DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

LINUX e UNIX

Prof. Luca Trevisan

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Padova
A.A. 2025/2026

LINUX e UNIX

Linux: dalle origini a oggi

Nel 1991, **Linus Torvalds**, uno studente finlandese dell'Università di Helsinki, iniziò a sviluppare un nuovo sistema operativo libero, insoddisfatto delle soluzioni esistenti.

Punti chiave:

- **1991**: prima versione di Linux rilasciata come software libero.
- **Licenza GNU**: Linux è distribuito con licenza GNU, che consente a chiunque di usare, modificare e condividere il codice.
- Collaborazione globale: molti sviluppatori in tutto il mondo hanno contribuito al progetto.

Linux oggi:

- Oggi, Linux è utilizzato su un'ampia varietà di dispositivi, dai server ai telefoni cellulari.

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Autenticazione e Gestione Utenti

Cos'è l'autenticazione?

- Nei sistemi UNIX, ogni utente ha un **username** e una **password** per identificarsi al momento del login. Questi dati sono memorizzati nel file **/etc/passwd**.

- A differenza dei sistemi come MS-DOS e i primi Windows, UNIX è stato **multi-utente** fin dall'inizio: più utenti possono usare lo stesso sistema contemporaneamente, sia in locale che da remoto.

Diritti di accesso

- Ogni oggetto del sistema (file, periferica, processo) ha specifici **diritti di accesso**. Questi diritti determinano quali operazioni un utente può eseguire.

- Gli utenti sono organizzati in **gruppi logici** per gestire i permessi in modo flessibile e personalizzato.

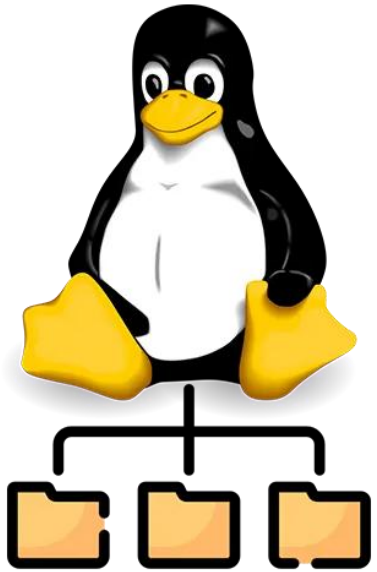
- Esiste un utente speciale, chiamato **root**, che ha il controllo completo del sistema. È l'account dell'amministratore e può eseguire qualsiasi operazione senza restrizioni.

Il Filesystem

Si tratta di una struttura (tipicamente ad albero) mantenuta su disco che contiene **tutti i dati del S.O. e dei suoi utenti**.

- **file**: unità di base di memorizzazione dei dati;
- **directory**: meccanismo di raggruppamento di altri oggetti in modo annidato, dando origine a strutture ad albero;
- altro.

Ogni filesystem ha una directory speciale che contiene tutti gli altri oggetti del filesystem (direttamente o indirettamente): **root** (o radice).



Il Path

Per lavorare con gli oggetti del filesystem ci serve poterli identificare in un modo ben preciso. Vengono utilizzati due metodi principali:

1. **Percorso Assoluto** (absolute path):

   - è il percorso che va dalla radice (inizia con il carattere /) del filesystem al file.

   - ogni elemento del path è separato dal precedente da un altro carattere /.

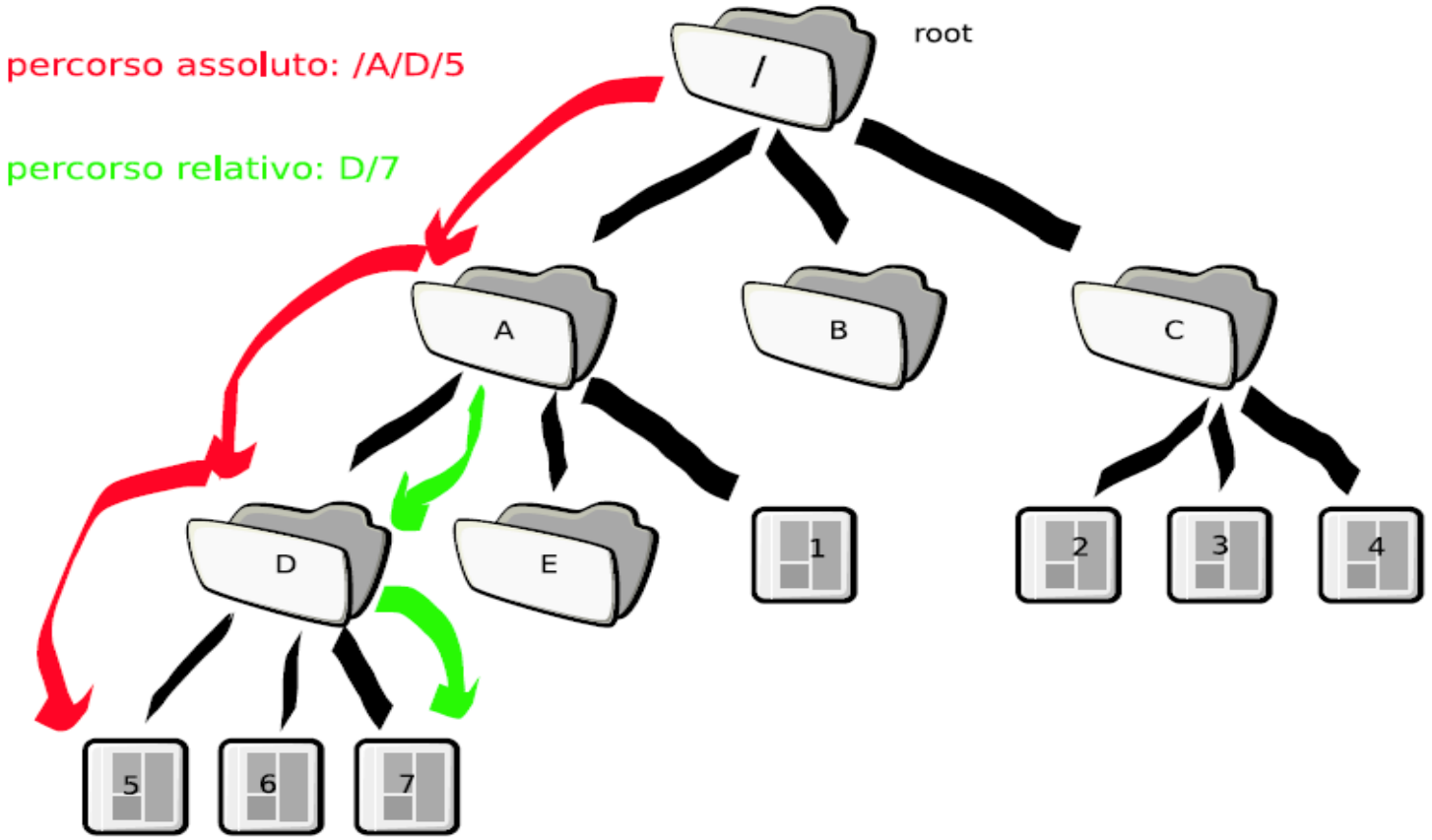
   Esempio: **`/home/utente/file.txt`**

2. **Percorso Relativo** (relative path):

   - fissando logicamente una directory (in genere quella corrente) è possibile identificare un file attraverso il percorso relativo che va da tale directory ad esso (non inizia mai con il carattere /).

   Esempio: **`utente/documenti/cv.pdf`**

Il Path


percorso assoluto: /A/D/5
percorso relativo: D/7
root
/
A
B
C
D
E
1
2
3
4
5
6
7

Il Path

- N.B.: il percorso assoluto di un file è il suo percorso relativo rispetto alla root.
- Nella costruzione dei percorsi possiamo utilizzare due "**cartelle virtuali**" che esistono in ogni directory:
  1. la cartella **.** : indica la **cartella stessa** ("auto-referenziamento");
  2. la cartella **..** : indica la **cartella genitore**.
- La cartella speciale **..** risulterà molto utile per navigare all'interno del nostro filesystem.

Il Filesystem Linux

Quanti filesytem ci sono?

- **MS-DOS/Windows**: si identifica ogni filesystem, uno per ogni disco, partizione, unità rimovibile, con una diversa lettera dell'alfabeto: A:, C:, ...

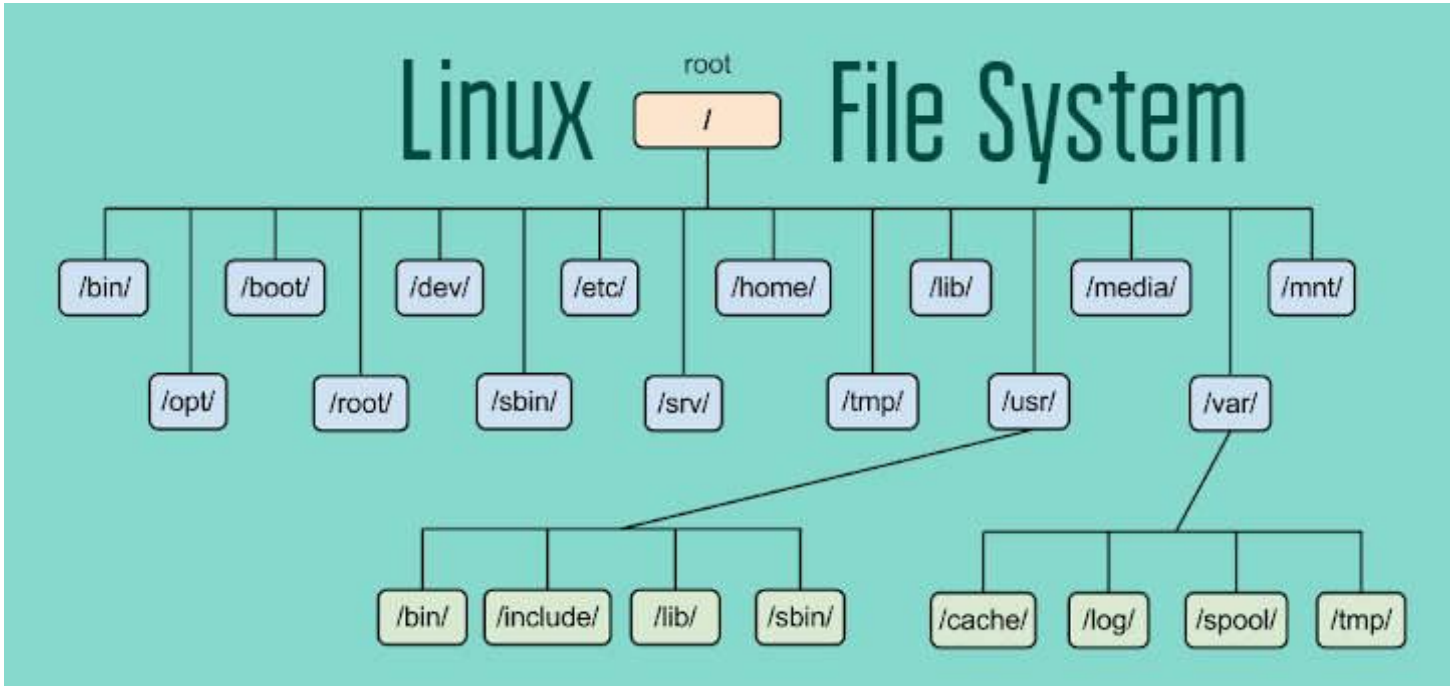
- **LINUX**: i diversi i filesystem sono **montati in cascata**.

  - Esiste un **filesystem principale** (in genere quello da cui viene caricato il S.O.) e la sua radice sarà la root di tutto il sistema.

  - Ogni filesystem secondario viene "montato" come se fosse un **ramo** dell'albero principale (i filesystem secondary nella cartella /mnt, mentre le unità rimovibili in /media).

  - Un **unico grande albero** ed ogni file del sistema può essere identificato con un path assoluto.

Struttura del Linux File System



- ***/***: la directory **radice** è la directory di livello superiore nel file system. Tutte le altre directory e file sono organizzati sotto questa directory.

- **/home**: la directory personale degli utenti. Ogni utente ha una sottodirectory come propria **home** (ad esempio, /home/nomeutente) dove sono memorizzati file e impostazioni personali.

Struttura del Linux File System

- **/bin**: contiene **eseguibili binari** essenziali (comandi) necessari affinché il sistema operi in modalità utente singolo e per tutti gli utenti. Esempi includono *ls*, *cp* e *mv*.
- **/boot**: contiene file necessari per l'**avvio del sistema**, inclusi il **kernel** Linux e i file di configurazione del **boot loader**.
- **/dev**: contiene file di dispositivo che rappresentano le **periferiche hardware** nel sistema, consentendo al software di interagire con l'hardware.
- **/etc**: contiene file di **configurazione** a livello di sistema per il sistema operativo e le applicazioni installate. Esempi includono configurazioni di rete e impostazioni degli account utente.
- **/lib**: contiene **librerie condivise** essenziali per l'esecuzione dei binari in /bin e /sbin. Questa directory è simile ai file DLL di Windows.

Struttura del Linux File System

- **/media**: Un punto di montaggio standard per i **media rimovibili** come unità USB, CD-ROM e DVD. Quando inserisci un dispositivo rimovibile, viene tipicamente montato qui.
- **/mnt**: un punto di montaggio **temporaneo** per i file system. Gli amministratori di sistema utilizzano spesso questa directory per montare file system manualmente.
- **/opt**: utilizzato per pacchetti software **opzionali** che non fanno parte della distribuzione standard di Linux. Le applicazioni installate qui sono tipicamente autonome.
- **/proc**: un file system **virtuale** che contiene informazioni sui processi di sistema e sui parametri del kernel. Viene utilizzato per accedere dinamicamente a informazioni su processi e sistema.

Struttura del Linux File System

- **/root**: la directory personale dell'utente **root** (l'account amministrativo).
- **/sbin**: contiene **eseguibili di sistema**, che sono comandi utilizzati principalmente dagli amministratori di sistema per compiti di manutenzione del sistema.
- **/srv**: contiene dati per i **servizi** forniti dal sistema, come file di server web o dati FTP.
- **/tmp**: una directory temporanea utilizzata per memorizzare **file temporanei**. I file qui vengono spesso eliminati al riavvio del sistema.
- **/usr**: contiene **utilità e applicazioni** per gli utenti. Ha sottodirectory come /usr/bin per i comandi degli utenti, /usr/lib per le librerie e /usr/share per file indipendenti dall'architettura.
- **/var**: contiene file **variabili**, come log, database e file di spool. Questa directory ospita dati che cambiano di dimensione, come email e log di sistema.

File Speciali

Esistono dei file speciali chiamati **file di dispositivo**:

- Identificano una particolare **periferica** del sistema ed attraverso esso il S.O. e i programmi possono interagire con tale periferica.
- In genere risiedono nella cartella predefinita **/dev/**.

Ne esistono di due tipi:

1. a **caratteri**: le operazioni sul dispositivo vengono fatte per flussi di **input/output** con il carattere come unità di base.

   Esempio: tastiera, stampante, scheda audio;

2. a **blocchi**: si opera con modalità ad accesso casuale e si **agisce per blocchi**.

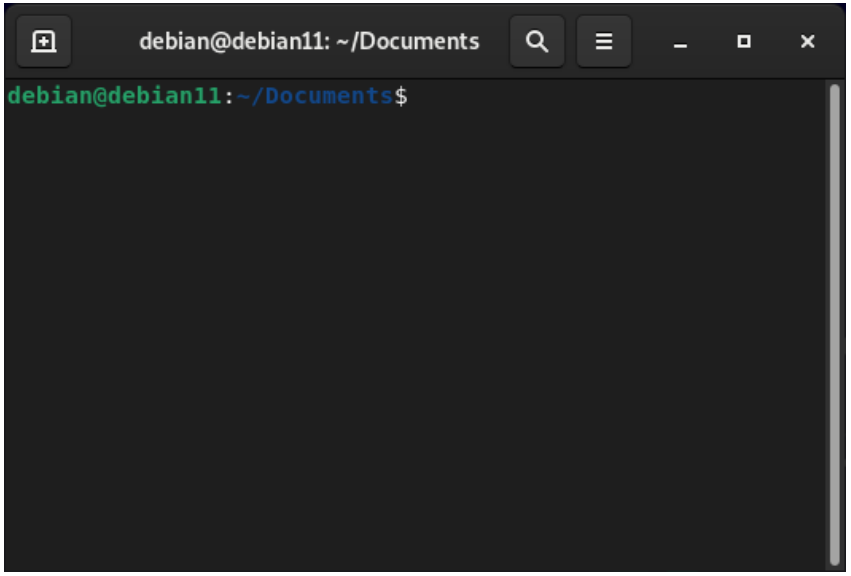
   Esempi: dischi SATA (/dev/sda,/dev/sdb,...), partizioni all'interno dei dischi (/dev/sda1,/dev/sda2,...), CD/DVD, terminali testuali (/dev/tty1), schede audio (/dev/dsp).

La shell LINUX

- La **shell** Linux è un componente fondamentale dei sistemi operativi basati su Linux e Unix.
- La shell è l'interfaccia utente a **riga di comando** che consente all'utente di interagire con il sistema operativo, eseguire comandi e gestire file e directory.

- Quello che ci viene presentato è un **prompt** che ci indica che il sistema è pronto a ricevere comandi da noi.
- **Esempio di prompt dei comandi:**

```
utente@host:directory$ _
```



La Shell Linux

- Essa offre una vasta gamma di **comandi**, che consentono all'utente di eseguire operazioni come:
  - la gestione dei file e delle directory;
  - la navigazione nel sistema di file;
  - la gestione dei processi e la configurazione del sistema.
- Possiamo inserire i comandi che vengono poi mandati in esecuzione premendo il tasto di invio (o “enter”).
- Finché il comando non viene inviato in esecuzione, abbiamo la possibilità di modificarne la sintassi correggendolo.

I comandi

I **comandi** che si possono invocare da una **shell** sono di due tipi:

1. **Comandi builtin**

   - Vengono eseguiti dalla shell stessa senza coinvolgere processi aggiuntivi.

   - Alcuni dei comandi builtin più comuni sono **`cd`**, **`ls`**, **`cp`**, **`mv`**, **`rm`**, **`mkdir`**, **`ps`** e **`kill`**.

2. **Comandi esterni**:

   - Applicazioni indipendenti dalla shell che sono memorizzate all'interno del filesystem ed eseguite indipendentemente dal processo della shell.

   - Possono andare dalla semplice utility testuale ad vere e proprie applicazioni grafiche complete;

I parametri

- Tipicamente un **comando** può richiedere dei **parametri**.
- Si inseriscono dopo il nome del comando separati da uno spazio.
- Questi parametri possono essere:
  - **file**: il **percorso** (assoluto o relativo) al/ai file su cui il comando deve lavorare;
  - **dati**: identificano dei **dati da passare** ed in genere sono passati sotto forma di stringa con una formattazione opportuna;
  - **switch** o **opzioni**: sono dei parametri che danno al comando alcune **indicazioni** sull'esecuzione del suo compito.

Le opzioni

- Le **opzioni** solitamente iniziano con un carattere "-".
- Spesso la stessa opzione ha più di una sintassi: una **breve** ed una **lunga**.
  - Quella breve è inizia con un **trattino** seguito da un **singolo carattere**, come ad esempio **`-h`** (opzione di richiesta di help).
  - Quella lunga inizia con un **doppio trattino** seguito da una **stringa**, ed ha in genere un nome più mnemonico, tipo **`--help`**.
- Spesso le opzioni si possono **collassare**: al posto di due opzioni **`-a -b`**, si può spesso utilizzare **`-ab`**.
- Alcuni parametri sono opzionali e nella sintassi vengono indicati tra parentesi quadre [-].

  Esempio: **`cal [[mese] anno]`**

Cosa fa questo comando?

- I comandi UNIX sono tanti e le opzioni sono centinaia!
- Usando l'opzione **`-h`** o **`--help`** è possibile ottenere una breve **descrizione** delle sue funzionalità e delle sue principali opzioni (nella lingua dell'OS).

- Per avere più aiuto ci sono altri metodi:

1. **`apropos`**:
   - Sintassi: **`apropos stringa`**
   - Visualizza la lista di tutti i comandi che contengono la stringa passata per parametro nella loro descrizione sintetica;

2. **`whatis`**:
   - Sintassi: **`whatis nome_comando`**
   - Visualizza la descrizione sintetica del comando passato come parametro;

3. **`man`**:
   - Sintassi: **`man nome_comando`**
   - Visualizza la pagina del manuale in linea relativa al comando passato come parametro.

Cosa c'è qui? ls

- Il comando **`ls`** consente di **visualizzare informazioni** sugli oggetti presenti nel filesystem.

- **Sintassi** (semplificata, ci sono altre opzioni): **`ls [-l] [-a] [-R] [pathname]`**

  - **`-pathname`**: identifica l'oggetto sul quale reperire le informazioni
  - **`-l`**: visualizza informazioni dettagliate
  - **`-a`**: visualizza anche i file nascosti
  - **`-R`**: visualizza anche il contenuto delle cartelle ricorsivamente
  - pathname: oggetto del filesystem sul quale visualizzare le informazioni

- Se pathname è una **directory** allora viene visualizzato **tutto il contenuto** di quest'ultima. Se lanciato senza parametri assume come pathname la **directory corrente**.

- È possibile specificare più pathname sulla stessa linea di comando, il comando visualizzerà informazioni per ciascuno di essi.

Esempio di uso di ls

```
$ ls
cartella esempio2.txt esempio.txt script.sh

$ ls -la
drwxr-xr-x 3 mario mario 4096    2005-09-20 12:30 .
drwxr-xr-x 3 mario mario 4096    2005-09-20 12:28 ..
drwxr-xr-x 2 mario mario 4096    2005-09-20 12:30 cartella
-rw-r--r-- 1 mario mario   27    2005-09-20 12:30 esempio2.txt
-rw-r--r-- 1 mario mario   21    2005-09-20 12:29 esempio.txt
-rwxr-xr-x 1 mario mario   10    2005-09-20 12:30 script.sh

$ ls -R
.:
cartella esempio2.txt esempio.txt script.sh

./cartella:
agenda.txt eseguimi.sh

$ ls -l cartella/
-rw-r--r-- 1 mario mario 21 2005-09-20 12:33 agenda.txt
-rwxr-xr-x 1 mario mario 10 2005-09-20 12:33 eseguimi.sh
```

I metacaratteri

Per abbreviare il nome di un file da specificare o per specificarne più di uno si possono utilizzare i metacaratteri:

- *****: rappresenta una qualunque stringa di 0 o più caratteri;
- **?**: rappresenta un qualunque carattere;
- **[ ]**: singolo carattere tra quelli elencati;
- **{ }**: stringa tra quelle elencate.

```
$ ls
esempio2.txt esempio3.txt esempio.txt script.sh scheda.pdf documento.pdf prova.sh

$ ls esempio*.txt
esempio2.txt esempio3.txt esempio.txt

$ ls esempio?.txt
esempio2.txt esempio3.txt

$ ls *.{txt,pdf}
documento.pdf esempio2.txt esempio3.txt esempio.txt scheda.pdf

$ ls /dev/tty[acd]{2-5}
/dev/ttya2 /dev/ttya4 /dev/ttyc2 /dev/ttyc4 /dev/ttyd2 /dev/ttyd4
/dev/ttya3 /dev/ttya5 /dev/ttyc3 /dev/ttyc5 /dev/ttyd3 /dev/ttyd5
```

Cambiamo directory: cd

- Il comando **`cd`** si usa per cambiare directory.
- Sintassi: **`cd [pathname]`**
  - **`pathname:`** diventa la nuova directory corrente!
- Se si omette il parametro pathname la directory corrente viene impostata alla home directory per l'utente attuale (in genere /home/nomeutente).
- Si possono usare sia i pathname assoluti che relativi, compresa l'utilissima directory virtuale **`..`**

Dove ci troviamo? pwd

- Il comando **pwd** (present working directory) permette di conoscere il **pathname assoluto** della directory corrente.

- Sintassi: `pwd`

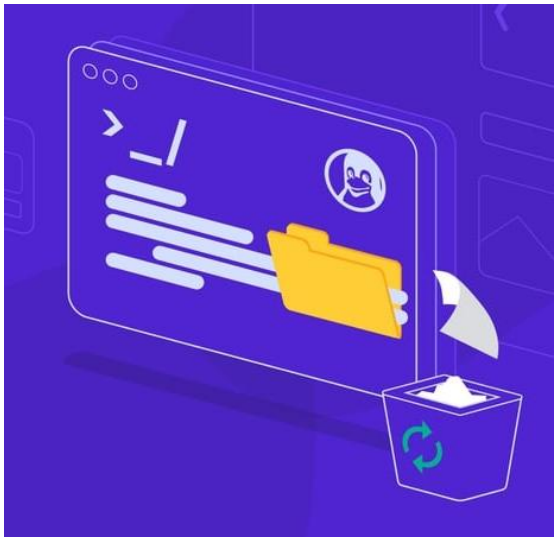
- Può sembrare scontato, ma in alcuni casi con cartelle omonime è meglio non fare confusione!

Creiamo una directory: mkdir

- Il comando **`mkdir`** modifica la struttura del filesystem creando le directory specificate mediante i parametri.
- Sintassi: **`mkdir [-p] pathname`**
  - **`pathname`**: percorso da creare.
  - **`-p`**: non genera errori se il pathname esiste già ed inoltre crea tutte le directory necessarie per creare il pathname passato.
- È possibile specificare più pathname sulla stessa linea di comando, il comando verrà eseguito per ciascuno di essi.

Eliminiamo una directory: rmdir

- Il comando **`rmdir`** modifica la struttura del filesystem **eliminando le directory** specificate mediante i parametri.
- Sintassi: **`rmdir [-p] pathname`**
  - **`pathname`**: directory da eliminare
  - **`-p`**: tenta di rimuovere **tutte** le directory che compongono il pathname;
- È possibile specificare più pathname sulla stessa linea di comando, il comando verrà eseguito per ciascuno di essi.
- Le directory devono essere **vuote** altrimenti non vengono cancellate.
- Esempio di utilizzo del parametron `-p`: un comando `rmdir -p /a/b/c/` è equivalente a `rmdir /a/b/c/ /a/b/ /a/`



Copiamo dei file: cp

- Il comando **`cp` copia** i file specificati dalla sorgenge (source) alla destinazione (dest).

- Sintassi (semplificata): **`cp [-R] [-i] source dest`**

  - **`-R`**: copia **tutto** il contenuto (**ricorsivamente**) di source se è una directory
  - **`-i`**: chiede conferma prima di **sovrascrivere** i file
  - **`source`**: oggetti da copiare in dest; si possono specificare più sorgenti nella stessa riga di comando
  - **`dest`**: destinazione in cui copiare i file specificati

- Se si specifica una sola sorgente (ad esempio un file) allora specificando il pathname completo di un file come destinazione, questo viene **copiato e rinominato**.
- Se si specifica una cartella esistente come destinazione, i file vengono copiati al suo interno.
- Se si specificano più sorgenti, allora la destinazione deve essere necessariamente una cartella (esistente).

Eliminiamo dei file: rm

- Il comando **`rm`** **elimina** tutti i file specificati con i parametri pathname.
- Sintassi: **`rm [-r] [-i] [-f] pathname`**
  - **`-r`**: se pathname è una directory, elimina **ricorsivamente** tutti i file o cartelle contenuti al suo interno
  - **`-i`**: chiede **conferma** prima di cancellare ogni file
  - **`-f`**: cancella gli oggetti **senza chiedere conferma**
  - **`pathname`**: oggetti da eliminare
- Se viene specificato il parametro **-r** vengono **eliminate ricorsivamente** tutte le directory presenti nel sottoalbero della directory specificata con pathname.
- **QUINDI**: Usando l'opzione **-r** ed abbastanza permessi è possibile **danneggiare irrimediabilmente** il sistema.
- Cosa succede se con privilegi elevati eseguo: **`rm -rf /`** ?

Spostiamo dei file: mv

- Il comando **`mv` sposta** il file o directory sorgente nella destinazione specificata.
- Sintassi: `mv source dest`
  - **`source`**: file o directory da spostare
  - **`dest`**: file o directory di destinazione
- Se si specifica solo una sorgente (un file o una directory) e la destinazione non esiste, allora la sorgente viene **rinominata** oltre che spostata.
- Se si specificano più parametri source, dest deve essere una directory.
- Nota: spostare una directory in un'altra implica lo spostamento di **tutto il sottoalbero** della directory sorgente.

Cosa contiene il file? cat

- Il comando **`cat`** permette di visualizzare (nel senso di mandare allo standard output) il **contenuto** di uno o più file.
- Sintassi: `cat [pathname]`
  - **`pathname`**: file da visualizzare

- Volendo si può invocare senza alcun parametro ed in questo caso cat prenderà il suo **standard input** come input.
- Molto utile quando combinato con i meccanismi di comunicazione tra processi.

Ripeti ciò che dico: echo

- Il comando **echo** dà in output una stringa passata come parametro.
- Sintassi: `echo [-n] [-e] [string]`
  - **-n**: non manda a capo il carrello quando ha finito;
  - **-e**: permette l'uso di alcuni caratteri speciali;
  - string: stringa da visualizzare.
- Esempi di **caratteri speciali** che è possibile utilizzare con l'opzione -e sono: \a bell (campanello), \n new line, \t tabulazione, \\ backslash.
- Si utilizza spesso per conoscere il contenuto di una **variabile**. Supponiamo di aver dichiarato una variabile **var**, e che **var = 100**.
  - Con il comando:

```
echo The value of variable var is $var
```

    otteniamo:

```
The value of variable var is 100
```

Flussi di dati di un processo

- Ogni processo ha **3 flussi di dati**:

  1. **standard input**: da cui prende il suo input; in genere corrisponde alla tastiera;
  2. **standard output**: in cui invia il suo output; in genere corrisponde al terminale video;
  3. **standard error**: in cui emette gli eventuali messaggi di errore; anche qui si usa in genere il terminale.

- Attraverso l'invocazione da riga di comando è possibile **redirezionare** tali flussi su altri file.

  - **>**: redireziona l'output su un file;
  - **>>**: redireziona l'output su un file in modalità append;
  - **<**: prende l'input da un file;
  - **2>**: redireziona lo standard error su un file.

Esempi di redirezione dell'I/O

```
$ ls
esempio2.txt esempio3.txt esempio.txt

$ ls > output.txt

$ cat output.txt
esempio2.txt esempio3.txt esempio.txt output.txt

$ echo Ciao >> output.txt

$ cat output.txt
output.txt esempio2.txt esempio3.txt esempio.txt Ciao

$ cat < output.txt
esempio2.txt esempio3.txt esempio.txt
Ciao

$ man comandochenonesiste
No manual entry for comandochenonesiste

$ man comandochenonesiste 2> /dev/null
```

Cambio dei permessi di accesso: chmod

- Il comando **`chmod`** permette di **cambiare i permessi** a file o cartelle.
- Sintassi: **`chmod [-R] mode [pathname]`**
  - **`-R`**: applica i permessi in modo **ricorsivo** alle sotto-cartelle.
  - **`pathname`**: oggetti a cui applicare la nuova maschera dei permessi.
  - **`mode`**: **nuova maschera** dei permessi: serve per indicare "chi può fare cosa" con quei file o cartelle.
- Esistono due sintassi per specificare il parametro mode: quella quella **simbolica** e quella **numerica**.

chmod: il parametro mode

Sintassi simbolica

- Si utilizza una **stringa mode** del tipo: **target grant permission**
  - **target**: definisce a chi cambiare i permessi tra **user owner (u)**, **group owner (g)**, **other (o)** o **all (a)** (user + group + other);
  - **grant**: un carattere tra **+**, **-**, e **=**. I caratteri **+** e **-** **aggiungono** e **sottraggono** i permessi specificati lasciando inalterati tutti gli altri. Il carattere **= imposta** la maschera esattamente ai permessi specificati;
  - **permission**: è una sottostringa di **rwx** ed indica i permessi che si vogliono alterare. I diritti sono di **read (r)**, **write (w)** ed **execute (x)**.

- Specifiche multiple vanno separate da virgola. Se target è omesso, si sottintende all.

- Esempi:

```
rw-r-----                        ↔   u=rw,g=r,o-rwx
rw-rw-rw-   +   u+x,o-rw         →   rwxrw-rw-
rw-r--r--   +   a-r,u+r          →   rw-------
rw-r--r--   +   +x               →   rwxr-xr-x
```

chmod: il parametro mode

Sintassi numerica

- Si usa un **numero ottale** a **3 cifre** in cui ogni cifra corrisponde rispettivamente a: **user owner (u)**, **group owner (g)**, **other (o)**.
- I diritti sono di **read (r)**, **write (w)** ed **execute (x)**. Ognuno è rappresentato da un bit: **1** per **abilitato**, **0** per **disabilitato**.
- Ne consegue che: **- (no permission) = 0**, **x (execute) = 1**, **w (write) = 2** e **r (read) = 4**.
- Quindi, il permesso di esecuzione e scrtittura per un determinate utente sarà indicato con 1+2=3, il permesso di esecuzione e lettura con 1+4=5, quello di scrittura e lettura con 2+4=6, e quello complete con 1+2+4=7.
- Esempio:

```
rw-r----   →   110 100 000   →   640
-wx----w-  →   011 000 000   →   302
rwxr-xr-x  →   111 101 101   →   755
```

Grazie per l'attenzione!