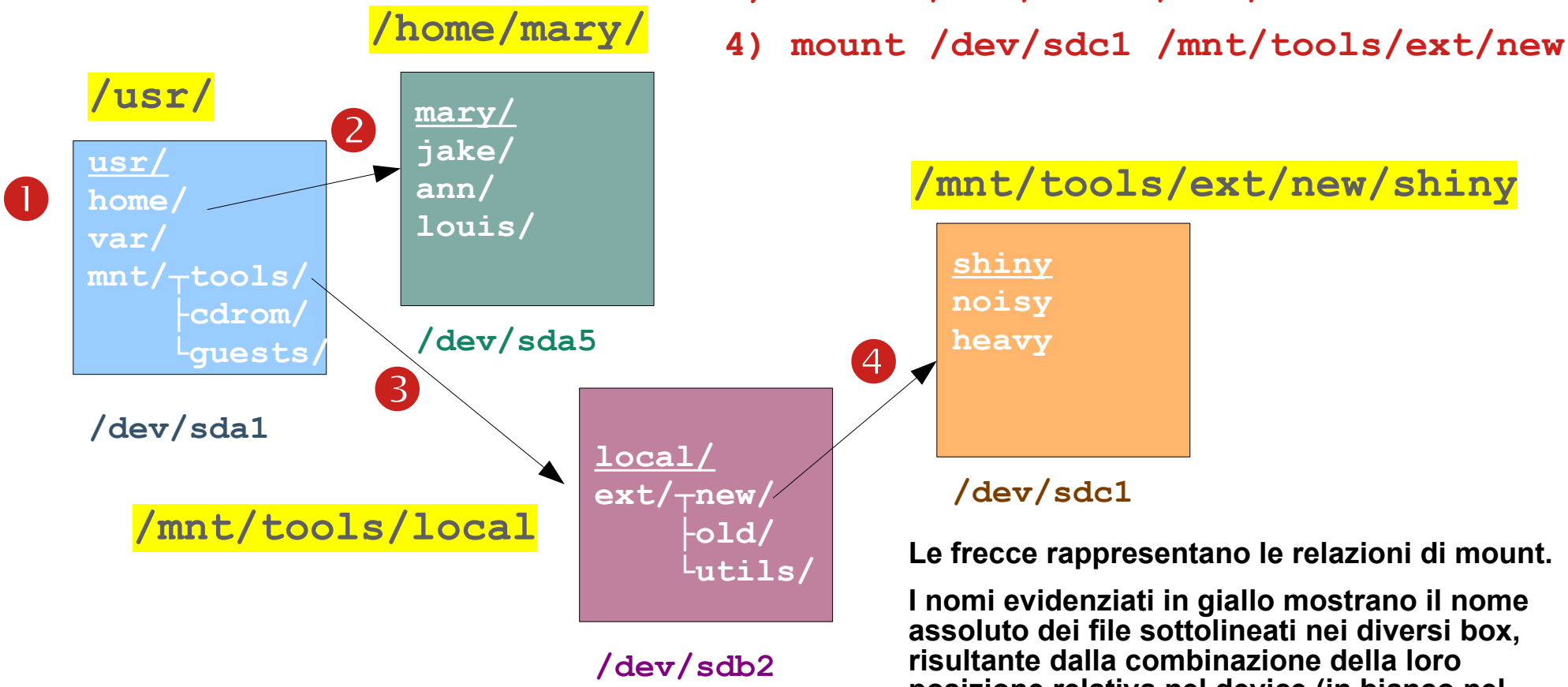
Mount

- Il partizionamento crea una gerarchia locale di directory
- L'operazione di *mount* la "innesta" nella gerarchia globale



Mount

```
1) mount /dev/sda1 /
2) mount /dev/sda5 /home
3) mount /dev/sdb2 /mnt/tools
4) mount /dev/sdc1 /mnt/tools/ext/new
```



Le frecce rappresentano le relazioni di mount.

I nomi evidenziati in giallo mostrano il nome assoluto dei file sottolineati nei diversi box, risultante dalla combinazione della loro posizione relativa nel device (in bianco nel box) e del punto di mount dello stesso

Mount automatico delle partizioni

- L'associazione tra partizione e mount point è mantenuta nel file `/etc/fstab` perchè all'avvio del sistema il filesystem possa essere automaticamente predisposto

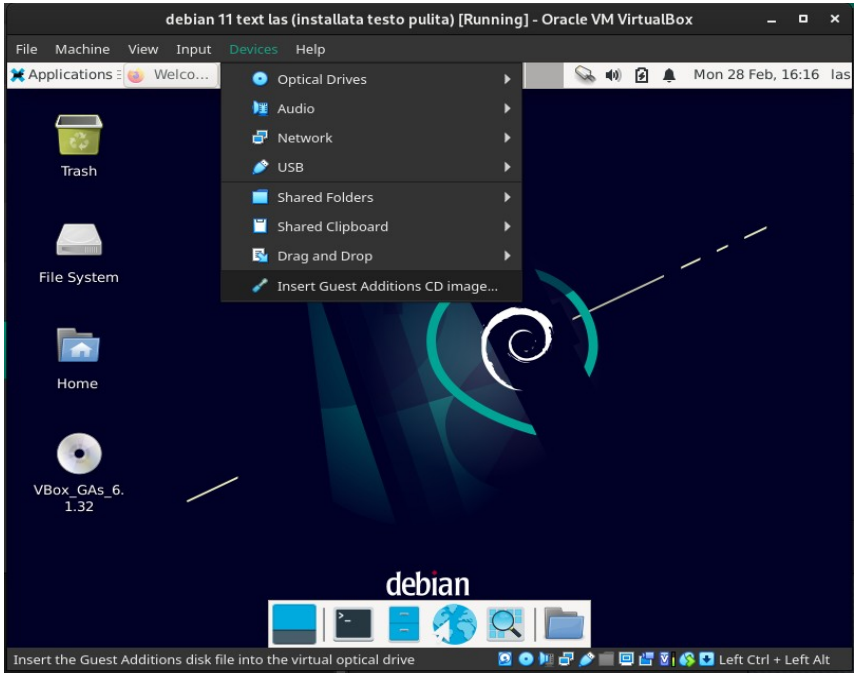
```
# <file system> <mount point>   <type>  <options>    <dump>  <pass>
/dev/sda1       /     ext3    defaults,errors=remount-ro,relatime    0 1
/dev/sda4     /home   ext3    defaults,relatime                      0 2
```

Scambio file tra tra host e guest

Attraverso la cartella condivisa (1)

- Preliminare: installare le *guest additions* nell'HOST
  - Devices → Insert guest addition CD
  - Aprire un terminale e diventare root
  - Impartire (come root):

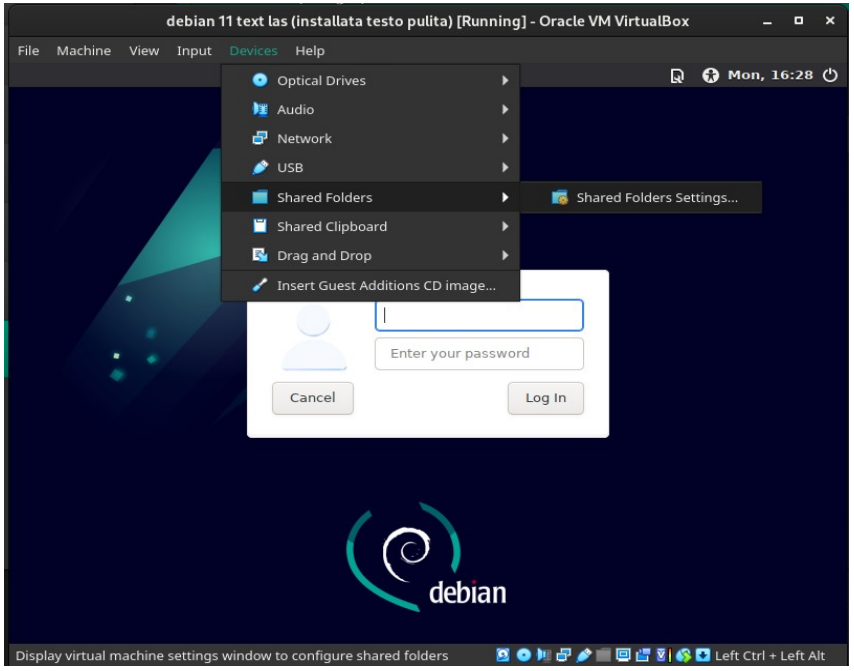
```
mount /dev/cdrom /mnt
/mnt/VBoxLinuxAdditions.run
reboot
```

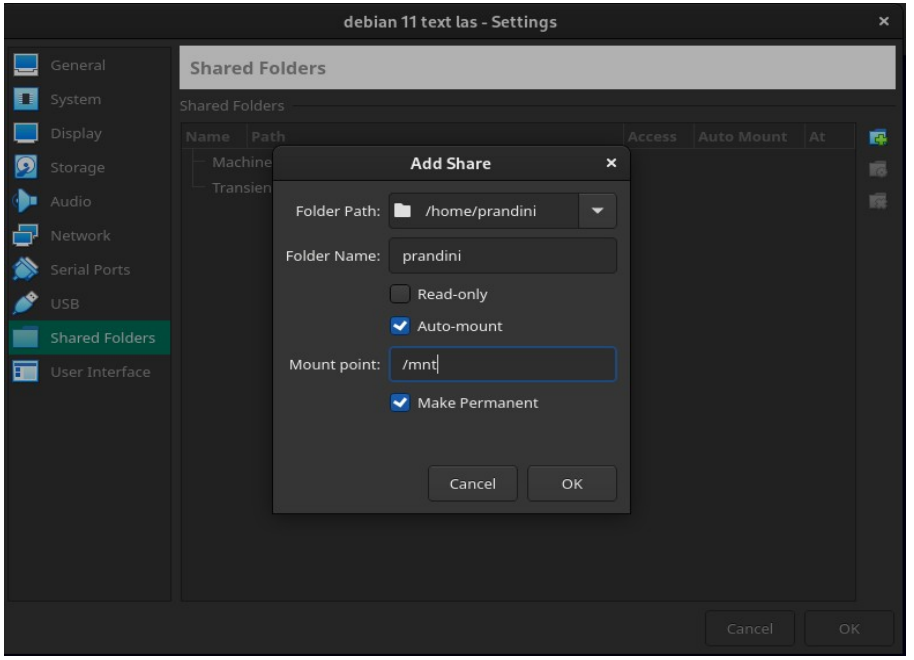


Scambio file tra tra host e guest

Attraverso la cartella condivisa (2)

- Nelle impostazioni VirtualBox → *Cartelle condivise (Shared folders)*
  - Scegliere dal menu di esplorazione la cartella desiderata dell'host
  - Spuntare "Monta automaticamente" (Auto-mount) e "Rendi permanente" (Make Permanent)
  - Scrivere `/mnt` in "Mount point"





Laboratorio

- Utilizziamo VirtualBox per creare dischi aggiuntivi
- Nel SO guest
  - Osserviamo come appaiono
  - Li partizioniamo
  - Formattiamo le partizioni
  - Eseguiamo un mount manuale
  - Creiamo file e osserviamo cosa succede smontando una partizione e rimontandola in un mount point diverso
  - Configuriamo il mount automatico

Filesystem virtuali

Esaminando un tipico sistema Linux si osservano diversi filesystem montati, che non hanno corrispondenza in alcun dispositivo fisico:

```
# mount
...
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
none on /proc/bus/usb type usbfs (rw,devgid=129,devmode=664)
```

Filesystem virtuali principali

- Tra i filesystem virtuali notiamo:

  `proc` e `sys`: permettono l'accesso diretto ai dati del kernel, quali
  - aree di memoria
  - parametri dei processi
  - parametri di configurazione dei moduli

  (vale la pena dare un'occhiata direttamente per rendersi conto di cosa è disponibile)
- `udev`: permette la generazione automatica da parte dei device drivers degli special file per l'accesso ai dispositivi
- `tmpfs` aree per la mappatura in memoria anzichè su disco di dati volatili

Utenti

- Gli utenti sono i *soggetti* di tutte le operazioni svolte sul sistema, utilizzati per determinare i permessi di accesso a qualsiasi risorsa (*oggetto*)
- Ogni utente **deve** appartenere a un gruppo
  - al login l'utente si trova a operare come membro di tale gruppo
- Ogni utente **può** appartenere a un numero arbitrario di gruppi supplementari
  - durante una sessione di lavoro, l'utente può liberamente assumere l'identità di qualsiasi gruppo del quale sia membro

useradd

- useradd è lo standard per creare nuovi utenti, ha una granularità molto fine ed è molto utile per automatizzare tale processo
- I valori predefiniti per le caratteristiche dell'utente creato sono impostati nel file /etc/login.defs
- Parametri principali
  - m crea la home del utente, usa come template i files dentro /etc/skel
  - s assegna la shell all'utente, le possibili shell sono indicate dentro il file /etc/shells, altrimenti prende il default
  - U crea un gruppo con lo stesso nome dell'utente
  - K con questo parametro è possibile specificare la UMASK=0077
  - p dopo questo parametro è possibile inserire la password utente MA come riportato nella man page è sconsigliato, molto meglio usare passwd separatamente
  - G posso assegnare l'utente all'atto della creazione ad un gruppo supplementare esempio sudo

Il db degli utenti

- Le credenziali locali sono in
  - `/etc/passwd`, world-readable, una riga per utente:
    `prandini:x:500:500:Marco Prandini:/fat/home:/bin/bash`
  - `/etc/shadow`, accessibile solo a root, linee corrispondenti a **passwd**
    `prandini:$1$/PBy29Md$kjC1F8dvHxKhnvMTWelnX/:12156:0:99999:7:::`
  - Nota: non rimuovere il segnaposto 'x' nel secondo campo di passwd, o il sistema non guarderà il file shadow e non riconoscerà la password
- L'appartenenza ai gruppi è l'unione dell'informazione presente in **/etc/passwd** riguardante il gruppo principale di ogni utente e del contenuto dei file
  - `/etc/group`, world-readable, una riga per gruppo:
    `sudo:x:27:prandini`
  - `/etc/gshadow`, accessibile solo a root, linee corrispondenti a **group**
    `sudo:*::prandini`
- Il comando **id <USER>** riporta tutte le informazioni di identità

Gestione di utenti esistenti

- Il comando **usermod** permette di modificare, coi suoi diversi parametri, tutte le caratteristiche dell'utente
  - come useradd, può essere usato solo da root
- Esiste anche una serie di comandi specifici per cambiare singole proprietà
  - possono essere invocati da root per gestire qualsiasi utente
  - possono essere invocati anche da utenti standard per agire ovviamente solo sul proprio account

chsh modifica della shell di login

chfn modifica del nome reale

passwd modifica della password

Età delle password

- Il file shadow contiene dati sulla validità temporale della password, esaminabili e modificabili con **chage**:

```
<name>:<pw>:<date>:PASS_MIN_DAYS:PASS_MAX_DAYS:PASS_WARN_AGE:INACTIVE:EXPIRE:
```

- Significato e nome del file da cui viene preso il valore di default::

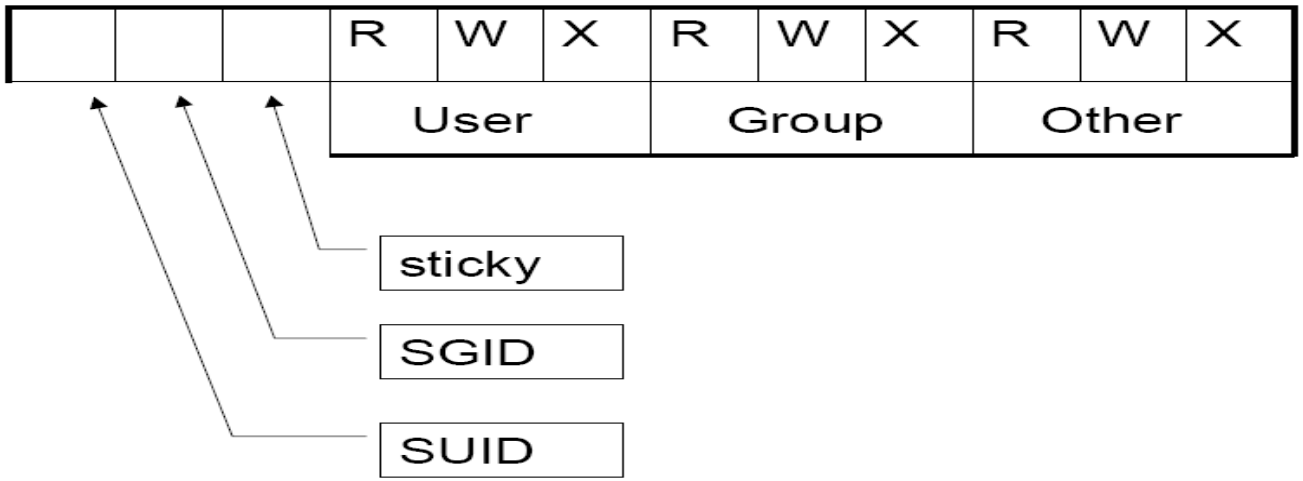
/etc/login.defs	PASS_MAX_DAYS	Maximum number of days a password is valid.
/etc/login.defs	PASS_MIN_DAYS	Minimum number of days before a user can change the password since the last change.
/etc/login.defs	PASS_WARN_AGE	Number of days when the password change reminder starts.
/etc/default/useradd	INACTIVE	Number of days after password expiration that account is disabled.
/etc/default/useradd	EXPIRE	Account expiration date in the format YYYY-MM-DD.

Altri comandi di creazione e gestione

- **adduser** è uno script in Perl per creare un nuovo utente non è lo standard in tutte le distribuzioni, è presente in Debian, Ubuntu
  - il programma chiede i dettagli interattivamente
  - utile quindi se usato in maniera estemporanea, molto poco invece se abbiamo bisogno di scriptare il processo di creazione utenti
- Per la creazione di gruppi, esistono analogamente **groupadd** e **addgroup**
- Altri comandi utili:
  - **gpasswd** modifica password e lista utenti di un gruppo
  - **getent** interroga il db utenti o gruppi
  - **last** elenca i login effettuati sul sistema
  - **lastlog** mostra la data di ultimo login di ogni utente
  - **faillog** mostra i login falliti sul sistema

Autorizzazioni su Unix Filesystem

- Ogni file (regolare, directory, link, socket, block/char special) è descritto da un i-node
- Un set di informazioni di autorizzazione, tra le altre cose, è memorizzato nell'i-node
  - (esattamente un) utente proprietario del file
  - (esattamente un) gruppo proprietario del file
  - Un set di 12 bit che rappresentano permessi standard e speciali



Significato dei bit di autorizzazione

- Leggermente diverso tra file e directory, ma in gran parte deducibile ricordando che
  - Una directory è semplicemente un file
  - Il contenuto di tale file è un database di coppie (nome, i-node)

R = read (lettura del contenuto)

Lettura di un file

Elenco dei file nella directory

W = write (modifica del contenuto)

Scrittura dentro un file

Aggiunta/cancellazione/rinomina di file in una directory

X = execute

Esegui il file come programma

Esegui il lookup dell'i-node nella

NOTA che il permesso 'W' in una directory consente a un utente di cancellare file sul contenuto dei quali non ha alcun diritto

NOTA: l'accesso a un file richiede il lookup di tutti gli i-node corrispondenti ai nomi delle directory nel path → serve il permesso 'X' per ognuna, mentre 'R' non è necessario

Assegnazione dell'ownership

- Alla creazione di un file
  - l'utente creatore è assegnato come proprietario del file
  - Il gruppo attivo dell'utente creatore è assegnato come gruppo proprietario
    - Default = gruppo predefinito, da /etc/passwd
    - L'utente può rendere attivo nella sessione un altro tra i propri gruppi con newgrp
    - Può cambiare automaticamente nelle directoy con SGID settato (vedi seguito)
- Successivamente
  - Comando chown [new_owner]:[new_group] <file>
    modifica owner e/o group owner del file
  - Comando chgrp [new_group] <file>
    modifica group owner del file
    - comunque solo tra quelli di cui l'utente è membro
  - Per entrambi l'opzione -R attiva la ricorsione su cartelle

Assegnazione dei permessi

- Alla creazione: permessi = “tutti quelli sensati” tolta la umask
  - “tutti quelli sensati” significa due cose diverse:
    - `rw-rw-rw-` (666) per i file, l’eseguibilità è un’eccezione
    - `rwxrwxrwx` (777) per le directory, la possibilità di entrarci è la regola
    - la *umask* quindi può essere unica: una maschera che toglie i permessi da non concedere
  - poiché in Linux il gruppo di default group di un utente contiene solo l’utente stesso, una umask sensata è 006 (toglie agli “other” lettura e scrittura)
    - È un settaggio utile per collaborare, crea file manipolabili da tutti i membri del gruppo, a patto che questo sia settato correttamente
  - col comando umask si può interrogare e settare interattivamente nella sessione corrente, per rendere persistente la scelta si usano i file di configurazione della shell

Assegnazione dei permessi

- Successivamente, **chmod** è usato per modificare i permessi
  - Modo numerico (base ottale):

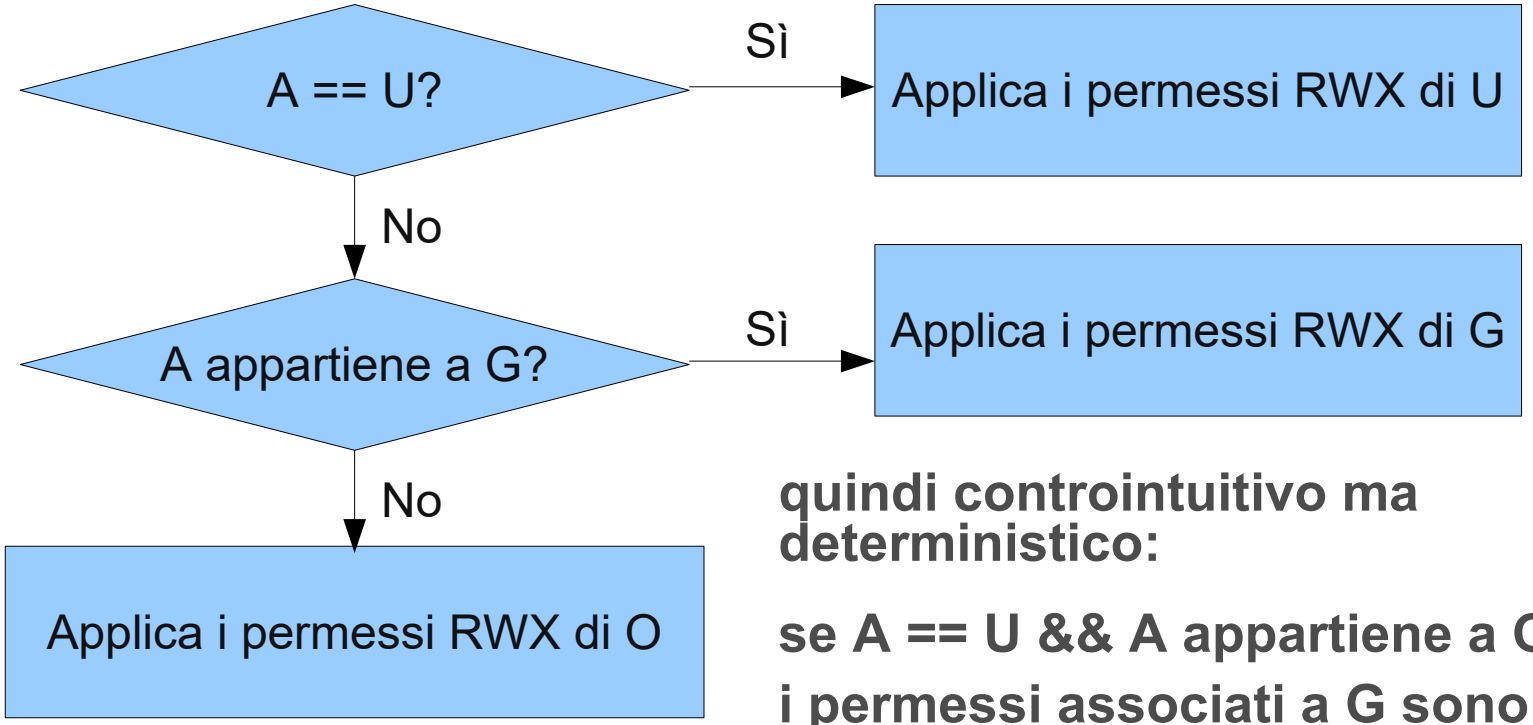
```
chmod 2770 miadirectory
2770 octal = 010 111 111 000 binary = SUID SGID STICKY rwx rwx ---

chmod 4555 miocomando
4555 octal = 100 101 101 101 binary = SUID SGID STICKY r-x r-x r-x
```

  - Modo simbolico:

```
chmod 'a=r,g-rx,u+rw' file
```

Composizione dei permessi

- **Quando un utente “A” vuole eseguire un’operazione su di un file, il sistema operativo controlla i permessi secondo questo schema:**



quindi controintuitivo ma deterministico:

se A == U && A appartiene a G

i permessi associati a G sono ignorati anche se più favorevoli

SUID e SGID

- Supponiamo che un utente U, che in in dato momento ha come gruppo attivo G, lanci un programma
- Il processo viene avviato con una quadrupla di identità:
  - real user id (ruid) = U
  - real group id (rgid) = G
  - effective user id (euid) identità assunta dal processo per operare come soggetto diverso da U
  - effective group id (egid) identità di gruppo assunta dal processo per operare come soggetto diverso da G
- Normalmente euid=ruid e egid=rgid
- Alcuni permessi speciali attribuiti a file eseguibili possono fare in modo che euid e/o egid siano diversi dai corrispondenti ruid / rgid
  - si definiscono programmi Set-User-ID o Set-Group-ID

Bit speciali / per i file

I tre bit più significativi della dozzina (11, 10, 9) configurano comportamenti speciali legati all'utente proprietario, al gruppo proprietario, e ad altri rispettivamente

- BIT 11 – SUID (Set User ID)
  - Se settato a 1 su di un programma (file eseguibile) fa sì che al lancio il sistema operativo generi un processo che esegue con l'identità dell'utente proprietario del file, invece che quella dell'utente che lo lancia
- BIT 10 – SGID (Set Group ID)
  - Come SUID, ma agisce sull'identità di gruppo del processo, prendendo quella del gruppo proprietario del file
- BIT 9 – STICKY
  - OBSOLETO, suggerisce al S.O. di tenere in cache una copia del programma

Bit speciali / per le directory

- Bit 11 per le directory non viene usato
- Bit 10 – SGID
  - Precondizioni
    - un utente appartiene (anche) al gruppo proprietario della directory
    - il bit SGID è impostato sulla directory
  - Effetto:
    - l'utente assume come gruppo attivo il gruppo proprietario della directory
    - I file creati nella directory hanno quello come gruppo proprietario
  - Vantaggi (mantenendo umask 0006)
    - nelle aree collaborative il file sono automaticamente resi leggibili e scrivibili da tutti i membri del gruppo
    - nelle aree personali i file sono comunque privati perché proprietà del gruppo principale dell'utente, che contiene solo l'utente medesimo
- Bit 9 – Temp
  - Le "directory temporanee" cioè quelle world-writable predisposte perché le applicazioni dispongano di luoghi noti dove scrivere, hanno un problema: chiunque può cancellare ogni file
  - Questo bit settato a 1 impone che nella directory i file siano cancellabili solo dai rispettivi proprietari