Introduzione a Unix/Bash

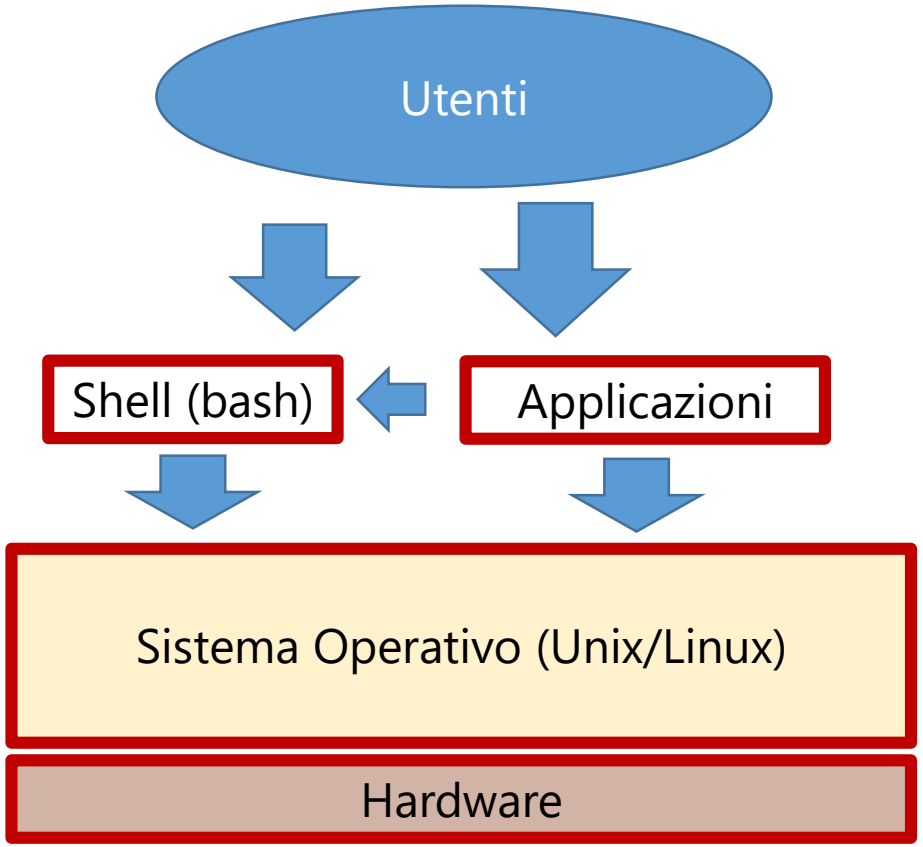
Insieme minimo di strumenti per interagire con

la shell Bash, compilare ed eseguire

Sistema operativo e shell

Il **sistema operativo** è un programma che

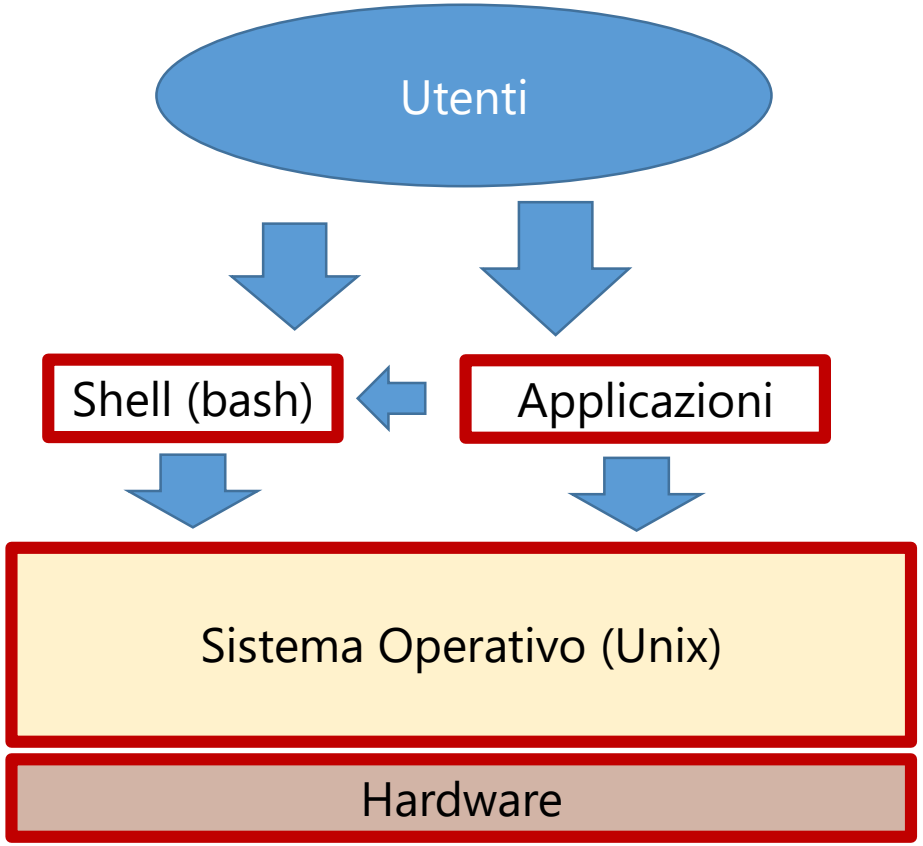
- Fa da intermediario fra le applicazioni e l'hardware
- Gestisce le situazioni potenzialmente pericolose (ad esempio l'interazione con le unità di ingresso/uscita, i dati privati degi utenti)
- Si occupa di semplificare l'accesso alle risorse (file, i/o) e di usare efficacemente le risorse hadware (processore, momoria) in modo trasparente all'utilizzatore
- Nel corso faremo riferimento al SO Unix/Linux



Sistema operativo e shell

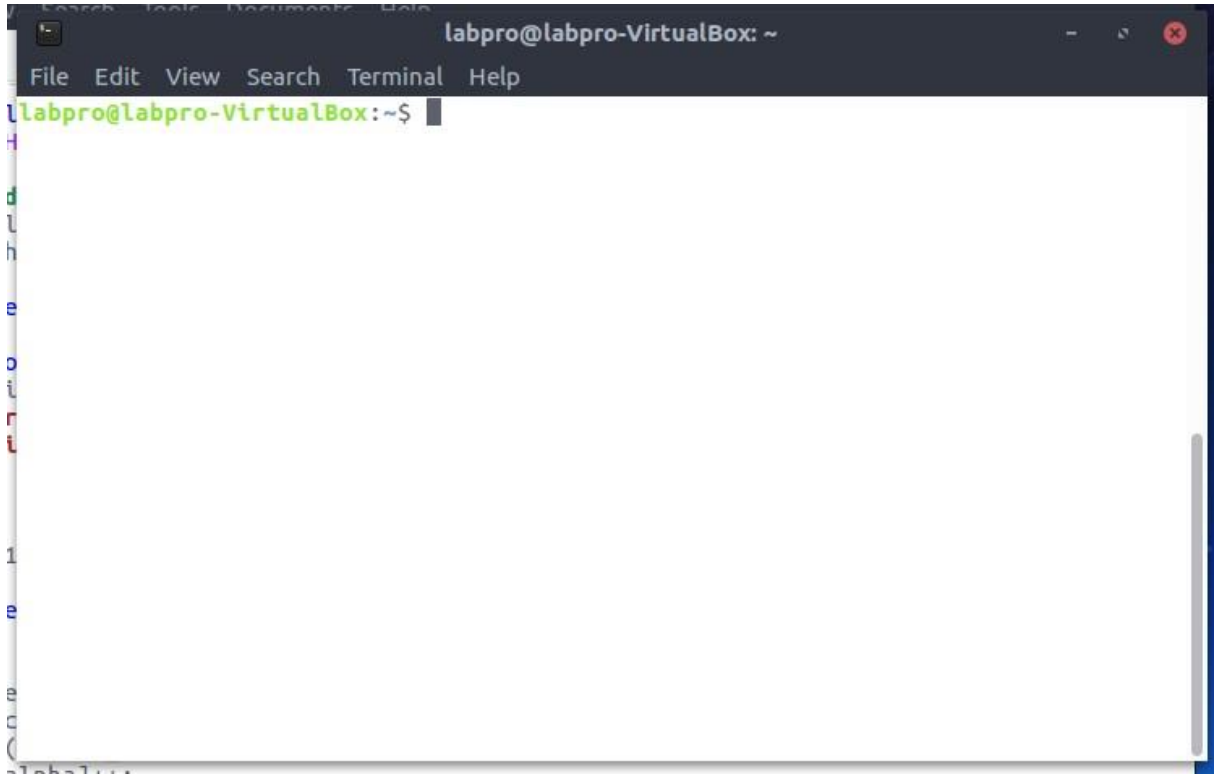
La shell

- è un programma utilizzabile direttamente dall'utente
- Permette di richiedere operazioni direttamente al Sistema Operativo (ad esempio accedere ai file, compilare programmi, eseguirli, attivare applicazioni)
- Le applicazioni che usiamo di solito posso norichiedere oprazione al SO attraverso la shell o direttamente con funzioni speciali (chiamate di sistema) (non le vedremo)
- Noi vedremo come usare la Bash (la shell più usata nel modo unix/linux)



Come è fatta una shell

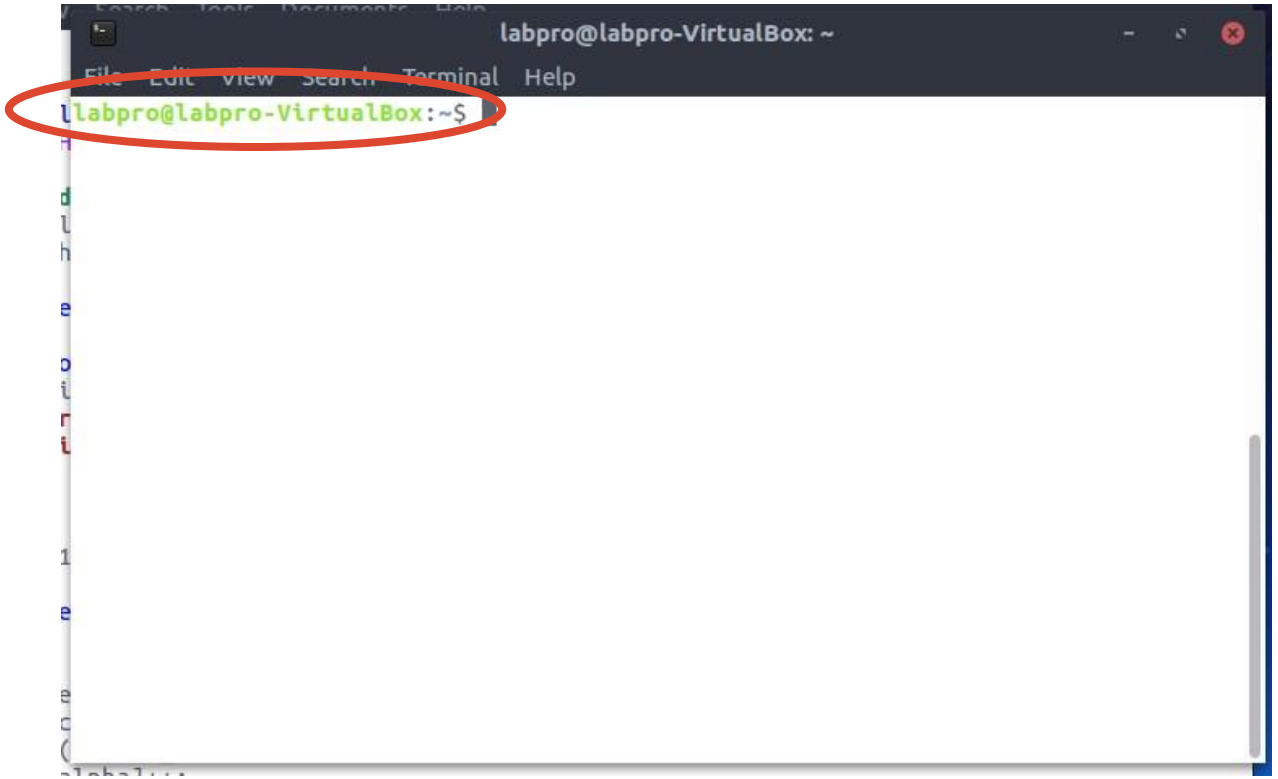
Programma testuale



Come è fatta una shell

Programma testuale

- La shell visulizza del testo (**prompt**)) e si mette in attesa di un comando da eseguire

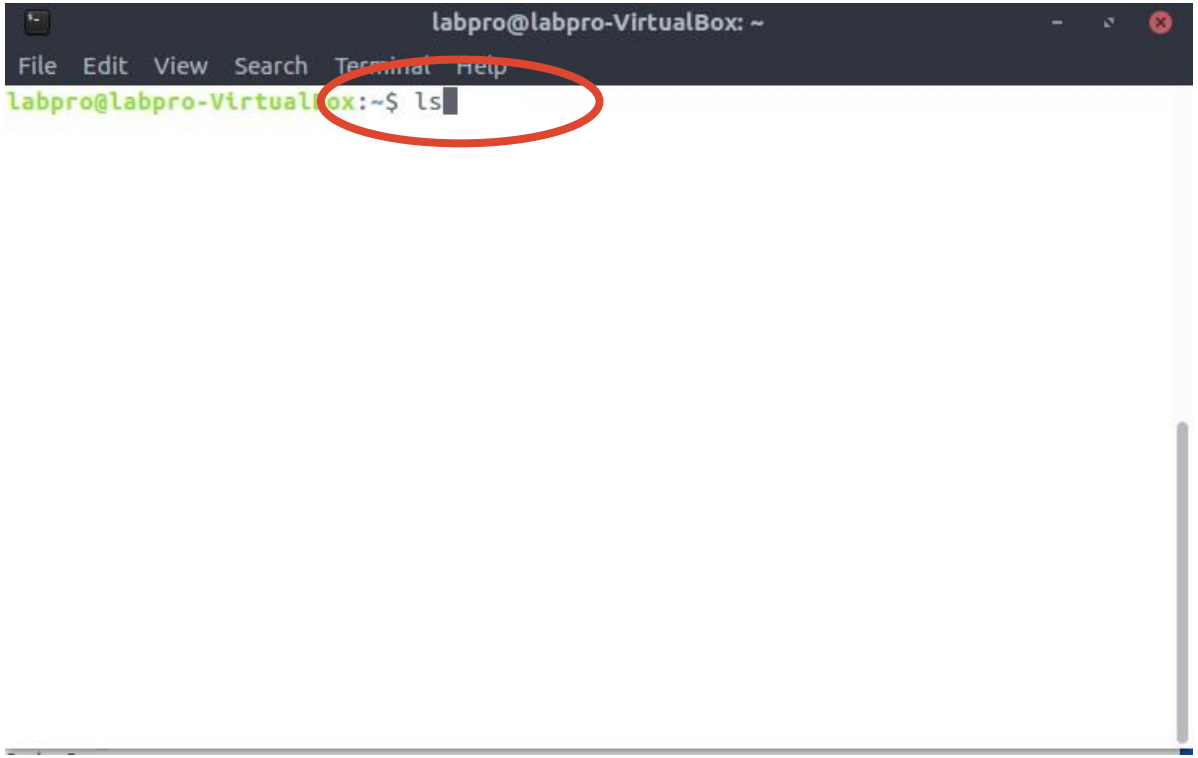


Come è fatta una shell

Programma testuale

- La shell visualizza del testo (**prompt**)) e si mette in attesa di un comando da eseguire

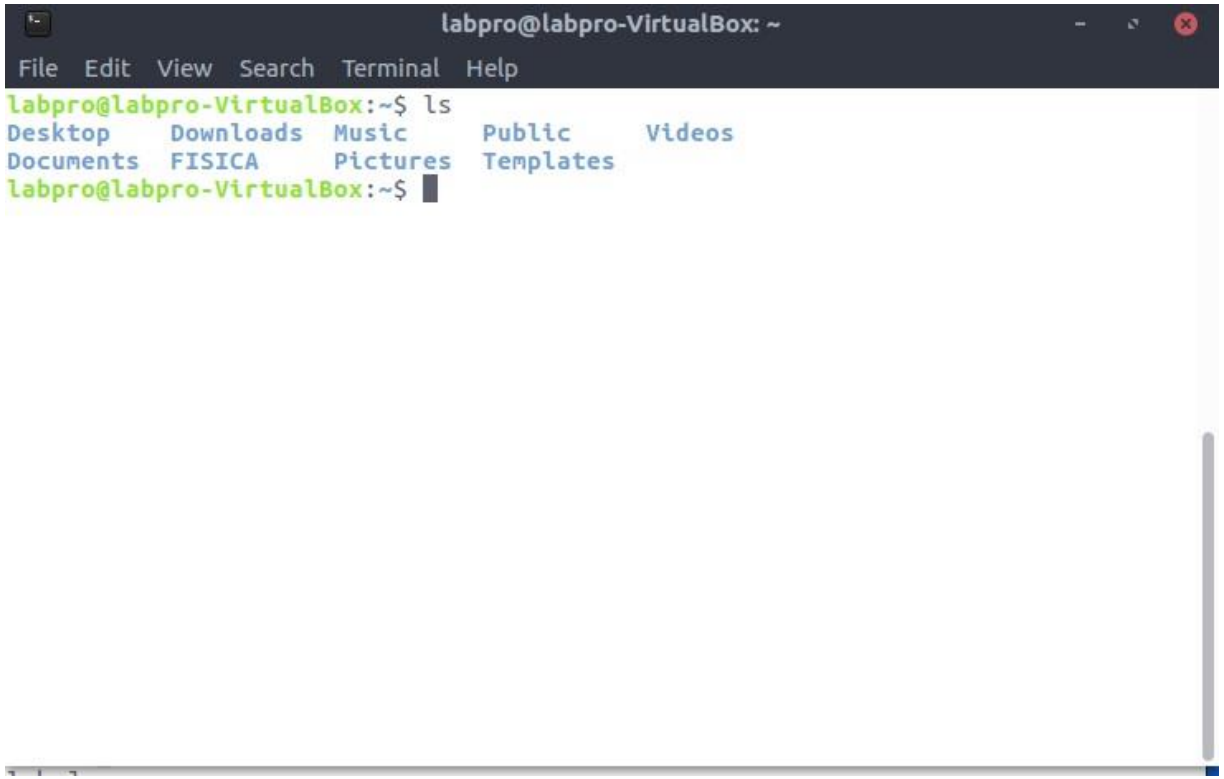
- L'utente inserisce un comando testuale (da tastiera) seguito da newline



Come è fatta una shell

Programma testuale

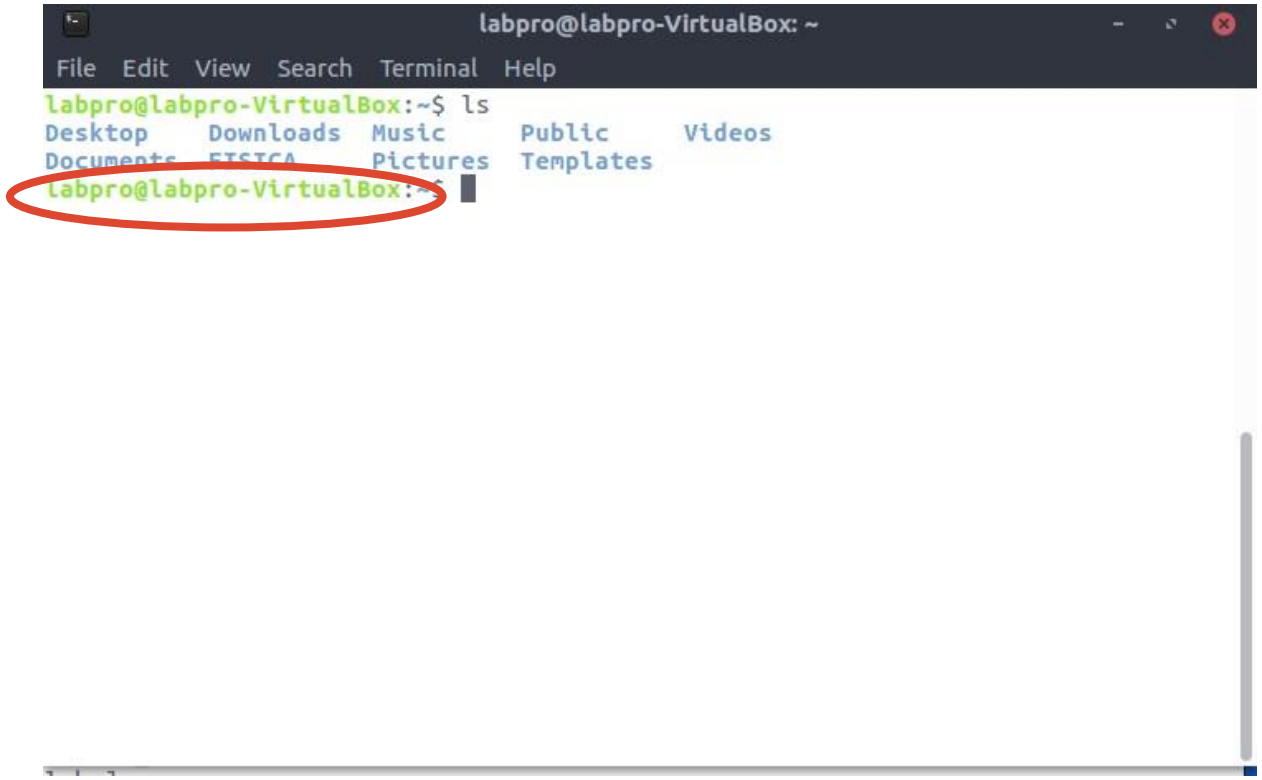
- La shell visualizza del testo (**prompt**)) e si mette in attesa di un comando da eseguire
- L'utente inserisce un comando testuale (da tastiera) seguito da newline
- La shell elabora il comando, lo esegue e visualizza la risposta



Come è fatta una shell

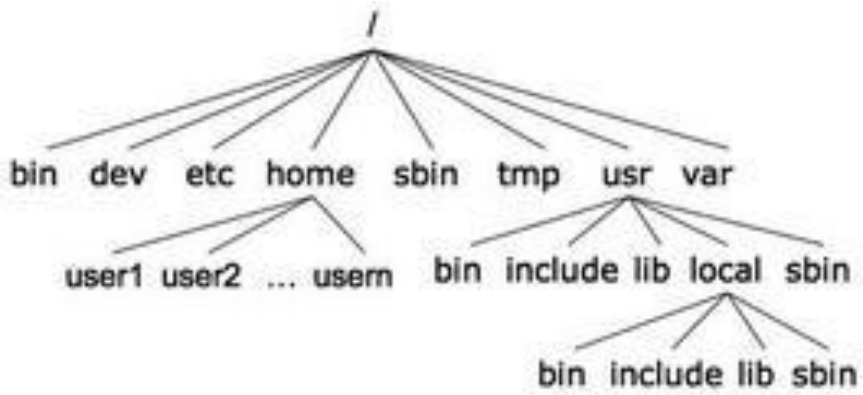
Programma testuale

- La shell visualizza del testo (**prompt**) e si mette in attesa di un comando da eseguire
- L'utente inserisce un comando testuale (da tastiera) seguito da newline
- La shell elabora il comando, lo esegue e visualizza la risposta
- Dopo l'esecuzione la shell visualizza un nuovo prompt e si mette in attesa del prossimo comando



File system di Unix

Il **file system** è la parte del sistema operativo che si occupa della memorizzazione persistente delle informazioni

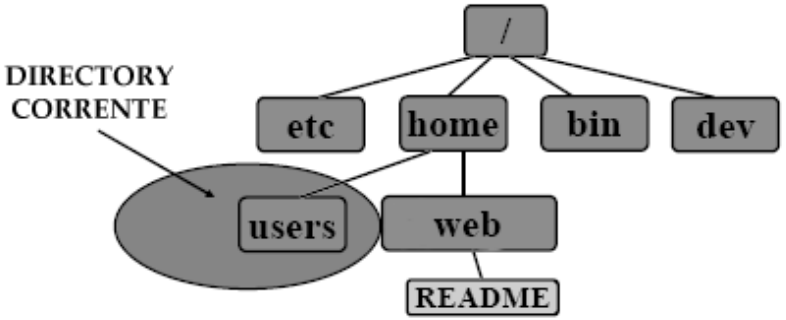


- **Struttura gerarchica:** basata su cartelle (directory) annidate
- **/ :** directory root (contiene tutto il file system)
- **/home** : directory che contiene i file dei singoli utenti
- Ogni file è caratterizzato da un nome e da permessi diaccesso (**r** - lettura , **w** - scrittura ed **x** - esecuzione) per i diversi utenti del sistema,
- Ogni file/cartella sono individuati da un percorso univoco nella gerarchia (**path name**)

Path name assoluto e relativo

Il **path name** identifica un file univocamente nel file system

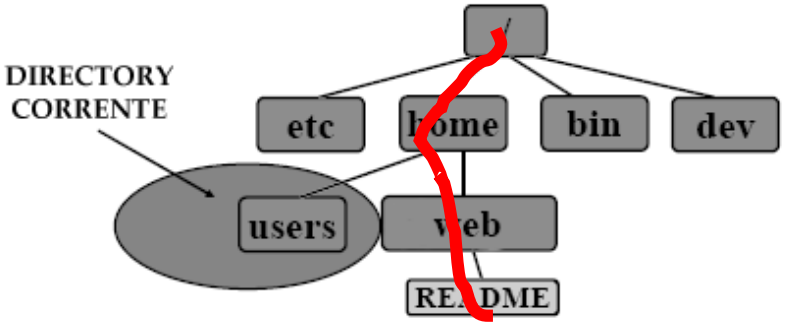
- **Path name assoluto :** percorso discendente dalla cartella root al file stesso



Path name assoluto e relativo

Il **path name** identifica un file univocamente nel file system

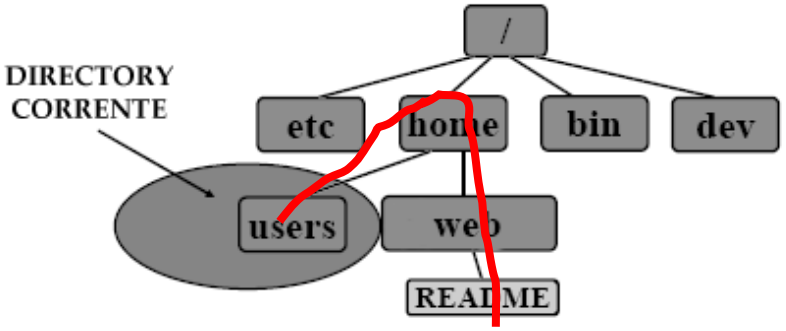
- **Path name assoluto :** percorso discendente dalla cartella root al file stesso
  - *Esempio*: path name assoluto di README (in rosso) si denota in modo testuale come **`/home/web/README`**



Path name assoluto e relativo

Il **path name** identifica un file univocamente nel file system

- **Path name assoluto :** percorso discendente dalla cartella root al file stesso
  - *Esempio*: path name assoluto di README (in rosso) si denota in modo testuale come **`/home/web/README`**
- **Path name relativo :** percorso ascendente e discendente da una cartella fissata (directory corrente o PWD
  - Utilizza due simboli speciali : **punto (.)** per indicare la directory corrente e **doppio punto (..)** per indicare la directory che contiene la directory corrente (detta directory padre)
  - *Esempio*: path name relativo di README (in rosso) si denota in modo testuale come **`./../web/README`**



Comandi Unix

Comandi utili per interagire con la shell

Sintassi dei comandi Unix

La sintassi tipica dei comandi UNIX è la seguente:

comando <opzioni> <argomenti>

<opzioni> generalmente consistono nel simbolo del "-" seguito da una sola lettera.

<argomenti> si possono avere piu`argomenti o anche nessuno in base al comando.

- history (freccia SU/GIU).
- I file system dei sistemi unix-like sono **case-sensitive**: maiuscole e minuscole sono importanti.

Il comando `man`

```
man comando
```

mostra la pagina del manuale di comando, con istruzioni sull'uso e sulle opzioni disponibili, es. man cd;

Cambiare la working directory

`cd [<dir>]`

- Serve per muoversi attraverso le directory. Il parametro **<dir>** è opzionale — se non viene indicato, il comando porta nella home directory.
- Per la navigazione risultano utili le directory: "**.**" (working directory), "**..**" (directory padre) e tilde "**~**" (directory home).

`pwd`

Per conoscere il path assoluto della directory in cui siamo

Visualizzare il contenuto di una directory

```
ls [-alsFR] [<dir1> ... <dirN>]
```

Se non viene specifica alcuna directory, si riferisce alla directory corrente.

- **Funzione di autocompletamento (tasto TAB)**
- **Metacaratteri (wildcards)**

`ls *.html` elenca tutti i file nella cartella corrente con nome che termina con ".html"

`ls ab*` elenca tutti i file/cartelle nella cartella corrente con nome che comincia con "ab"

`ls ab*c.html`

Eliminare file e directory

```
rm [-rif] <file1> ... <fileN>
```

- **-r <dir>** cancella la directory con il suo contenuto;
- **-i** prima di cancella il file chiede conferma all'utente;
- **-f** cancella senza chiedere conferma.

Per eliminare le directory si può usare anche il comando **`rmdir`**

Creare una directory

```
mkdir [-p] <dir1> ... <dirN>
```

- I parametri **dir** indicano i nomi (path assoluti o relativi) delle directory da creare.
  L'opzione **-p** crea eventuali directory intermedie esplicitate nei parametri **dir**.

- Nomi con caratteri come **/**, *****, **&** e **%** devono essere evitati per evitare possibili errori di sistema;

Copiare e spostare i file

```
cp [-if] <file1> <file2>
```

Copia **file1** in **file2** — se **file2** esiste viene sovrascritto!

```
cp [-if] <file1> ... <fileN> <dir>
```

Copia i **file** nella directory **dir** — se un **file** esiste in **dir** viene sovrascritto!

```
mv [-if] <file1> <file2>
```

Sposta **file1** in **file2** — se **file2** esiste viene sovrascritto!

```
mv [-if] <file1> ... <fileN> <dir>
```

Sposta i **file** nella directory **dir** — se un **file** esiste in **dir** viene sovrascritto!

Visualizzare e confrontare il contenuto di file

```
cat file
```

Stampa il contenuto di **file**

```
diff file1 file2
```

Stampa le righe di file1 che sono diverse da quelle di file2 e viceversa. Non stampa nulla se file1 e file2 sono uguali.

Altri comandi utili per visualizzare contenuti (se file hanno molte righe):

`head file` oppure **`tail file`** oppure **`more file`** o **`less file`**

Redirezione

Di default i comandi Unix prendono l'input da tastiera (**standard input** - **stdin**) e mandano l'output ed eventuali messaggi di errore su video (**standard output** - **stdout**, **standard error** - **stderr**).

L'input/output in Unix puo`essere rediretto da/verso file, utilizzando opportuni metacaratteri:

- > redirezione dell'output
- >> redirezione dell'output (append)
- < redirezione dell'input

Esempio

```
$ echo pippo Topolino

pippo Topolino

$ echo pippo Topolino > file.txt

$ cat file.txt

pippo Topolino

$ echo e anche Minnie » file.txt

$ cat file.txt

pippo Topolino e anche Minnie
```

Gestione dei Processi

Un processo è un programma in esecuzione. Quando viene lanciato un comando, viene creato un processo:

- **Ctrl-z** — combinazione di tasti che sospende il comando in esecuzione;
- **Ctrl-c** —Combinazione di tasti che termina il processo in esecuzione.

```
ps
```

— elenca i processi (e pid) della shell corrente; con opzione **-aux** elenca tutti i processi in esecuzione;

```
kill -signal_name <p>
```

— invia il segnale **signal_name** al processo con un certo pid <p>; **9** è il segnale di terminazione forzata di un processo;

Pipe

Il metacarattere "**|**" (pipe) serve per comporre comandi "in cascata" in modo che l'output di ciascuno sia fornito in input al successivo. L'output dell'ultimo comando è l'output della pipeline (di default sullo standard output).

```
command1 | command2
```

— l'output dell'esecuzione del primo comando viene passato come input del secondo comando.