

Reti informatiche cod. 545II [9 CFU]

Corso di Laurea in Ingegneria Informatica

Laboratorio e Programmazione di Rete
A. A. 2020/2021

Francesco Pistolesi, PhD
Dipartimento di Ingegneria dell'Informazione
`francesco.pistolesi@unipi.it`

Francesco Pistolesi

Dipartimento di Ingegneria dell'Informazione

Università di Pisa

francesco.pistolesi@unipi.it



Ricevimento

Martedì dalle 16:30 alle 18:30, su Microsoft Teams.
Prenotare inviando una email, non usate la chat!



Materiale

Le slide saranno rese disponibili sulla mia homepage
www.iet.unipi.it/f.pistolesi, nell'area *Teaching*.

Compilare il form

https://docs.google.com/forms/d/e/1FAIpQLSdN3_-YX8bV-guP961JzNA1uG3mHBZTZebvFMLLUWZ-hJaSSA/viewform
per accedere al folder Drive con le registrazioni delle lezioni.

Esame (parte pratica)

- Sviluppo di un **progetto** relativo alla programmazione di rete in linguaggio C

(è individuale, va consegnato 72 ore prima dell'esame, usando un link creato dal docente nella pagina del corso sul portale Elearn, seguendo le istruzioni disponibili sulla stessa pagina)

- **Domande orali** su concetti teorici e pratici illustrati durante le lezioni

(per esempio: spiegare funzionalità e comandi, eseguire particolari configurazioni di rete, effettuare test, risolvere problemi in tempo reale)

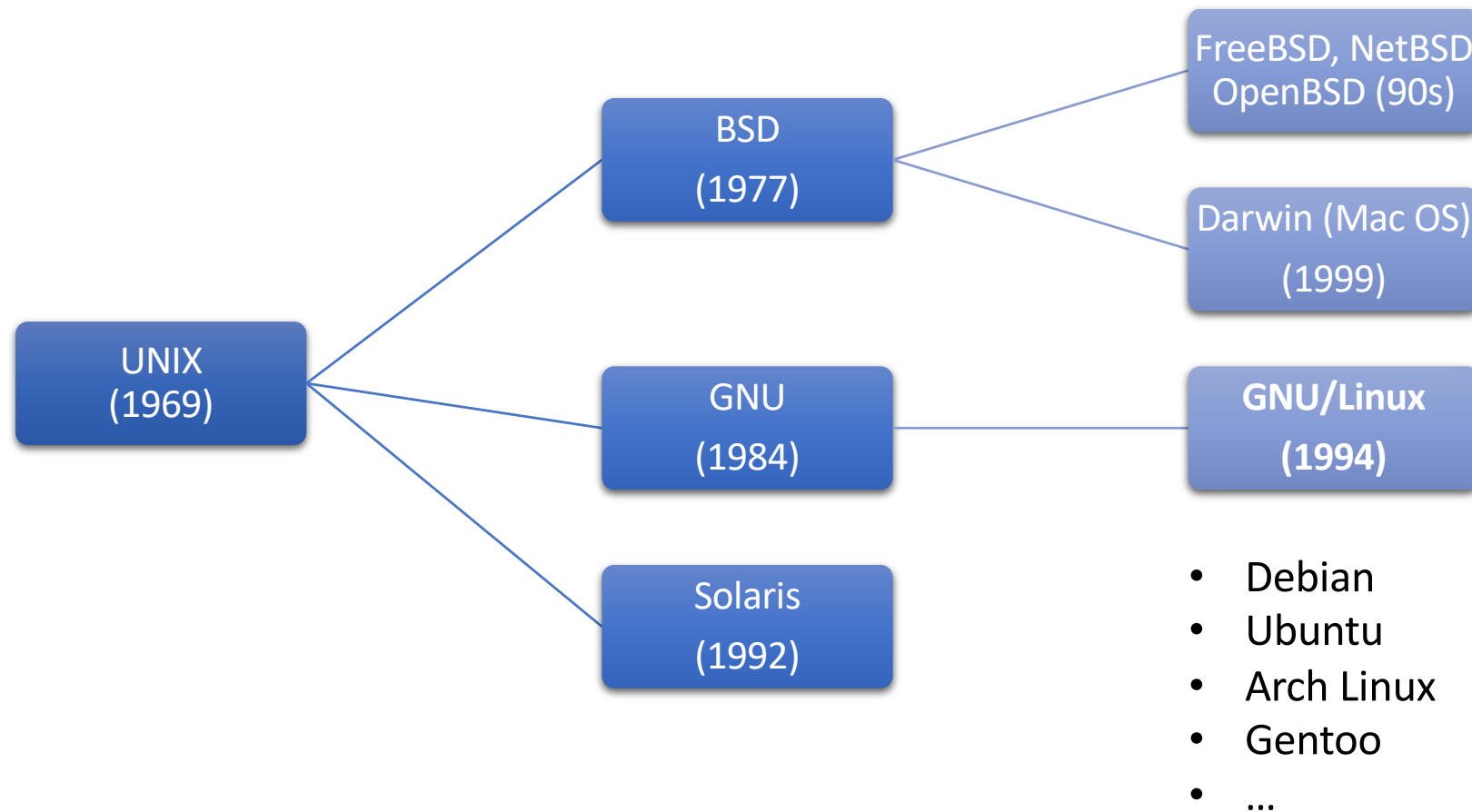
Lezione 1

Introduzione ai sistemi Linux/Unix

Programma di oggi

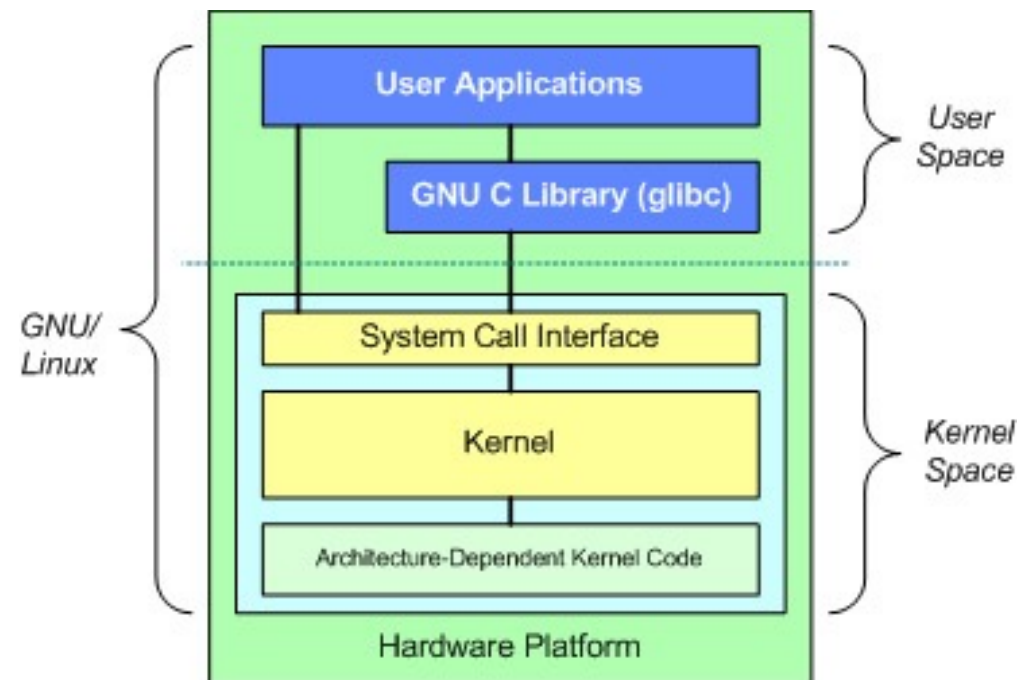
- Breve introduzione
- File system
- Interprete dei comandi (shell)
- Comandi di base

Unix e derivati



Caratteristiche di Unix

- Componenti principali:
 - Il **kernel** interagisce e controlla l'hardware
 - Le **applicazioni** sfruttano il kernel per offrire servizi e funzionalità:
 - Interprete dei comandi
 - Software di sistema (es. pannelli di controllo)
 - Programmi utente



Caratteristiche di Unix

- ***Multitasking***: più processi eseguiti «contemporaneamente»
- ***Multiutente***: più utenti possono interagire contemporaneamente, eventualmente con privilegi diversi
- ***Portabile***: grazie anche al linguaggio C, Unix è stato progettato fin dall'inizio per la *portabilità*
- ***Modulare***: programmi semplici, componibili e riusabili

Installazione di GNU/Linux

- La distribuzione di riferimento per il corso è **Debian 8** www.debian.org

Opzioni di installazione:

Installazione su **macchina fisica**

- 1) Tramite CD o chiavetta USB;
- 2) Utilizzo dell'intero disco (o su partizioni separate)

Installazione su **macchina virtuale** (es. VirtualBox)

- 1) Da zero, tramite ISO;
- 2) Importando la macchina fornita (link sul sito del corso di [Sistemi Operativi](#))

Utenti

- Utente **root**
 - Amministratore del sistema
 - Può compiere *qualsiasi* tipo di operazione
- Utenti **normali**
 - Utilizzatori del sistema
 - Hanno privilegi limitati
- Solitamente, si crea almeno un utente normale per l'uso abituale e si ricorre a root ***solo se necessario***.

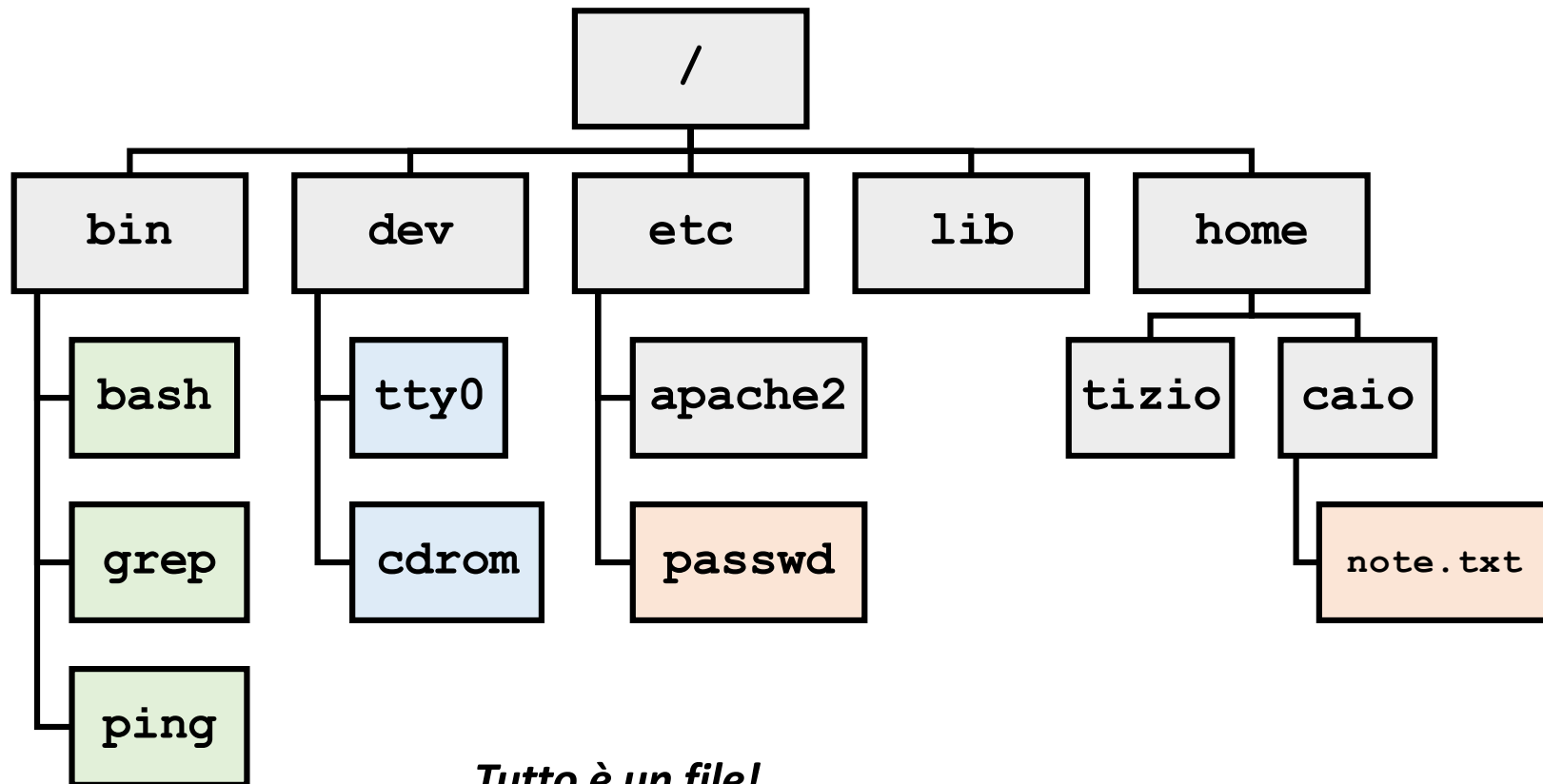
File system

Introduzione ai sistemi Unix/Linux

File system

- Tutti i dischi vengono resi accessibili (*montati*) tramite un unico file system virtuale:
 - `/` è la directory principale
 - `/home` contiene le varie *home directory* degli utenti
 - `/sbin` contiene i programmi di sistema
 - `/etc` contiene i file di configurazione
 - ...
 - `/media` rende accessibili i supporti rimovibili
 - `/media/cdrom`
 - `/media/kingston8gb`
 - ...

File system



Tutto è un file!

Ogni documento, cartella, dispositivo I/O, interfaccia di rete, stream di byte, ecc., è accessibile dall'unico file system.

File system: path

Come descrivere un percorso (*path*) del file system:

- **Percorso assoluto** – si esprime l'intero percorso partendo dalla radice:

`/home/tizio/Documents/appunti.txt`

- **Percorso relativo** – si esprime il percorso a partire dalla directory in cui si trova:

`Documents/appunti.txt`

- Caratteri speciali:
 - `~` indica la **nostra** home directory
 - `.` indica la directory **corrente**
 - `..` indica la directory **padre**

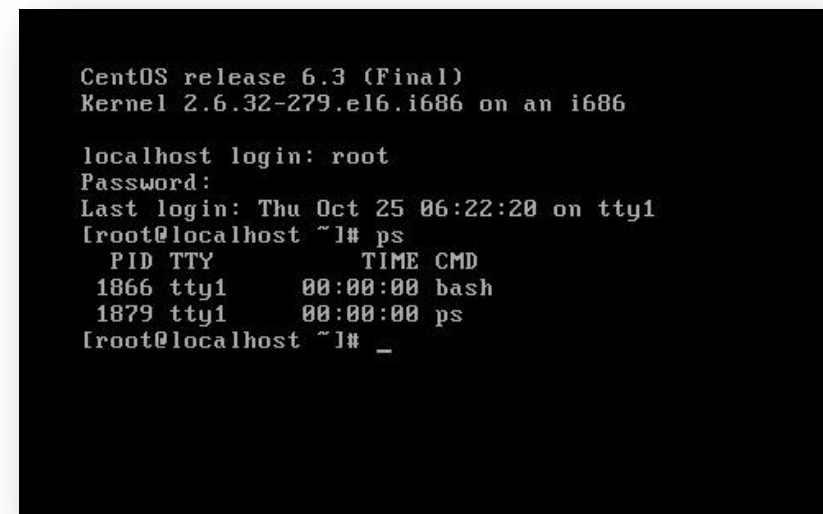
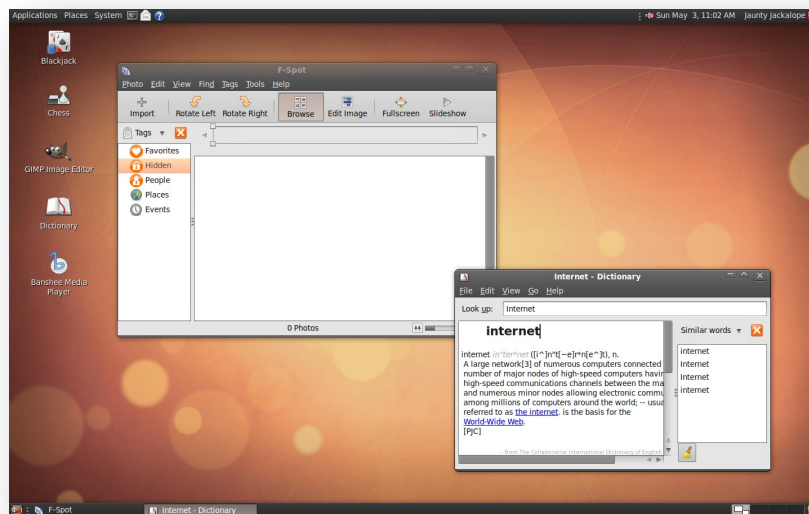
Unix è
case-sensitive!

Shell

Introduzione ai sistemi Unix/Linux

Shell

- Un interprete dei comandi, o *shell*, consente all'utente di richiedere informazioni e servizi al SO:
 - Shell **grafica** – *Graphical User Interface* (GUI)
 - Più facile da usare
 - Shell **testuale** – *Command Line Interface* (CLI)
 - Senza fronzoli, veloce e potente (sì... e anche più brutta)



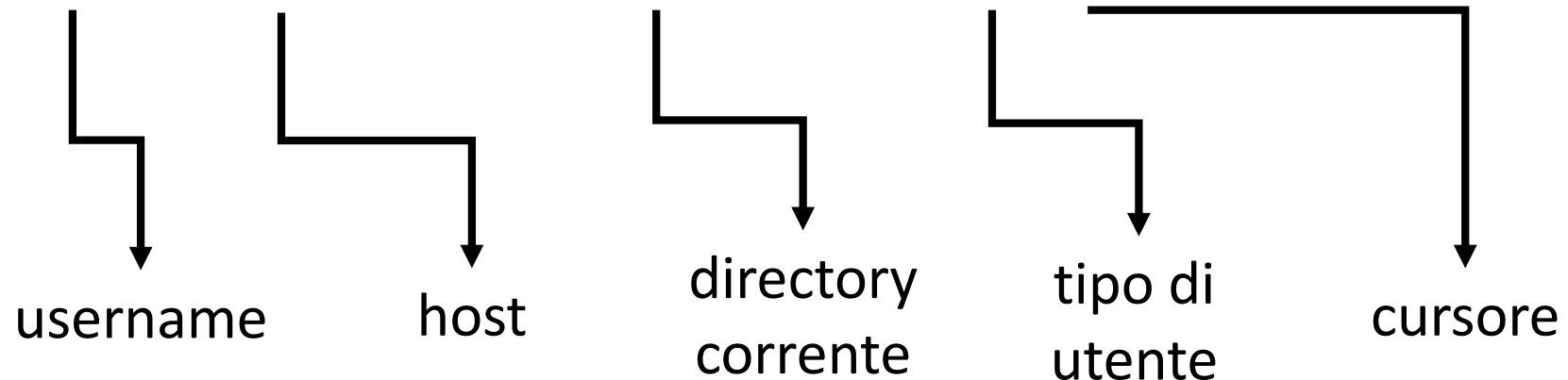
Shell

- Una shell testuale ripetutamente:
 - Mostra un *prompt*
 - Legge un comando digitato dall'utente, terminato con invio/enter/return
 - Esegue il comando
 - Se non è in grado di completarlo segnala un errore
 - Se previsto, stampa l'output del comando
- Esistono diverse shell per Unix:
 - sh, csh, tcsh, **bash**, zsh, ...
 - Differiscono per aspetto del prompt e per funzioni avanzate, ma i comandi di base sono gli stessi.

Bash

- Prompt:

```
amy@tardis:~/Documents$
```



`$`: utente normale

`#`: utente root

Accesso al sistema

- Login
 - Si accede usando username e password
- Comando **logout**
 - Per uscire dalla sessione
 - Scorciatoia: Ctrl+D
- Funzioni utili
 - Auto-completamento di comandi e directory: TAB
 - *History* dei comandi recenti: Frecce su/giù
 - Ricerca attraverso la storia Ctrl+R
 - Terminali virtuali: Ctrl+Alt+F1, F2, ...
 - In Debian/Ubuntu F7 è l'interfaccia grafica

Arresto e riavvio

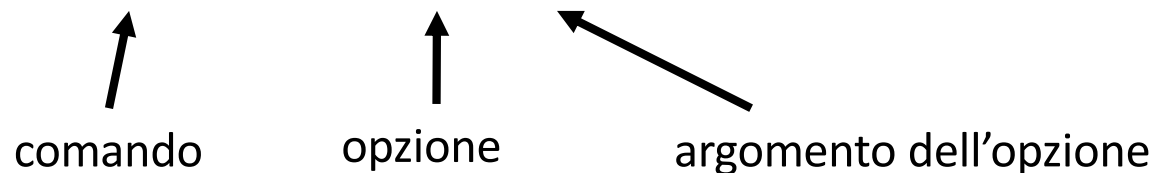
- Comando **shutdown**
 - Per arrestare/riavviare il sistema
 - Di default, solo l'utente root può invocarlo

- Arresto

shutdown -h now

- Riavvio

shutdown -r now



Comandi di base

Introduzione ai sistemi Unix/Linux

Comando **cd**

- **cd** (*change directory*) consente di passare da una directory all'altra
- Se mi trovo in **/var**, posso digitare:

\$ **cd /lib**

- Path assoluto, vado in **/lib**

\$ **cd lib**

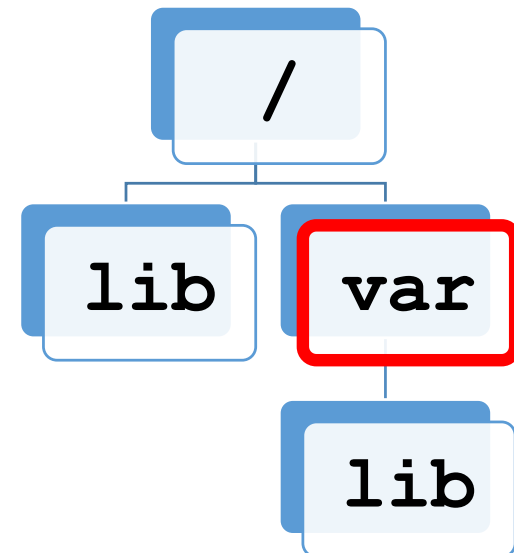
- Path relativo, vado in **/var/lib**

\$ **cd ..**

- Vado nella directory padre, cioè **/**

\$ **cd ~** (oppure solo **cd**)

- Vado nella mia home, es. **/home/amy**



Comando **pwd**

- **pwd** (*print working directory*) stampa il percorso **assoluto** della directory corrente:

```
$ pwd
```

```
/var (output)
```

```
$ cd lib
```

```
$ pwd
```

```
/var/lib (output)
```

```
$ cd ../../ (directory padre due volte)
```

```
$ pwd
```

```
/ (output)
```

```
$
```

Comando **ls**

- **ls** (*list*) serve per elencare il contenuto della directory specificata
 - Se non si specifica nulla, elenca la directory corrente
- Si possono usare percorsi assoluti o relativi
- Si possono specificare più percorsi
 - `$ ls /etc /var`
- Spesso file e cartelle sono di colori diversi

Comando **ls**

- Opzione **-l** (*long*)
 - Mostra dettagli (permessi, proprietario, dimensioni, data di ultima modifica)
\$ ls -l
- Opzione **-a** (*all*)
 - Mostra anche i file nascosti (cioè il cui nome inizia con **.**)
\$ ls -a
- Le opzioni sono cumulabili
\$ ls -a -l oppure **\$ ls -al**

Metacaratteri (*wildcards*)

- Si usano per indicare insiemi di file o cartelle
 - ***** sostituisce zero o più caratteri
 - **?** sostituisce un singolo carattere
 - **[a,b,c]** oppure **[a-z]** sostituisce un carattere nell'insieme specificato (anche con cifre)

```
$ ls
```

```
aa.c  abc.c  a.c  a.h  axc.c
```

```
$ ls *.c
```

```
aa.c  abc.c  a.c  axc.c
```

Metacaratteri (*wildcards*)

```
$ ls a*.c
```

```
aa.c  abc.c  a.c  axc.c
```

```
$ ls ?.*
```

```
a.c  a.h
```

```
$ ls a??.c
```

```
abc.c  axc.c
```

```
$ ls a[b-t]c.c
```

```
abc.c
```

```
$ ls a[4,f,x]c.c
```

```
axc.c
```

Comando **man**

- Non sapete cosa fa un comando o come si usa?
`$ man nome_comando`
- Il manuale contiene la descrizione esaustiva del comando, la sintassi, le opzioni, i messaggi di errore
- È diviso in sezioni (provate `$ man man`)
- **Non è solo** per i comandi (sezione 1)
 - Funzioni del kernel (2)
 - Funzioni delle librerie C (3)
 - File di configurazione (5)
 - ...
- Serve specificare la sezione se ci sono ambiguità:
`$ man printf` va al comando
`$ man 3 printf` va alla funzione C

Cercare nel manuale

- Comando **whatis**
 - Serve per visualizzare la descrizione breve di una pagina del manuale. Indica anche le ambiguità e le sezioni giuste.
- Comando **apropos**
 - Serve per ricercare una parola in nomi e descrizioni.
- **whatis** si usa per sapere velocemente cosa fa un comando, **apropos** per sapere che comandi ho a disposizione per fare qualcosa
 - Es. **whatis unzip** e **apropos unzip**

Comandi su file e directory

\$ mkdir nome_dir

- Crea una directory

\$ rmdir nome_dir

- Rimuove una directory, **solo se vuota**

\$ cp src dst

- Copia un file o una directory **\$ cp src1 src2 ... dst_dir**
- Copia più file o directory in un'unica directory

\$ mv src dst

- *Rinomina* un file o una directory

\$ mv src1 src2 ... dst_dir

- *Sposta* più file o directory in un'unica directory

Comandi su file e directory

\$ touch nome_file

- Aggiorna il *timestamp* di accesso e modifica di un file
- Se il file non esiste, viene creato

\$ cat file1 file2 ...

- Concatena il contenuto di due file e li stampa nello *standard output*
- Può essere utile per visualizzare velocemente file brevi

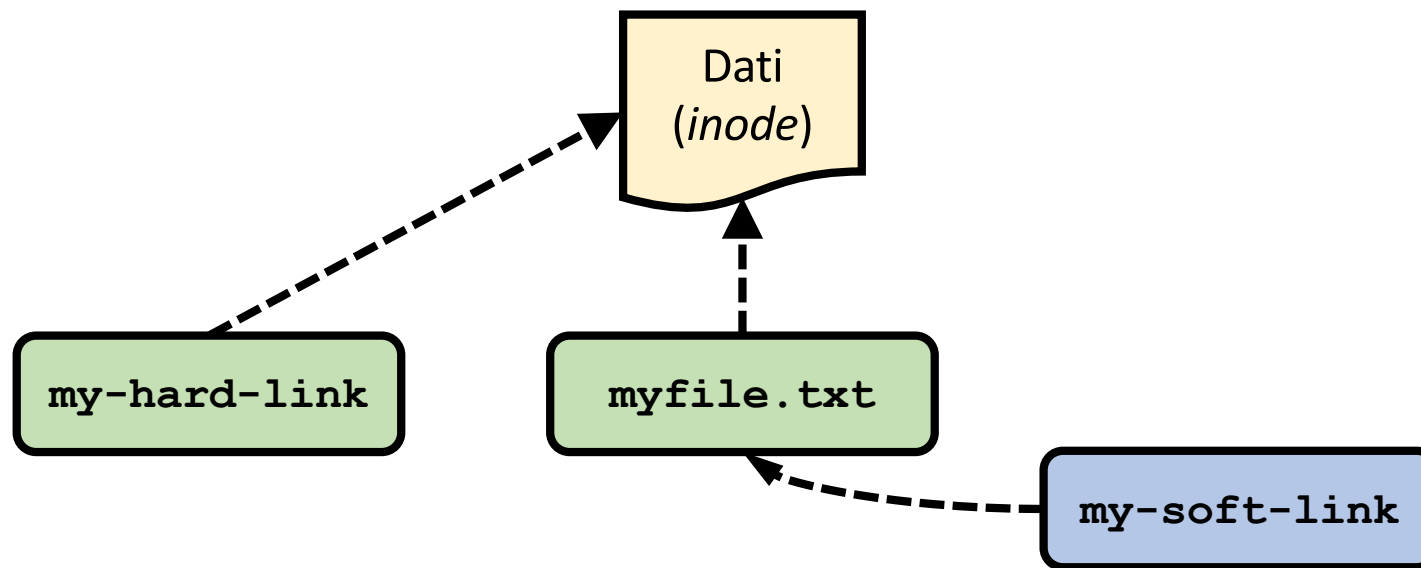
\$ rm file1 file2 ...

- Rimuove file o directory
- In mancanza di opzioni, le cartelle non vengono rimosse
- Per rimuovere una cartella e *tutto il suo contenuto*, usare **-r**

Hard link e soft link

`$ ln target nome_link`

- Crea *hard link* a file o directory
- Per creare *symbolic link* (o *soft link*) si usa `-r`

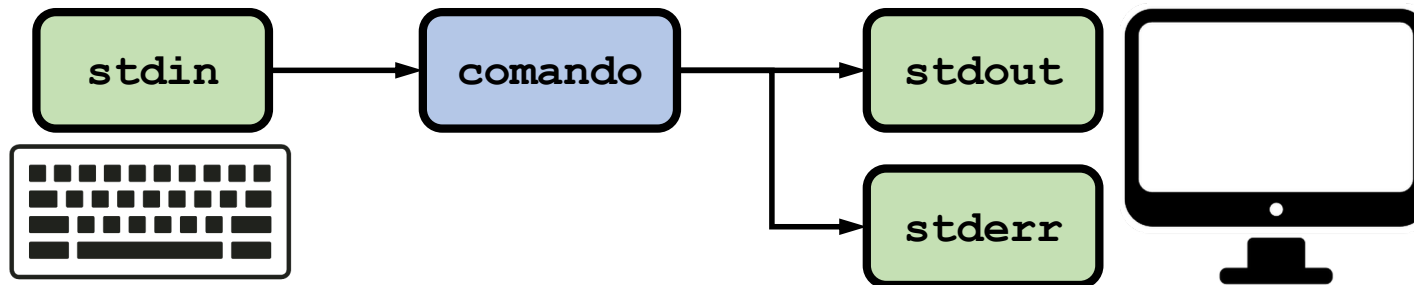


Lettura di file

- Comando **less**
 - Per visualizzare un file «un po' alla volta» e interattivamente
 - Versione migliorata di **more**
- Comandi **head/tail**
 - Per visualizzare la prima/ultima parte di uno o più file
 - Si può specificare il numero di byte da mostrare con **-c** o il numero di righe con **-n**
 - Di default, 10 righe

Redirezione I/O

- Gli input e output da riga di comando sono rappresentati come tre «file»:
 - **stdin** – input da tastiera
 - **stdout** – output su schermo
 - **stderr** – messaggi di errore su schermo



- È possibile deviare l'output di un comando verso un file oppure acquisire l'input da un file

Redirezione dell'output

- **>** invia lo stdout a un file
 - Se il file non esiste viene creato
 - Se il file esiste viene sovrascritto

```
$ ls -l > filelist.txt
```
- **2>** come sopra, per lo stderr
- **&>** come sopra, per entrambi
- **>>**, **2>>** e **&>>** come sopra, ma scrivono in *append* invece di sovrascrivere
- Si possono inviare i due output su file diversi
 - **\$ comando > out.txt 2> errors.txt**

Redirezione dell'input

- < recupera l'input da un file

```
$ sort < list.txt
```

- Si può usare in combinazione con >

```
$ sort < list.txt > sortedlist.txt
```

Pipeline

- `|` (*pipe*) collega l'output di un comando all'input del successivo

```
$ ls -l mydir | less
```

- Si può usare più volte e in combinazione con le altre redirezioni

```
$ cat *.txt | sort | uniq > result-file
```

su e sudo

- **su** (*switch user*) serve per accedere al terminale di un altro utente
 - Se non specificato, si accede al terminale di root
 - Viene chiesta la password dell'*utente con cui si vuole accedere*
- **sudo nome_comando** serve per lanciare un comando come un altro utente
 - Se non specificato, si usa l'utente root
 - Viene chiesta la password dell'*utente corrente*
 - L'utente deve fare parte nel gruppo **sudoers**

Esercizio 1

1. Aprite un terminale virtuale ed eseguite l'accesso
 - Potete aprire anche un secondo terminale da usare per **man**
2. Create la directory **Esercitazione1**
3. Create, *senza usare un editor*, un file **esercitazione.txt** all'interno di **Esercitazione1** che contenga la parola **Esercizio**
 - Per stampare parole usate **echo parola**
4. Visualizzate il contenuto del file **esercitazione.txt** usando il comando **less** (Passate a **less** prima il path relativo e poi il path assoluto del file)
5. Spostatevi in **Esercitazione1** e subito dopo usate il comando per tornare nella vostra home

Esercizio 2

1. Visualizzate il percorso della directory corrente
2. Spostatevi in **Esercitazione1** e create 3 file **f1.txt**, **f2.txt**, **f3.txt** contenenti rispettivamente le parole **Uno**, **Due**, e **Tre**
3. Con *un solo comando* create il file **f_tot.txt** partendo da **f1.txt**, **f2.txt**, **f3.txt** fatto come segue, e visualizzatene il contenuto:
 - **Uno (a capo) Due (a capo) Tre**
4. Cancellate i file **f_tot.txt**, **f1.txt**, **f2.txt**, **f3.txt**
5. Adesso create il file **fcitta.txt** fatto come segue:
 - **Milano (a capo) Perugia (a capo) Asti**
6. Visualizzate il contenuto di **fcitta.txt** in ordine alfabetico
7. Salvate il contenuto di **fcitta.txt** ordinato in un file **fcittaord.txt**

Esercizio 3

1. Usando la funzione di autocompletamento della shell passate **fcittaord.txt** al comando **less**. Fino a che punto riesce ad aiutarvi?
2. Create un file **fcitta.c** e due cartelle **Testi** e **Sorgenti**
3. Usando i metacaratteri copiate in **Testi** tutti i file **.txt** ed in **Sorgenti** i file **.c**
4. Cancellate tutti i file di testo della directory **Esercitazione**
5. Create 3 file chiamandoli **fa.txt**, **fb.txt**, **fc.txt**
6. Usate un'espressione che permetta di spostare solo **fa.txt** ed **fc.txt** e non **fb.txt** nella cartella **Testi**
7. Eliminate **fc.txt**

Esercizio 4

1. Cancellate i file della cartella **Sorgenti**
2. Usando **rmdir** eliminate le cartelle **Testi** e **Sorgenti**
 1. Ci riuscite?
3. Create una cartella **sotto** e, dentro **sotto**, una cartella **sotto1**. Usate il manuale per trovare l'opzione di **rmdir** che permette di cancellare anche **sotto** se cancellate **sotto1**
4. Create una cartella **origine** e dentro origine create la cartella **sotto_origine** ed il file **qwerty.txt**
5. Create la directory **destinazione** e copiate al suo interno *il contenuto* di **origine**. Se usate **cp** senza opzioni cosa succede? Come dovete fare?
6. Adesso copiate non solo il contenuto ma tutta la cartella **origine** in **destinazione**

Esercizio 5

1. Visualizzate il contenuto di **destinazione**
2. Adesso usate l'opzione di **ls** che visualizza anche i permessi
3. All'interno di **destinazione** create il file **.youcantseeme**
4. Visualizzatelo con **ls**
5. Salvate l'output di **ls /etc** in un file **ls_output.txt**
6. Visualizzate
 - Solo la parte iniziale del file
 - Solo la parte finale
 - Solo la prima riga
 - Solo le ultime 2 righe
7. Con *un solo comando* salvate sul file **terza.txt** solo la terza riga del file