

Cosa è un sistema operativo?

- È un *programma* (o un insieme di programmi) che agisce come *intermediario tra l'utente e l'hardware* del computer:
 - fornisce un **ambiente di sviluppo e di esecuzione** per i programmi applicativi
 - fornisce una **visione astratta** dell'HW
 - **gestisce** efficientemente le risorse del sistema di calcolo

SO e Hardware

- SO interfaccia programmi applicativi o di sistema con le risorse HW:
 - **CPU**
 - **memoria** volatile e persistente
 - **dispositivi** di I/O
 - **connessione di rete**
 - **dispositivi di comunicazione** - ...
- SO *mappa* le risorse HW in **risorse logiche**, accessibili attraverso interfacce ben definite:
 - *processi* (CPU)
 - *file system* (dischi)
 - *memoria virtuale* (memoria)...

Componenti principali SO

- Quali sono le ***componenti*** di un SO?
 - gestione dei ***processi***
 - gestione della ***memoria centrale***
 - gestione di ***memoria secondaria e file system***
 - gestione dell'***I/O***
 - interfaccia utente/programmatore

Processo

- **Processo = programma in esecuzione**
 - *programma* è *un'entità passiva* (un insieme di byte contenente le istruzioni che dovranno essere eseguite)
- ***il processo è un'entità attiva:***
 - è *l'unità di lavoro/esecuzione* all'interno del sistema. ***Ogni attività all'interno del SO è rappresentata da un processo***

**Processo = programma +
contesto di esecuzione (PC, registri, ...)**

Gestione dei processi

- In un sistema multiprogrammato: più **processi** possono essere *simultaneamente presenti* nel sistema
- **Compito cruciale del SO**
 - *creazione/terminazione* dei processi
 - *sospensione/ripristino* dei processi
 - *sincronizzazione/comunicazione* dei processi
 - *gestione del blocco critico (deadlock)* di processi

Gestione Memoria Centrale

- HW di sistema di elaborazione è equipaggiato con ***un unico spazio di memoria*** accessibile direttamente da CPU e dispositivi
- **Compito cruciale di SO**
 - ***separare gli spazi di indirizzi*** associati ai processi
 - ***allocare/deallocare memoria*** ai processi
 - ***memoria virtuale*** - gestire ***spazi logici di indirizzi*** di dimensioni complessivamente ***superiori allo spazio fisico***
 - realizzare i collegamenti (***binding***) tra memoria logica e fisica

Gestione I/O

- Gestione dell'I/O rappresenta una parte importante di SO:
 - ***interfaccia*** tra programmi e dispositivi
 - per ogni dispositivo: ***device driver***
 - ***routine per l'interazione con un particolare dispositivo***
 - contiene ***conoscenza specifica*** sul dispositivo (ad es., routine di gestione delle interruzioni)

Gestione file system

- Ogni sistema di elaborazione dispone di uno o più dispositivi per la memorizzazione persistente delle informazioni (***memoria secondaria***)
- **Compito di SO**
 - fornire una ***visione logica uniforme della memoria secondaria*** (indipendente dal tipo e dal numero dei dispositivi):
 - realizzare il ***concetto astratto di file***, come unità di memorizzazione logica
 - fornire una struttura astratta per ***l'organizzazione*** dei file (***direttorio***)
 - Effettuare operazioni su file e direttori

Interfaccia Utente

- Ogni sistema di elaborazione dispone di uno o più SO che presenta un'interfaccia che consente
- l'interazione con l'utente
 - **interprete comandi (*shell*)**: l'interazione avviene mediante una linea di comando
 - **interfaccia grafica** (graphical user interface, ***GUI***): l'interazione avviene mediante ***interazione*** mouse-elementi grafici su desktop; di solito è organizzata a finestre

Shell

- Programma che permette di far *interagire l'utente (interfaccia testuale) con SO tramite comandi*
 - resta in attesa di un comando...
 - ... mandandolo in esecuzione alla pressione di <ENTER>
- In realtà *shell è un interprete comandi evoluto*
 - potente *linguaggio di scripting*
 - interpreta ed esegue comandi da *standard input* o da *file comandi*

Differenti Shell

- La shell non è unica, un sistema può metterne a disposizione varie
 - **Bourne shell** (standard), C shell, Korn shell, ...
 - L'implementazione della **bourne shell in Linux** è **bash** (**/bin/bash**)
 - 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
- La shell di login è quella che richiede inizialmente i dati di accesso all'utente
 - Per **ogni utente connesso** viene generato un **processo dedicato** (che esegue la shell)

Ciclo di esecuzione shell

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

Accesso al sistema: login

- Per accedere al sistema bisogna possedere una coppia ***username e password***
 - NOTA: UNIX è case-sensitive
- Il SO verifica le credenziali dell'utente e manda in esecuzione la sua ***shell di preferenza***, posizionandolo in un ***direttorio di partenza***
 - Entrambe le informazioni si trovano in `/etc/passwd`
- **Comando passwd**
 - È possibile ***cambiare la propria password*** di utente, mediante il comando `passwd`
 - Se ci si dimentica della password, bisogna chiedere all'amministratore di sistema (utente *root*)

Uscita dal sistema: logout

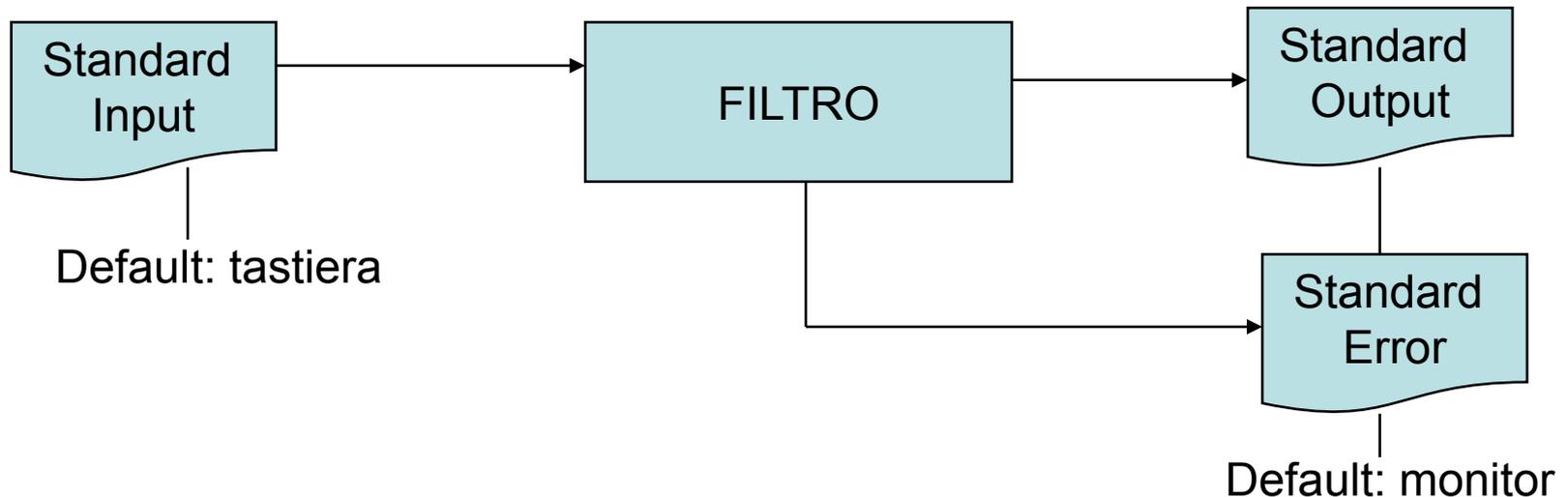
- Per uscire da una shell qualsiasi si può utilizzare il comando **exit**
- Per uscire dalla shell di login
 - **logout**
 - **CTRL+D** (che corrisponde al carattere <EOF>)
 - **CTRL+C**
- Per rientrare nel sistema bisogna effettuare un nuovo login

Comandi

- Ogni comando richiede al SO l'esecuzione di una particolare azione
- I ***comandi principali*** del sistema si trovano nella directory `/bin`
- Possibilità di ***realizzare nuovi comandi (scripting)***
- Per ogni comando, shell ***genera un processo dedicato alla sua esecuzione***

Comandi come filtri

- ***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



Manuale

- 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, digitare :q (sta per quit)***

Formato Comandi

- tipicamente: *nome -opzioni argomenti*
- esempio: `ls -l temp.txt`
- 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 | **arg1 | arg2**
 - quando un arg può essere ripetuto n volte, si aggiungono dei puntini **arg...**

File

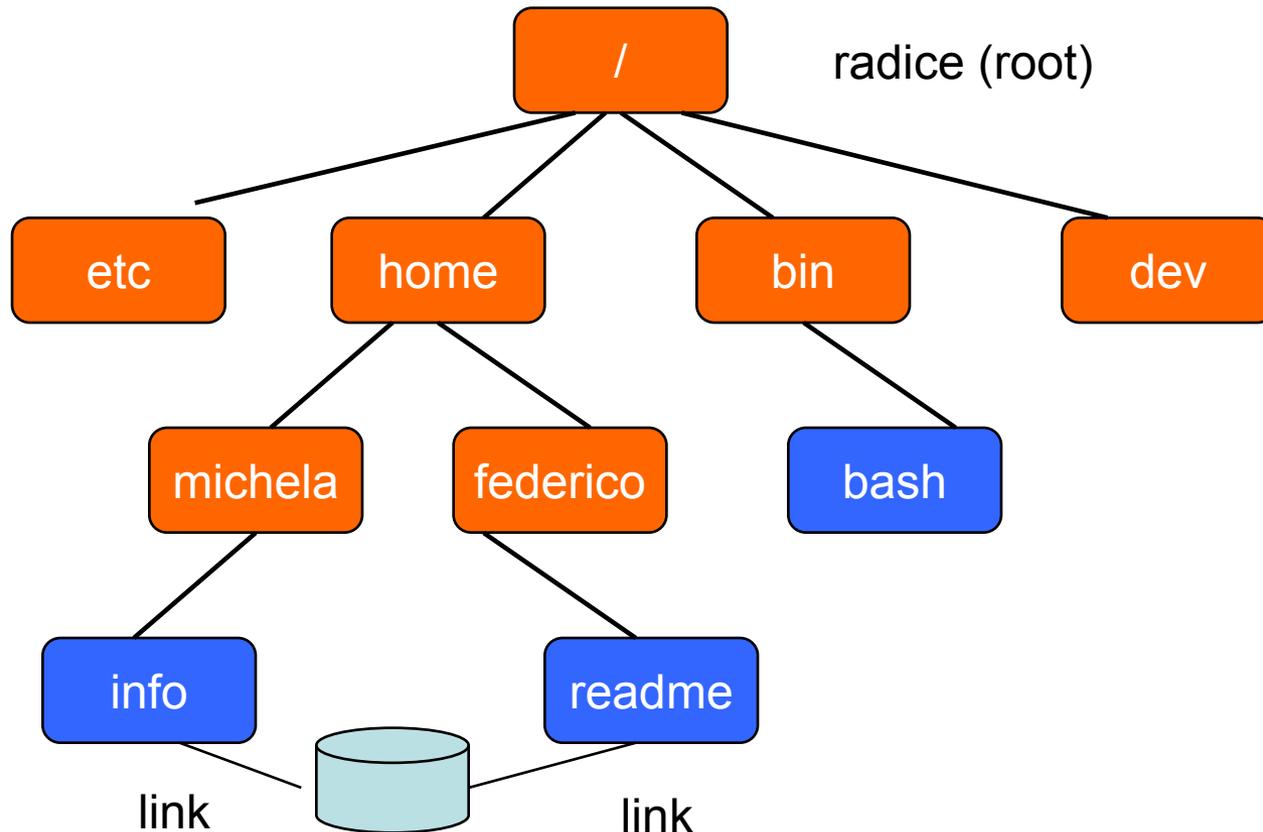
- **File** come *risorsa logica* costituita da **sequenza di bit**, a cui viene dato un nome
- **Astrazione** molto potente che consente di **trattare allo stesso modo entità fisicamente diverse** come file di testo, dischi rigidi, stampanti, direttori, tastiera, video, ...
 - **Ordinari**
 - archivi di dati, comandi, programmi sorgente, eseguibili, ...
 - **Directory**
 - gestiti direttamente solo da SO, contengono riferimenti a file
 - **Speciali**
 - dispositivi hardware, memoria centrale, hard disk, ...

Nomi 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 ***uno e un solo descrittore (i-node)*** identificato da un intero (i-number)

Directory

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



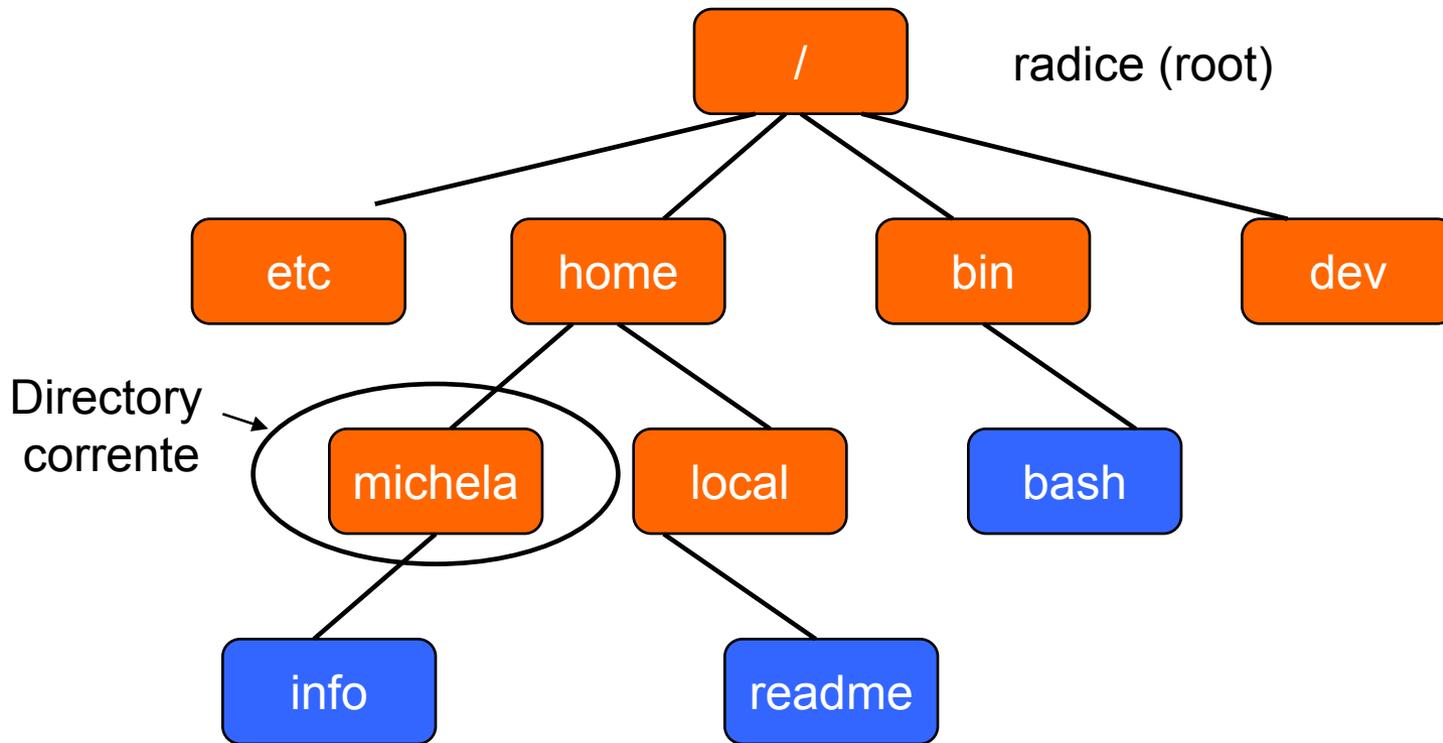
Gerarchie di Directory

- All'atto del login, l'utente può cominciare a operare all'interno di una specifica directory (**home**). In seguito è possibile cambiare directory
 - È possibile visualizzare il percorso completo attraverso il **comando** `pwd` (print working directory)
- Essendo i file organizzati in **gerarchie di directory**, SO mette a disposizione dei comandi per muoversi all'interno di essi

Nomi relativi e assoluti

- Ogni utente può specificare un file attraverso
 - **nome relativo**: è riferito alla posizione dell'utente nel file system (direttorio corrente)
 - **nome assoluto**: è riferito alla radice della gerarchia /
- Nomi particolari
 - . è il direttorio corrente (visualizzato da **pwd**)
 - .. è il direttorio 'padre'
 - ~ è la propria home utente
- Il comando **cd permette di spostarsi all'interno del file system**, utilizzando sia nomi relativi che assoluti
 - **cd** senza parametri porta alla home dell'utente

Nomi relativi e assoluti



`/home/local/readme` Percorso assoluto

`../local/readme` Percorso relativo

Link

- Le informazioni contenute in uno **stesso file** possono essere **visibili come file 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**
 - Il link **cambia i diritti? Meglio di no**
- Due tipi di link:
 - **link fisici** (si collegano le strutture del file system)
 - **link simbolici** (si collegano solo i nomi)
- comando: **ln [-s]**

Gestione file: comando ls

- consente di ***visualizzare nomi di file***
 - varie opzioni: esempio `ls -l` per avere più informazioni (non solo il nome del file)
 - possibilità di usare ***metacaratteri (wildcard)***
 - Per es. se esistono i file `f1`, `f2`, `f3`, `f4` ci si può riferire a essi scrivendo: `f*`
 - o più precisamente `f[1-4]`

Gestione file: comando ls

- Alcune opzioni
 - **l** (long format): per ogni file una linea che contiene **diritti**, **numero di link**, **proprietario** del file, **gruppo** del proprietario, **occupazione di disco** (blocchi), **data e ora** dell'ultima modifica o dell'ultimo accesso e **nome**
 - **t** (time): la lista è **ordinata per data** dell'ultima modifica
 - **u**: la lista è ordinata per data dell'ultimo accesso
 - **r** (reverse order): inverte l'ordine
 - **a** (all files): fornisce una **lista completa** (normalmente i file il cui nome comincia con il punto non vengono visualizzati)

Comandi gestione file system

- **Creazione/gestione di directory**
 - **mkdir** <nomedir> *creazione di un nuovo direttorio*
 - **rmdir** <nomedir> *cancellazione di un direttorio*
 - **cd** <nomedir> *cambio di direttorio*
 - **pwd** *stampa il direttorio corrente*
 - **ls** [<nomedir>] *visualizz. contenuto del direttorio*
- **Trattamento file**
 - **ln** <vecchionome> <nuovonome> *link*
 - **cp** <filesorgente> <filedestinazione> *copia*
 - **mv** <vecchionome> <nuovonome> *rinom. / spost.*
 - **rm** <nomefile> *cancellazione*
 - **cat** <nomefile> *visualizzazione*

Gestione processi

- Un processo utente in genere viene attivato a partire da un comando (da cui prende il nome). Ad es., dopo aver mandato in esecuzione il comando `hw`, verrà visualizzato un processo dal nome `hw`.
- *Tramite `ps` si può vedere la lista dei processi attivi*

```
pbellavis@lab3-linux:~$ ps
PID TTY STAT TIME COMMAND
4837 p2 S 0:00 -bash
6945 p2 S 0:00 sleep 5s
6948 p2 R 0:00 ps
```

Terminazione forzata processi

- È' possibile 'terminare forzatamente' un processo tramite il comando **kill** che invia un segnale ad un processo
- Ad esempio:
 - **kill -9 <PID>** provoca l'invio di un segnale
 - Esempio: **kill -9 6944**
- per conoscere il PID di un determinato processo, si può utilizzare il comando **ps**

Utenti e Gruppi

- ***Sistema multiutente*** \Rightarrow problemi di privacy e di possibili interferenze: necessità di ***proteggere/nascondere informazione***
- Concetto di gruppo (es. staff, utenti, studenti, ...): possibilità di lavorare sugli stessi documenti
- ***Ogni utente appartiene a un gruppo*** ma può far parte anche di altri a seconda delle esigenze e configurazioni
- • Comandi relativi all'identità dell'utente
 - `whoami`
 - `id`

Informazioni legate ai file

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

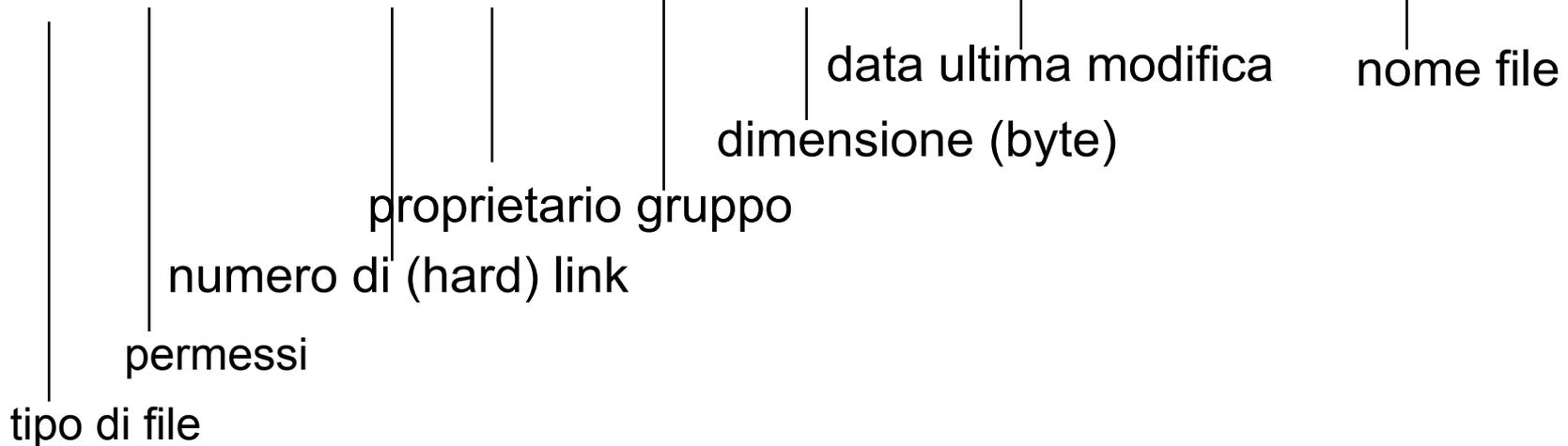
```
total 8 tot. spazio occupato (blocchi)
```

```
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
```



Protezione dei file

- Molti utenti
 - Necessità di **regolare gli accessi** alle informazioni
- Per un file, esistono 3 tipi di utilizzatori:
 - proprietario, **user**
 - gruppo del proprietario, **group**
 - tutti gli altri utenti, **others**
- Per ogni tipo di 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
 - **User-ID e Group-ID del proprietario**
 - **12 bit di protezione**

Bit di Protezione dei file

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											

- Sticky bit
 - il sistema cerca di *mantenere in memoria l'immagine del programma*, anche se non è in esecuzione

SUID e SGID

- SUID (Set User ID) (identificatore di utente effettivo)
Si applica a un file di **programma eseguibile solamente**
Se vale 1, fa sì che **l'utente** che sta eseguendo quel programma **venga considerato il proprietario di quel file (solo per la durata della esecuzione)**
 - È necessario per consentire operazioni di **lettura/scrittura su file di sistema**, che l'utente non avrebbe il diritto di leggere/ modificare.
 - Esempio: `mkdir` crea un direttorio, ma per farlo deve anche **modificare alcune aree di sistema** (file di proprietà di root), che non potrebbero essere modificate da un utente. Solo SUID lo rende possibile
- SGID bit: come SUID bit, per il gruppo

Protezioni e diritti sui file

- Per variare i bit di protezione:
 - `chmod [u g o] [+ -] [rwx] <nomefile>`
- I permessi possono essere concessi o negati dal solo **proprietario del file**
- Esempi di variazione dei bit di protezione:

- `chmod 0755 /usr/dir/file`

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		

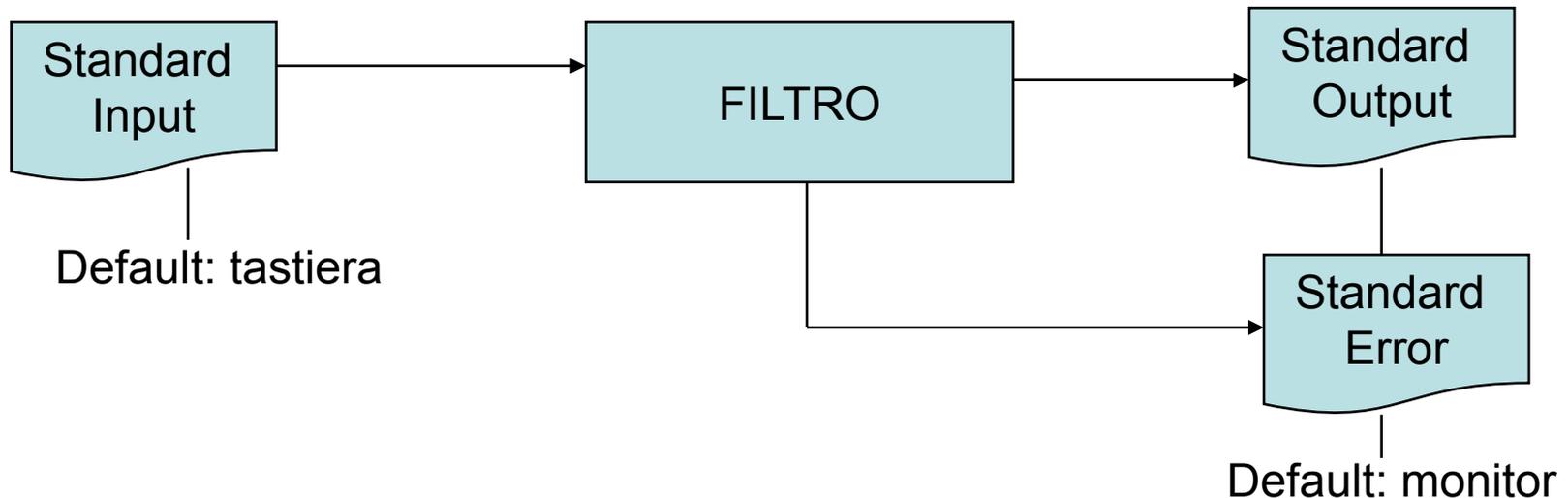
- `chmod u-w fileimportante`

- Altri comandi:
 - `chown <nomeutente> <nomefile>`
 - `chgrp <nomegruppo> <nomefile>`

Comandi e input/output

- ***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



Comandi shell linux: filtri

- **grep** <testo> [<file>...]
Ricerca di testo. Input: (lista di) file. Output: video
- **tee** <file>
Scrive l'input sia su file, sia sul canale di output
- **sort** [<file>...]
Ordina alfabeticamente le linee. Input: (lista di) file.
Output: video
- **rev** <file>
Inverte l'ordine delle linee di file. Output: video
- **cut** [-options] <file>
Selezione colonne da file. Output: video

Ridirezione

- Possibile ridirigere input e/o output di un comando facendo sì che non si legga da stdin (e/o non si scriva su stdout) ***ma da/su file***
 - ***senza cambiare il comando***
 - ***completa omogeneità tra dispositivi e file***
- Ridirezione dell'input
 - comando `< file_input`
- Ridirezione dell'output
 - comando `> file_output`
 - comando `>> file_output`

Aperto in lettura

Aperto in Scrittura
Nuovo o sovrascritto

Aperto in Scrittura
Append

Esempi

- `ls -l > file`

File conterrà il risultato di `ls -l`

- `sort < file > file2`

- Ordina il contenuto di `file` scrivendo il risultato su `file2`
- Cosa succede con `>file` ?

Piping

- ***L'output di un comando può esser diretto a diventare l'input di un altro comando (piping)***
 - In DOS: ***realizzazione con file temporanei*** (primo comando scrive sul file temporaneo, secondo legge da questo)
 - In UNIX: ***pipe come costruito parallelo*** (l'output del primo comando viene reso disponibile al secondo e consumato appena possibile, non ci sono file temporanei)
- Si realizza con il carattere speciale '|'

Esempi

- `who | wc -l`
Conta gli utenti collegati
- `ls -l | grep ^d | rev | cut -d' ' -f1 | rev`
Che cosa fa? Semplicemente mostra i nomi dei sottodirettori della directory corrente
- `ls -l` lista i file del direttorio corrente
- `grep` filtra le righe che cominciano con la lettera d (pattern `^d`,
- vedere il **man**) ovvero le directory (il primo carattere rappresenta il tipo di file)
- `rev` rovescia l'output di `grep`
- `cut` taglia la prima colonna dell'output passato da `rev`, considerando lo spazio come delimitatore (vedi **man**)
quindi, poiché `rev` ha rovesciato righe prodotte da `ls -l`, estrae il nome dei direttori 'al contrario'
- `rev` raddrizza i nomi dei direttori
- **Suggerimento:** aggiungere i comandi uno alla volta (per vedere cosa

Metacaratteri

- Shell riconosce **caratteri speciali (wild card)**
 - * una qualunque stringa di zero o più caratteri in un nome di file
 - ? un qualunque carattere in un nome di file
 - **[zfc]** un qualunque carattere, in un nome di file, compreso tra quelli nell'insieme. Anche **range** di valori: **[a-d]**
- Per esempio `ls [q-s]*` lista i file con nomi che iniziano con un carattere compreso tra q e s
 - # commento fino alla fine della linea
 - \ escape (segnala di **non interpretare** il carattere successivo come speciale)

Metacaratteri: esempi

- `ls [a-p,1-7]*[c,f,d]?`
 - elenca i file i cui nomi hanno come iniziale un carattere compreso tra 'a e 'p' oppure tra 1 e 7, e il cui penultimo carattere sia 'c', 'f', o 'd'
- `ls ***`
 - Elenca i file che contengono, in qualunque posizione, il carattere *