# Lezione 7
# Gestione dei file

Sistemi Operativi (9 CFU), CdL Informatica, A. A. 2022/2023
Dipartimento di Scienze Fisiche, Informatiche e Matematiche
Università di Modena e Reggio Emilia
http://weblab.ing.unimo.it/people/andreolini/didattica/sistemi-operativi

# Quote of the day

(Meditate, gente, meditate...)

**“I think the major good idea in UNIX was its clean and simple interface: open, close, read and write.”**

*Ken Thompson (1943-)*
*Programmatore*
*Ideatore dei SO UNIX e Plan 9*
*Ideatore dei linguaggi B e Go*
*Creatore di Belle (primo motore scacchistico americano)*

# SOLUZIONI DEGLI ESERCIZI

# Esercizio 1 (5 min.)

A quale file speciale di dispositivo corrisponde il dispositivo “primo disco SATA”? Che cosa si può ottenere con l’accesso a tale dispositivo?

# Studio del termine “SATA”

Si cerca con Google il termine “SATA hard disk”. Si ottiene un riferimento alla pagina di Wikipedia corrispondente:

SATA (Serial AT Attachment) è un bus veloce per la connessione di hard disk al bus host.
È una evoluzione dei bus PATA (Parallel ATA) e EIDE (Extended Integrated Device Electronics).
È una alternativa al bus SCSI (Small Computer Systems Interface).

# Identificazione directory del file

Si parla di “file speciale di dispositivo”. Pertanto, si parla di un file nella directory **`/dev`**, che contiene tutti i file speciali di dispositivo.

# Identificazione classe di dispositivo

Bisogna cercare di individuare il tipo di file speciale che corrisponde ad un disco SATA. A tal scopo, si può cercare con **apropos** una pagina di manuale contenente la parola chiave **disk**. Si effettua la ricerca nella sezione 4 del manuale, che documenta i file speciali di dispositivo.

```
apropos -s 4 disk
```

Si ottiene l'output seguente:

```
ram (4) - dispositivo per ram disk
hd (4) - MFM/IDE hard disk devices
initrd (4) - boot loader initialized RAM disk
sd (4) - driver for SCSI disk drives
```

# Identificazione classe di dispositivo

Dalla descrizione iniziale del bus, andando per esclusione si potrebbe provare a dedurre che la classe di dispositivo sia **`sd`**.

**`hd`** si riferisce ai dischi rigidi che usano il bus più vecchio IDE (l'antesignano di EIDE).

**`initrd`** e **`ram`** si riferiscono ad oscuri "dischi in RAM", e probabilmente non sono correlati con i dischi rigidi.

# Lettura pagina di manuale **sd**

Si legge la pagina di manuale della classe di dispositivi **sd**:

```
man 4 sd
```

La pagina parla solo di dispositivi attaccati al bus SCSI, e non fa menzione del bus SATA.

Dubbio: e se **sd** non fosse la classe di dispositivi SATA?

# Indagine sulla correlazione sd ←→ SATA

Si cerca con Google il termine “linux sd device sata”. Si ottiene un riferimento ad un documento del Linux Documentation Project che spiega le convenzioni usate per i file speciali di dispositivo:
https://tldp.org/HOWTO/Partition-Mass-Storage-Definitions-Naming-HOWTO/x99.html

Il documento parla chiaro: **`sd`** è la classe di tutti gli hard disk moderni che si connettono con un bus SATA o SCSI.

# Identificazione del file speciale

Leggendo la pagina di manuale di **`sd`**:

**`man 4 sd`**

si scopre che il formato dei file speciali di dispositivo è:

**`/dev/sdlp`**

dove:

**`l`** è una lettera dell'alfabeto (partendo dalla **`a`**) che denota l'intero dispositivo fisico;

**`p`** è un numero intero che denota la partizione all'interno del dispositivo fisico.

Pertanto, il file speciale di dispositivo associato al primo dispositivo SATA è **`/dev/sda`**.

# Accesso al file speciale con `less`

Si lanci un emulatore di terminale e si ottengano le credenziali di amministratore con il comando seguente:

```
su -
```

Si immetta la password di amministratore.

Si legga il contenuto del file speciale:

```
less /dev/sda
```

Si ottiene il contenuto del disco a basso livello! I byte, opportunamente interpretati, costituiscono la tabella delle partizioni, il file system, i file, le directory.

# Esercizio 2 (3 min.)

Si stampi l'insieme dei metadati per i file seguenti:

```
$HOME
/tmp/.X11-unix/X0
/dev/sda1
/dev/tty0
```

Notate delle differenze?

# Stampa dei metadati dei file con **`stat`**

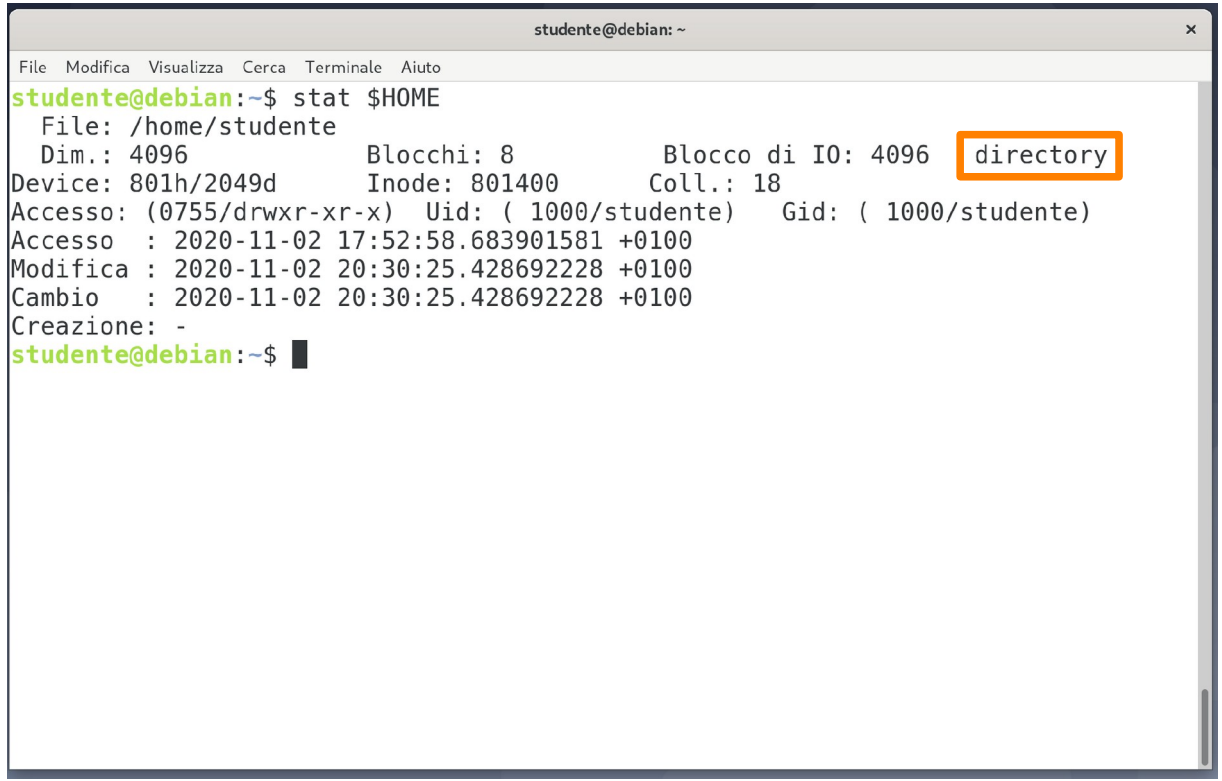
Si usa il comando **`stat`** per stampare i metadati dei file:

```
stat $HOME
stat /tmp/.X11-unix/X0
stat /dev/sda1
stat /dev/tty0
```

Nelle slide seguenti si evidenziano gli output dei comandi.

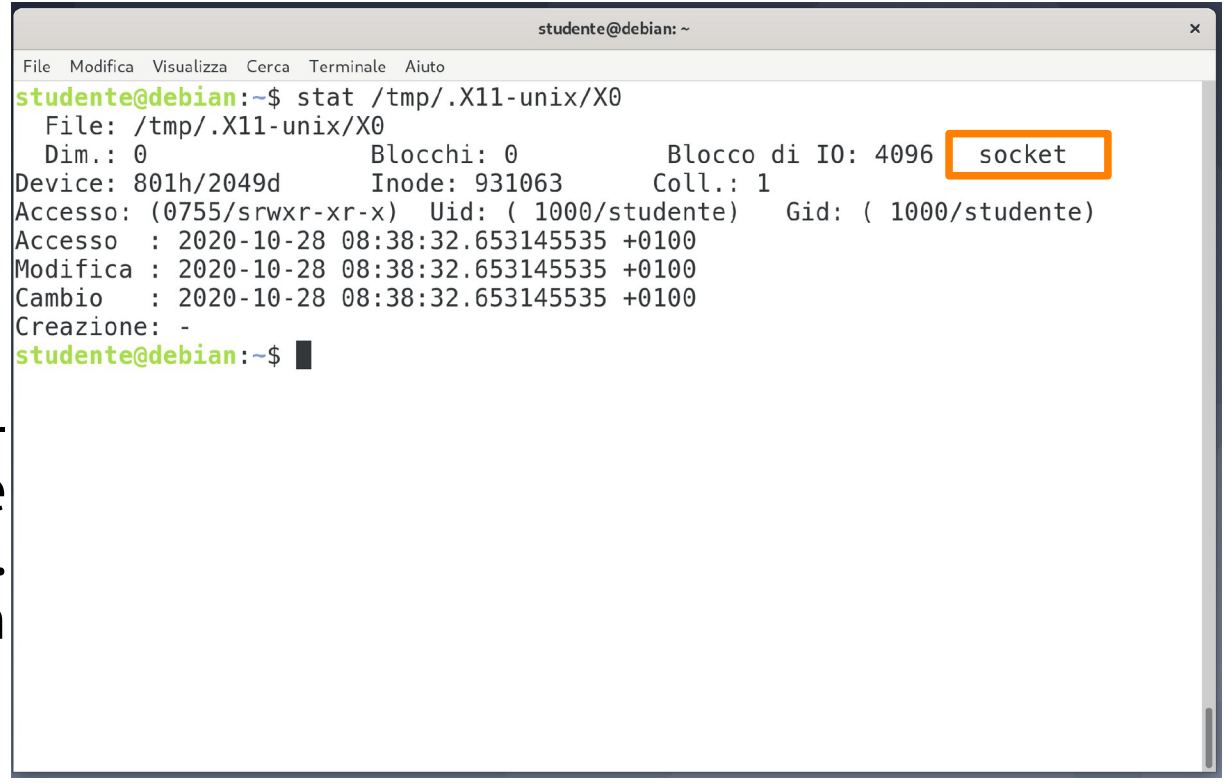
# Stampa dei metadati dei file con **`stat`**

**`$HOME`** è una directory.



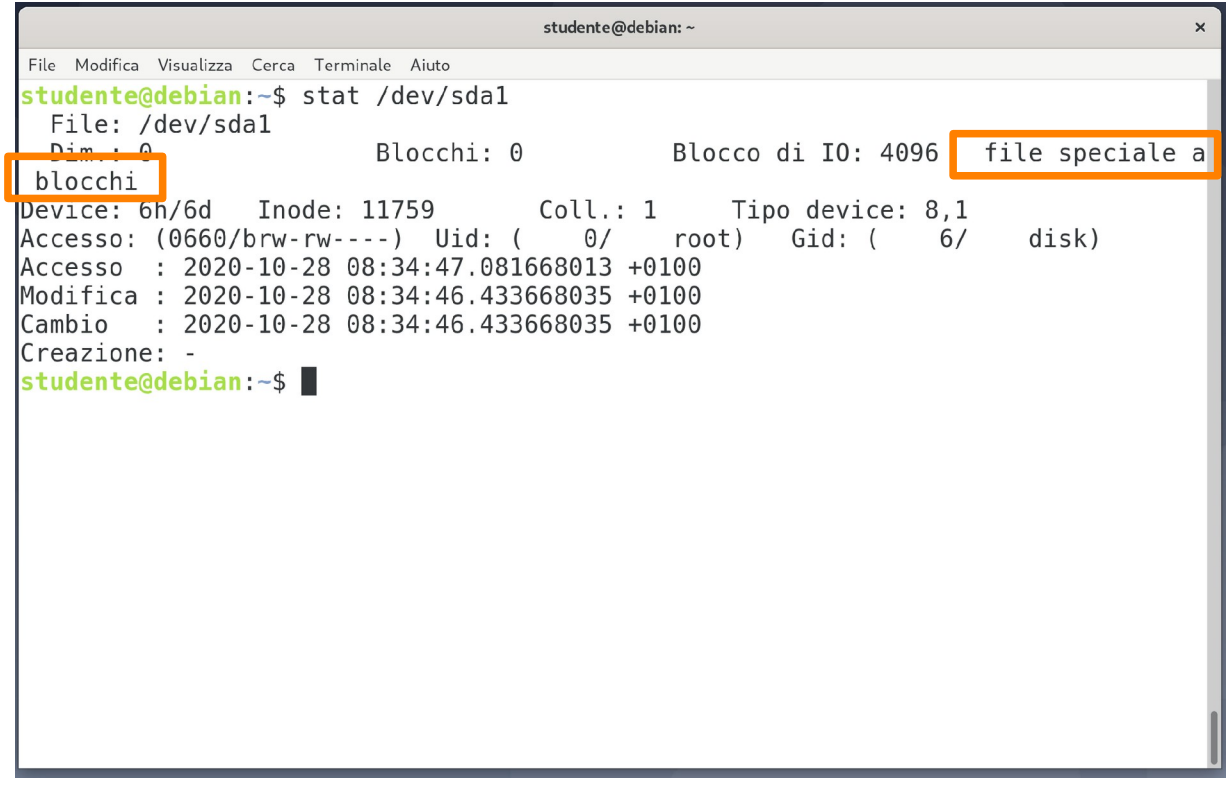
# Stampa dei metadati dei file con **`stat`**

**`/tmp/.X11-unix/X0`** è un socket.
Il socket è un meccanismo di comunicazione (locale o remoto). Non sarà affrontato in questo corso.



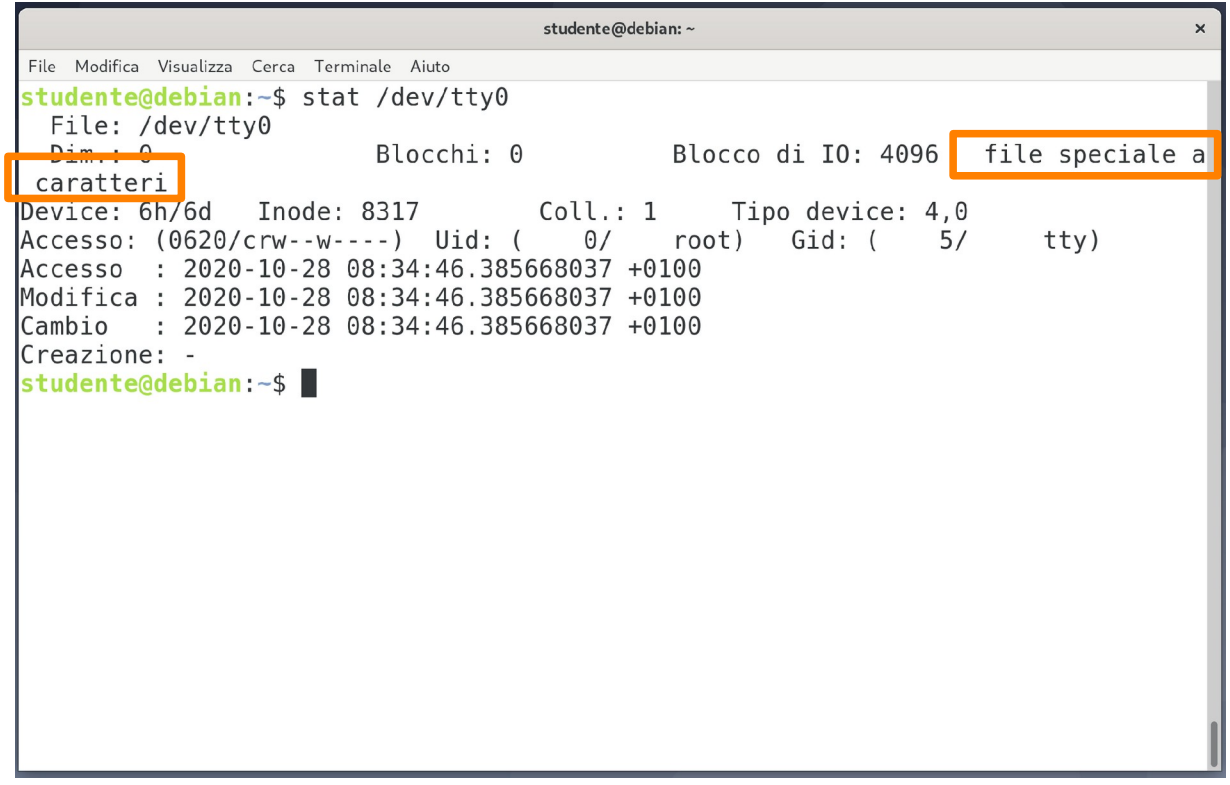
# Stampa dei metadati dei file con **`stat`**

**`/dev/sda1`** è un file speciale di dispositivo a blocchi. Esso rappresenta la prima partizione del primo disco rigido SATA.



# Stampa dei metadati dei file con **`stat`**

**`/dev/tty0`** è un file speciale di dispositivo a caratteri. Esso rappresenta il primo dispositivo terminale.



# Esercizio 3 (1 min.)

Elencate la directory **`/etc`** nella modalità seguente:

- ricorsivamente;
- con i metadati;
- con i file nascosti.

# Lettura pagina di manuale di `ls`

Si legge la pagina di manuale di **`ls`**:

```
man ls
```

L'opzione **`-R`** consente di elencare le sottodirectory ricorsivamente.
L'opzione **`-l`** consente di visualizzare i principali metadati del file.
L'opzione **`-a`** consente di visualizzare i file nascosti.

# Il comando richiesto

In definitiva, il comando richiesto è il seguente:

```
ls -alR /etc
```

# Esercizio 4 (1 min.)

Individuate il tipo di file dei file seguenti:

```
$HOME
/tmp/.X11-unix/X0
/dev/sda1
/dev/tty0
/usr/bin/editor
```

# Individuazione tipologia file con **`file`**

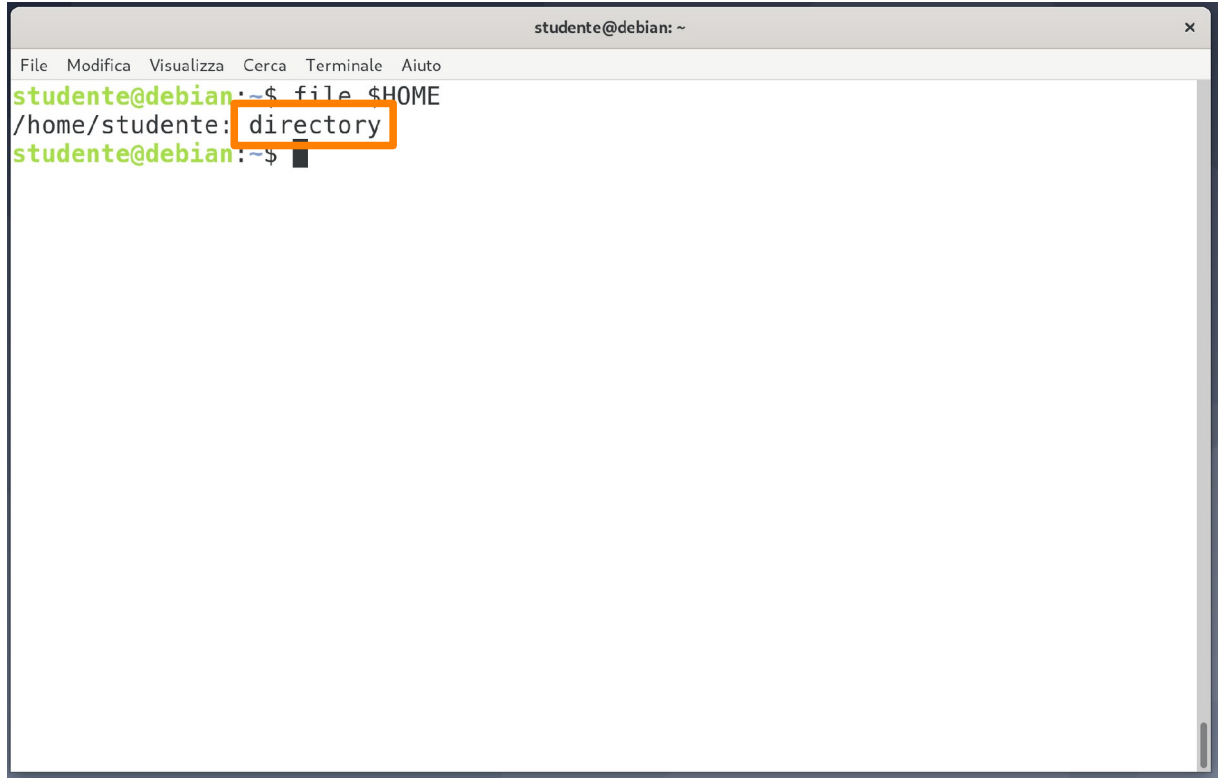
Si usa il comando **`file`** per individuare la tipologia dei file richiesti:

```
file $HOME
file /tmp/.X11-unix/X0
file /dev/sda1
file /dev/tty0
file /usr/bin/editor
```

Nelle slide seguenti si evidenziano gli output dei comandi.

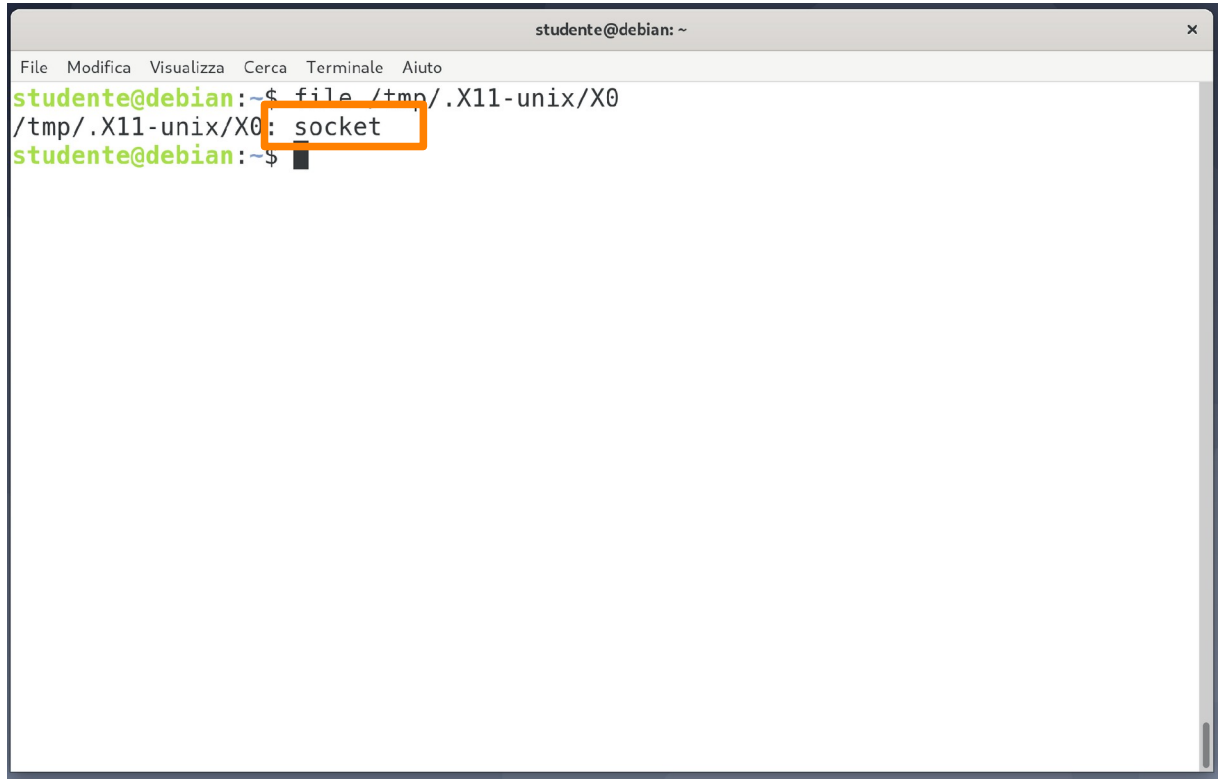
# Individuazione tipologia file con `file`

**`$HOME`** è una directory.



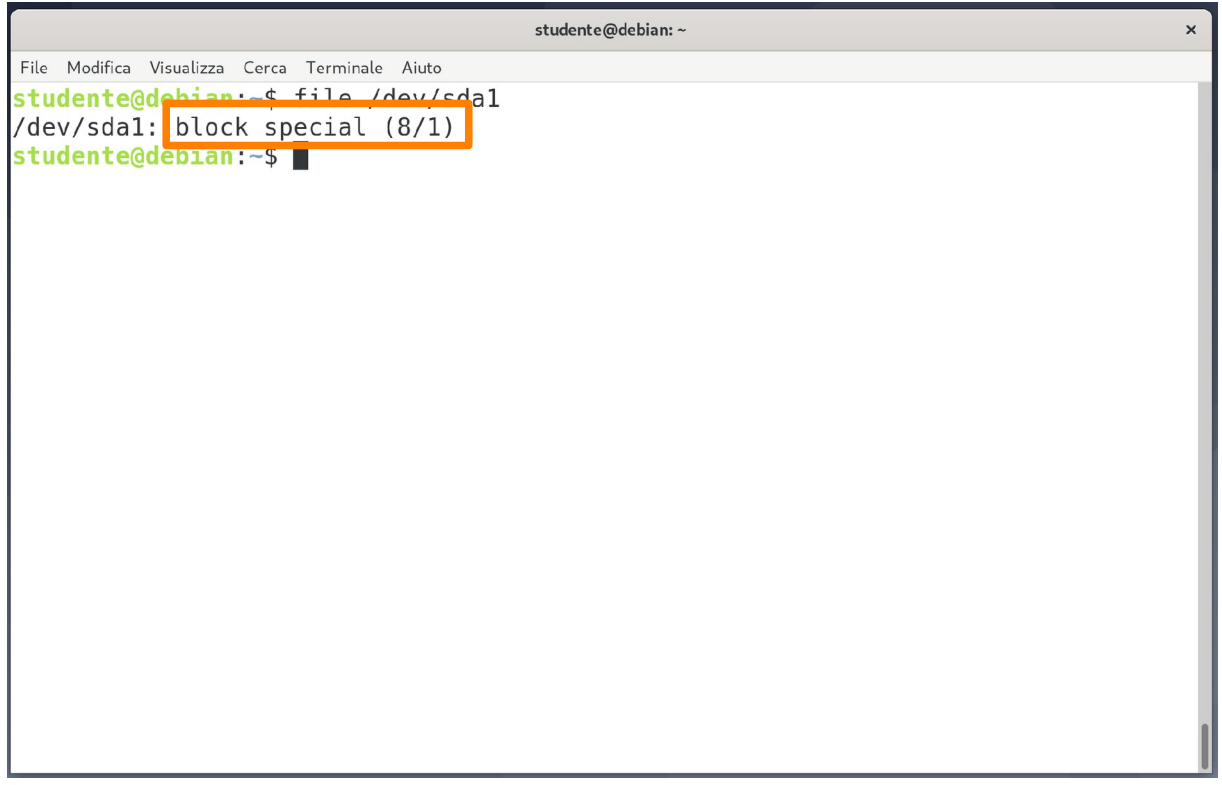
# Individuazione tipologia file con `file`

**`/tmp/.X11-unix/X0`**
è un socket.



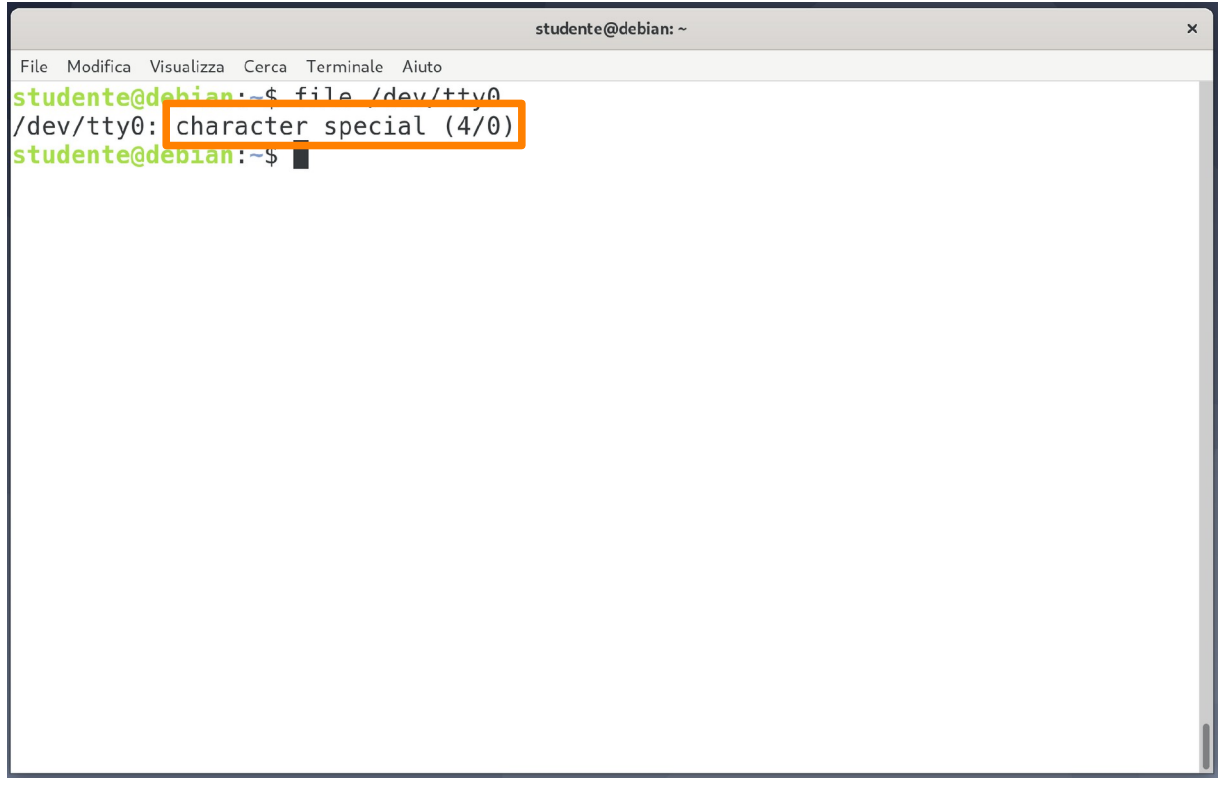
# Individuazione tipologia file con `file`

**`/dev/sda1`** è un file speciale di dispositivo a blocchi.
Major → 8 (SATA)
Minor → 1 (prima part.)

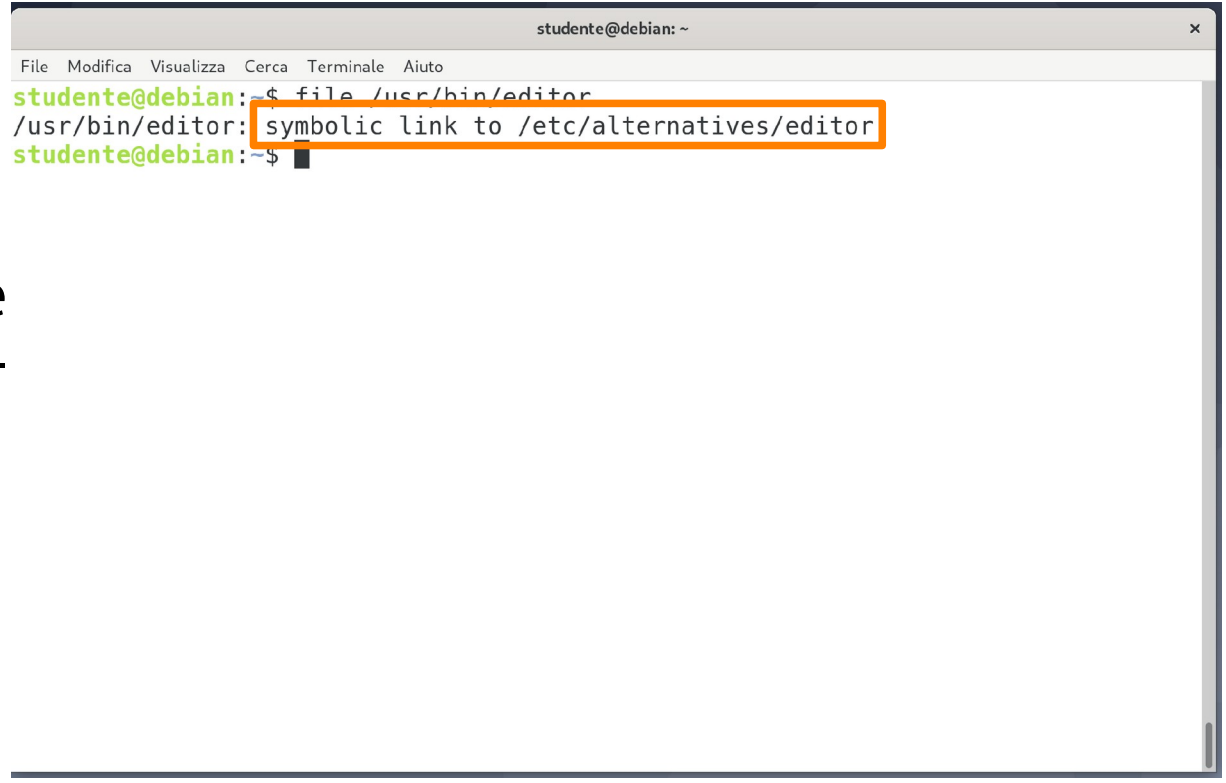


# Individuazione tipologia file con `file`

**`/dev/tty0`** è un file speciale di dispositivo a caratteri.
Major → 4 (TTY)
Minor → 0 (primo term.)



# Individuazione tipologia file con `file`




**`/usr/bin/editor`** è un collegamento simbolico ad un altro file.

# Esercizio 5 (3 min.)

Si crei un file nuovo di contenuto nullo. Leggendo la pagina manuale del comando relativo, si individui un modo per impostare i timestamp al 1/1/1970, ore 00:00.

# Lettura pagina di manuale di **`touch`**

Si legge la pagina di manuale di **`touch`**:

```
man touch
```

L'opzione **`-t`** consente di impostare un timestamp diverso da quello attuale. La forma Backus-Naur del timestamp è la seguente:

```
[[CC]YY]MMDDhhmm[.ss]
```

# Impostazione del timestamp

Il timestamp corrispondente alla data 1/1/1970, ore 00:00 è il seguente:

```
197001010000.00
```

Pertanto, il comando richiesto è il seguente:

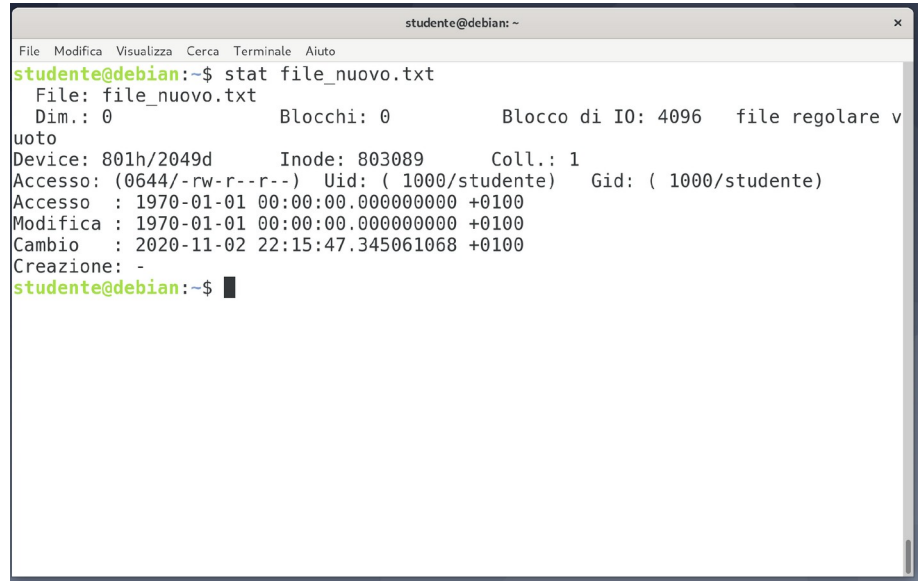
```
touch -t 197001010000.00 file_nuovo.txt
```

# Visione dei metadati

Si possono verificare i timestamp con il comando **`stat`**:

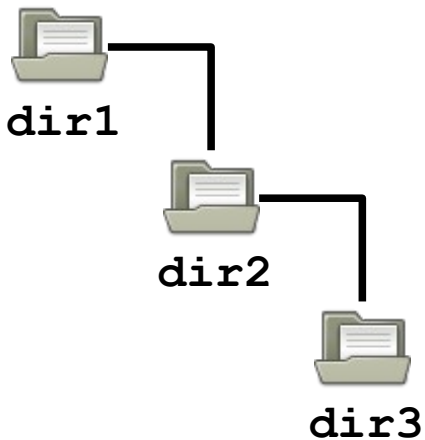
```
stat file_nuovo.txt
```

Si noti l'uguaglianza dei timestamp di accesso e modifica alla data desiderata.
Il timestamp di cambio, invece, riflette la vera data di creazione!



# Esercizio 6 (1 min.)

Si crei la seguente gerarchia di directory.



# Creazione delle directory con **`mkdir`**

La gerarchia di directory richiesta è creabile con il comando **`mkdir`**. Si può usare l'opzione **`-p`** per creare le directory mancanti.

```
mkdir -p dir1/dir2/dir3
```

# Esercizio 7 (2 min.)

Create dieci file vuoti con i nomi seguenti:

```
file1  file2  file3  file4  file5  file6
file7 file8 file9 file10
```

Cancellate i file appena creati.

# Creazione con ciclo for e **`touch`**

Si usa un ciclo for per creare i dieci file:

```
for i in 1 2 3 4 5 6 7 8 9 10; do
   touch file$i
done
```

# Creazione con ciclo for e `rm`

Si usa un ciclo for per cancellare i dieci file:

```
for i in 1 2 3 4 5 6 7 8 9 10; do
   rm file$i
done
```

# Esercizio 8 (2 min.)

Create un backup della propria home directory nella directory **`/tmp`**.

# Creazione copia archiviale con **`cp -a`**

Si crea una copia archiviale della directory **`$HOME`** con l'opzione **`-a`** del comando **`cp`**:

```
cp -a $HOME /tmp
```

# Esercizio 9 (3 min.)

Studiate la pagina di manuale di **`mv`** e individuate un metodo per creare backup dei file sovrascritti.

# Lettura pagina di manuale di **mv**

Si legge la pagina di manuale di **mv**:

```
man mv
```

L'opzione **-b** crea un backup del file destinazione secondo il formato standard (viene attaccato il carattere **~** al nome del file).

# Creazione dei file

Si apre un editor di testo e si crea il file **`file1.txt`** con il contenuto seguente:

```
file1
```

Si apre un editor di testo e si crea il file **`file2.txt`** con il contenuto seguente:

```
file2
```

# Spostamento e backup con **`mv -b`**

Si sposta **`file1.txt`** in **`file2.txt`** attivando il backup:

```
mv -b file1.txt file2.txt
```

A questo punto:

**`file1.txt`** dovrebbe sovrascrivere **`file2.txt`**;
dovrebbe comparire un file **`file2.txt~`**, backup di **`file2.txt`**.

# Esercizio 10 (2 min.)

Leggendo la pagina di manuale del comando opportuno, individuate un modo per stampare un file "al contrario" (dall'ultima alla prima riga).
Applicate tale metodo per stampare le righe del file **`/etc/passwd`** dall'ultima alla prima.

# Lettura pagina di manuale di **`cat`**

Si legge la pagina di manuale di **`cat`**:

```
man cat
```

Alla fine della pagina di manuale (nella sezione SEE ALSO) si fa menzione del comando **`tac`**.

# Lettura pagina di manuale di **`tac`**

Si legge la pagina di manuale di **`tac`**:

```
man tac
```

Il comando **`tac`** stampa le righe di un file dall'ultima alla prima.

# Stampa di **`/etc/passwd`** con **`tac`**

Si stampa il file **`/etc/passwd`** con **`tac`**:

```
tac /etc/passwd
```

# Esercizio 11 (3 min.)

Visualizzate nella forma canonica di **`hexdump`** i primi 32 byte dei file seguenti:

**`/usr/bin/ls`**
**`/usr/bin/bash`**
**`/usr/bin/which`**

Notate qualche differenza?

# Lettura pagina di manuale di **`hexdump`**

Si legge la pagina di manuale di **`hexdump`**:

```
man hexdump
```

L'opzione **`-n N`** consente di effettuare il dump dei primi **`N`** byte.

# Stampa dei file richiesti con **hexdump**

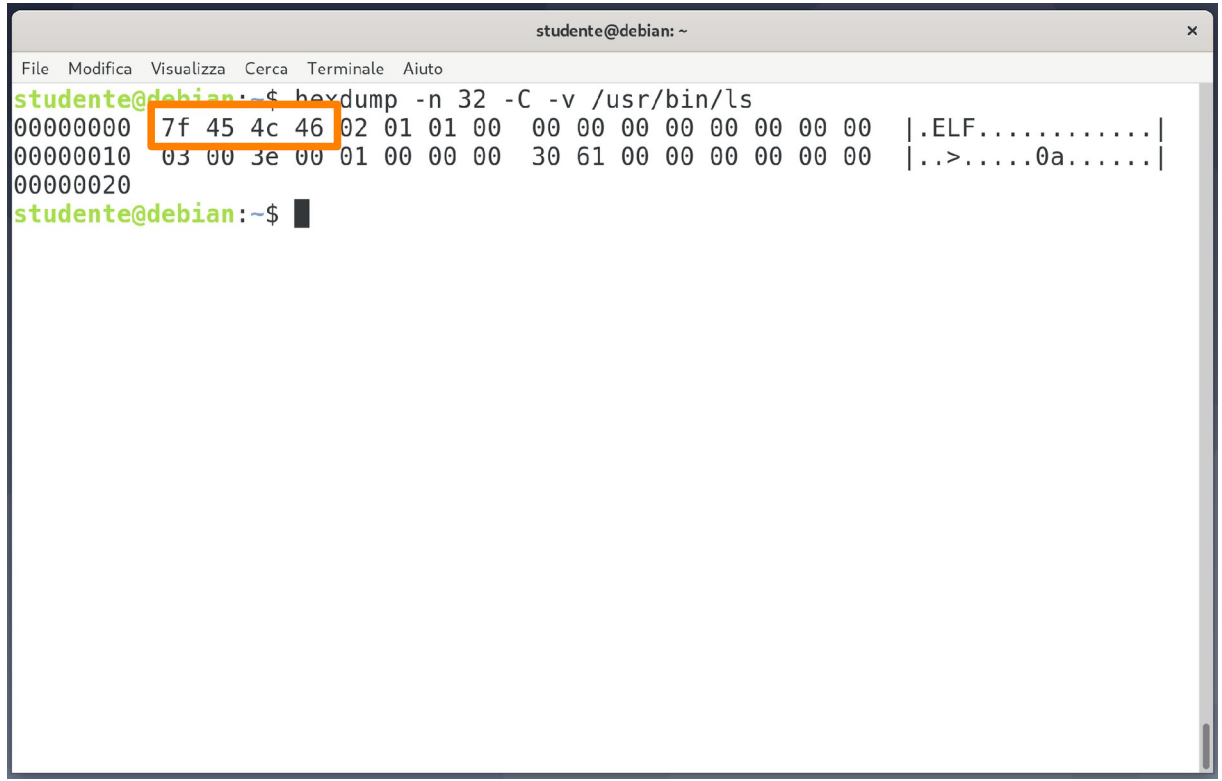
Per visualizzare nella forma canonica i primi 32 byte dei file richiesti, si può eseguire **hexdump** con l'opzione **-n 32**:

```
hexdump -n 32 -C -v /usr/bin/ls
hexdump -n 32 -C -v /usr/bin/bash
hexdump -n 32 -C -v /usr/bin/which
```

Nelle slide seguenti si evidenziano gli output dei comandi.

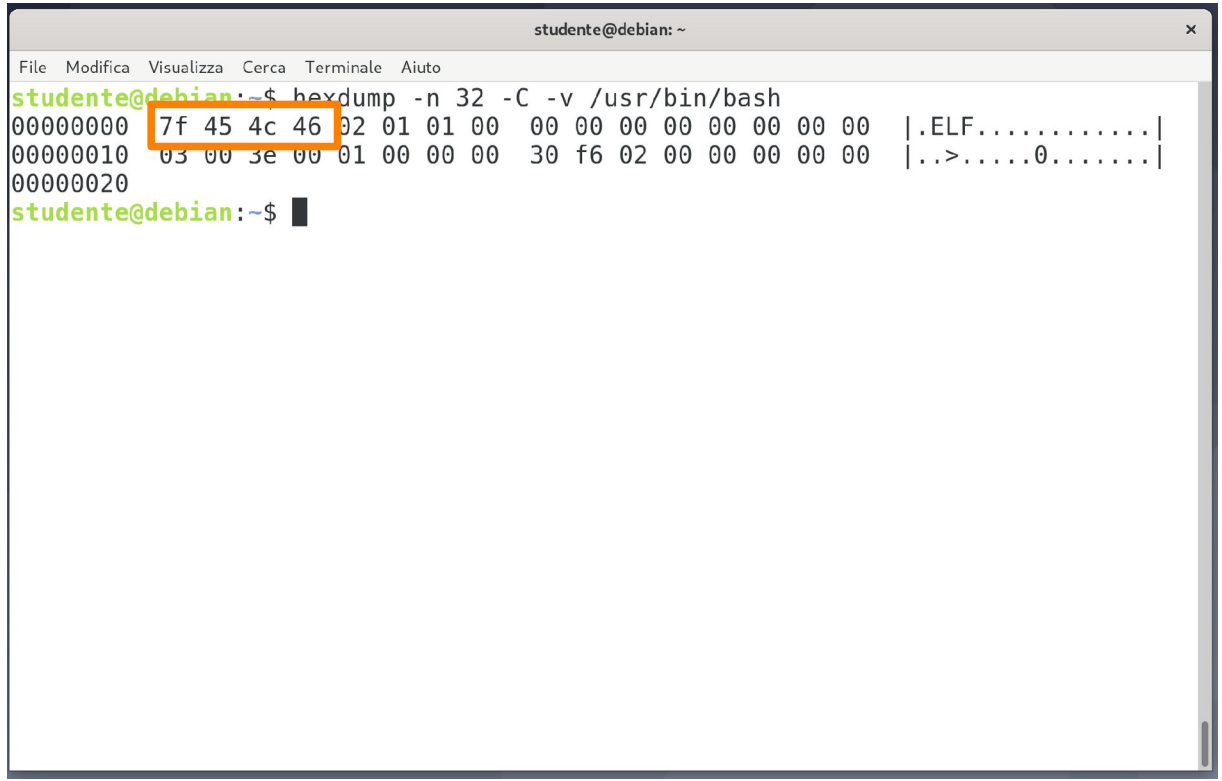
# Stampa dei file richiesti con **hexdump**

I primi quattro byte 7f 45 4c 46 sono la signature dei file in formato ELF.
Questo è un binario eseguibile.



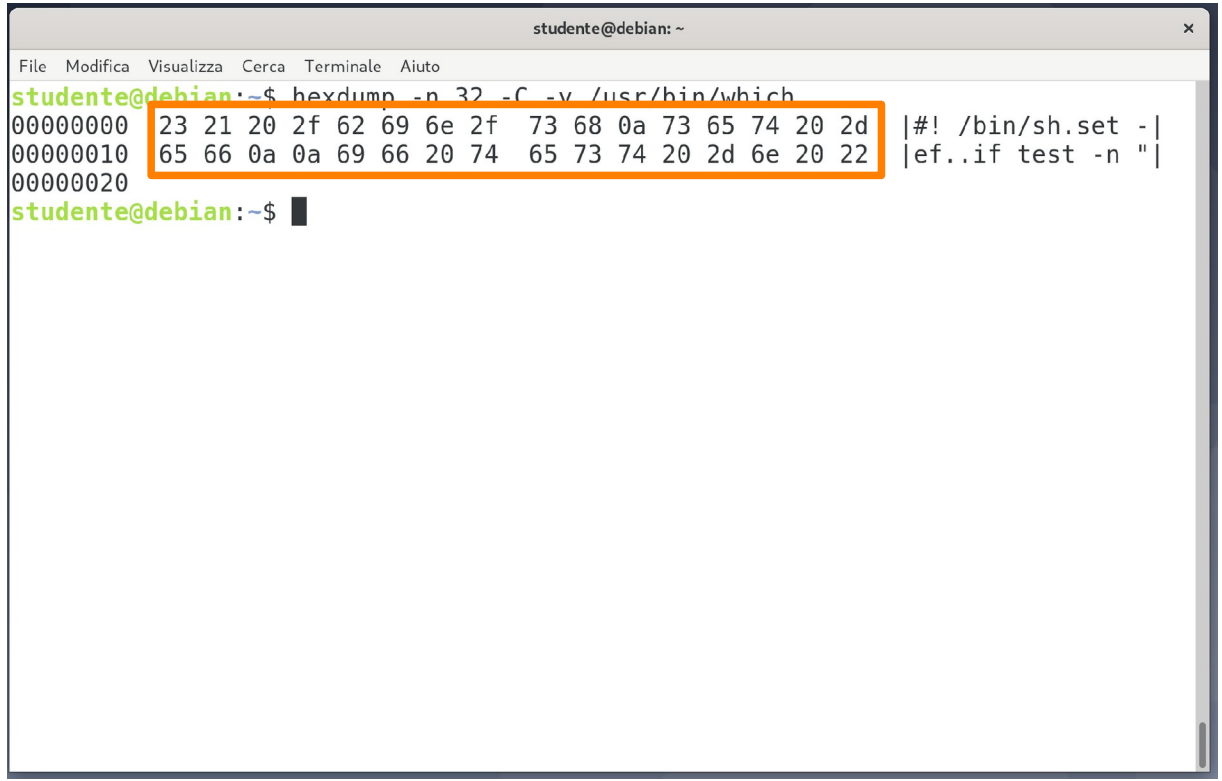
# Stampa dei file richiesti con **hexdump**

I primi quattro byte 7f 45 4c 46 sono la signature dei file in formato ELF.
Questo è un binario eseguibile.



# Stampa dei file richiesti con **hexdump**

I primi trentadue byte evidenziano uno script di shell.



# Esercizio 12 (3 min.)

Individuate due modi distinti in **`head`** per stampare i primi 1024 byte del file **`/bin/ls`**.

# Lettura pagina di manuale di **head**

Si legge la pagina di manuale di **head**:

```
man head
```

L'opzione **-c N** consente di stampare i primi **N** byte di un file.

Il numero **N** può espresso in forma umana (con le unità di misura internazionali):

```
1024
1K
```

# Stampa primo 1KB di **`/bin/ls`**

I due comandi seguenti stampano i primi 1024 byte di **`/bin/ls`**:

```
head -c 1024 /bin/ls
head -c 1K /bin/ls
```

# Esercizio 13 (3 min.)

Stampate il primo ed il settimo campo di ogni riga dell'archivio testuale **`/etc/passwd`**.

# Stampa dei campi richiesti con **`cut`**

Il file **`/etc/passwd`** ha per separatore dei campi il carattere `:`. Poiché tale separatore non è quello di default, va impostato con l'opzione **`-d ":"`**.
I campi richiesti sono il primo e il settimo; essi sono selezionabili con l'opzione **`-f 1,7`**.

In definitiva, il comando richiesto è:

```
cut -d ":" -f 1,7 /etc/passwd
```

# Esercizio 14 (1 min.)

Stampate i primi tre caratteri di ogni riga dell'archivio testuale **`/etc/passwd`**.

# Stampa dei campi richiesti con **`cut`**

È richiesta la stampa dei primi tre caratteri di ogni riga. Essi sono selezionabili con l'opzione **`-c 1-3`**.

In definitiva, il comando richiesto è:

```
cut -c 1-3 /etc/passwd
```

# Esercizio 15 (2 min.)

Producete un output multicolonnare nel modo seguente.
**Prima colonna**. Una sequenza di numeri interi crescenti a partire da 1.
**Seconda colonna.** Gli username esistenti nel sistema (primo campo).
**Terza colonna.** Le shell assegnate agli username (ultimo campo).

# Produzione prima colonna

Si apre il file **`/etc/passwd`** e si legge il numero di righe dalla riga di stato.
In Debian 10, il numero di righe è 38.

Si crea un file **`1.txt`** con la sequenza di interi da 1 a 38 (un intero per riga).

# Produzione seconda colonna

Si estrae il primo campo di **`/etc/passwd`** con **`cut`**:

```
cut -d ":" -f 1 /etc/passwd
```

Si crea un file **`2.txt`** con l'output del comando precedente.

# Produzione terza colonna

Si estrae il settimo campo di **`/etc/passwd`** con **`cut`**:

```
cut -d ":" -f 7 /etc/passwd
```

Si crea un file **`3.txt`** con l'output del comando precedente.

# Produzione file multicolonnare

Si usa il comando **`paste`** per produrre l'output multi-colonnare richiesto:

```
paste 1.txt 2.txt 3.txt
```

# Esercizio 16 (2 min.)

Ordinate nel modo seguente il file **`/etc/passwd`**.
Ordinamento: numerico crescente.
Campo: quarto.
Separatore: due punti.

# Ordinamento numerico tramite **sort**

Per ordinare numericamente **/etc/passwd** si ricorre al comando **sort**, con le opzioni seguenti.

- Impostazione del separatore dei campi: **-t ":"**
- Selezione della chiave di ordinamento: **-k 4**
- Selezione del criterio di ordinamento numerico: **-n**

In definitiva, il comando richiesto è il seguente:

```
sort -t ":" -k 4 -n /etc/passwd
```

# Esercizio 17 (2 min.)

Nell'ipotesi che un file di configurazione verifichi il pattern di shell `*.log`, individuate tutti i file di log nel sistema.

# Identificazione dei file con **`find`**

Per identificare i file a partire dal loro nome si può usare il comando **`find`**.
Poiché è richiesta l'individuazione a partire da un pattern, si può usare l'opzione **`-name "PATTERN"`**.
Non ci si dimentichi di quotare il pattern, altrimenti l'espansione di shell potrebbe alterare il comando.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **`/`**.
In definitiva, il comando richiesto è il seguente:

```
find / -name "*.log"
```

# Esercizio 18 (3 min.)

Individuate tutti i file HTML nel sistema.
In prima approssimazione, considerate come file HTML un file terminante in:

- `.html`
- `.htm`
- `.HTML`
- `.HTM`

# Scrittura di una espressione regolare

Per individuare i file che terminano con **`.htm`** si può usare l'espressione regolare seguente:

```
^.*\.htm$
```

Per trovare i file che terminano **`.htm`** o **`.html`** si può aggiungere il match di al più una lettera **`l`**:

```
^.*\.html?$
```

# Identificazione dei file con **`find`**

Per identificare i file a partire dal loro nome si può usare il comando **`find`**.
Poiché è richiesta l'individuazione a partire da una espressione regolare case insensitive, si può usare l'opzione **`-iregex "REGEX"`**.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **`/`**.
In definitiva, il comando richiesto è il seguente:

```
find / -iregex "^.*\.html?$"
```

# Esercizio 19 (2 min.)

Individuate tutti i file più grandi di 1MB presenti nell'intero sistema.

# Individuazione dei file con **`find`**

Per identificare i file a partire dalla loro dimensione si può usare il comando **`find`**.
Poiché è richiesta l'individuazione dei file più grandi di una specifica dimensione, si può usare l'opzione **`-size +SIZE`**. Nel caso specifico, la dimensione è 1MB.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **`/`**.
In definitiva, il comando richiesto è il seguente:

```
find / -size +1M
```

# Esercizio 20 (3 min.)

Producete un elenco di tutti i file nel sistema con relativa dimensione:

```
/file1 dimensione1
/file2 dimensione2
...
```

# Identificazione dei file con **`find`**

Per identificare i file a partire dalla loro dimensione si può usare il comando **`find`**.
Poiché è richiesta la stampa di un insieme specifico di metadati, si può usare l'opzione **`-printf`**. Nel caso specifico, si richiedono il nome del file e la dimensione; pertanto, la stringa di formato richiesta è **`"%p %s\n"`**.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **`/`**.
In definitiva, il comando richiesto è il seguente:

```
find / -printf "%p %s\n"
```

# Addendum

Alcuni file possono contenere, in maniera sciagurata, degli spazi. In queste condizioni, il comando scritto in precedenza non funziona, poiché verrebbe considerata come chiave numerica una porzione del file!
Per evitare questo problema, si può invertire la posizione del nome del file e della sua dimensione, per poi ordinare sulla prima chiave:

```
find / -printf "%s %p\n"
```

# Esercizio 21 (2 min.)

Individuate tutti i file che verificano una delle proprietà seguenti:

- terminano con **`.bak`**;
- terminano con **`~`**.

Cancellate forzatamente i file individuati.

# Scrittura della espressione regolare

I file che terminano con **`.bak`** sono individuabili con l'espressione regolare seguente:

```
^.*\.bak$
```

I file che terminano con **`~`** sono individuabili con l'espressione regolare seguente:

```
^.*~$
```

I file che terminano con **`.bak`** o **`~`** sono individuabili con l'espressione regolare seguente:

```
^.*\(\.bak\|~\)$
```

# Individuazione dei file con **find**

Per identificare i file a partire dalla loro estensione si può usare il comando **find**.
Poiché è richiesta l'individuazione dei file a partire da una espressione regolare, si può usare l'opzione **-regex "PATTERN"**.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **/**.
In definitiva, il comando richiesto è il seguente:

```
find / -regex "^.*\(\.bak\|~\)$"
```

# Cancellazione forzata dei file

Per cancellare forzatamente i file si può eseguire l'azione **`-exec`** di **`find`** che esegue un'applicazione su ogni match. La cancellazione si ottiene con l'esecuzione del comando **`rm -f MATCH`**:

```
-exec rm -f '{}' \;
```

In definitiva, il comando richiesto è il seguente:

```
find / -regex "^.*\(\.bak\|~\)$" -exec
   rm -f '{}' \;
```

# Esercizio 22 (4 min.)

Stampate i valori di tutte le etichette **`UUID`** nel file **`/etc/fstab`**.
Se possibile, evitate di stampare la stringa **`UUID=`**.

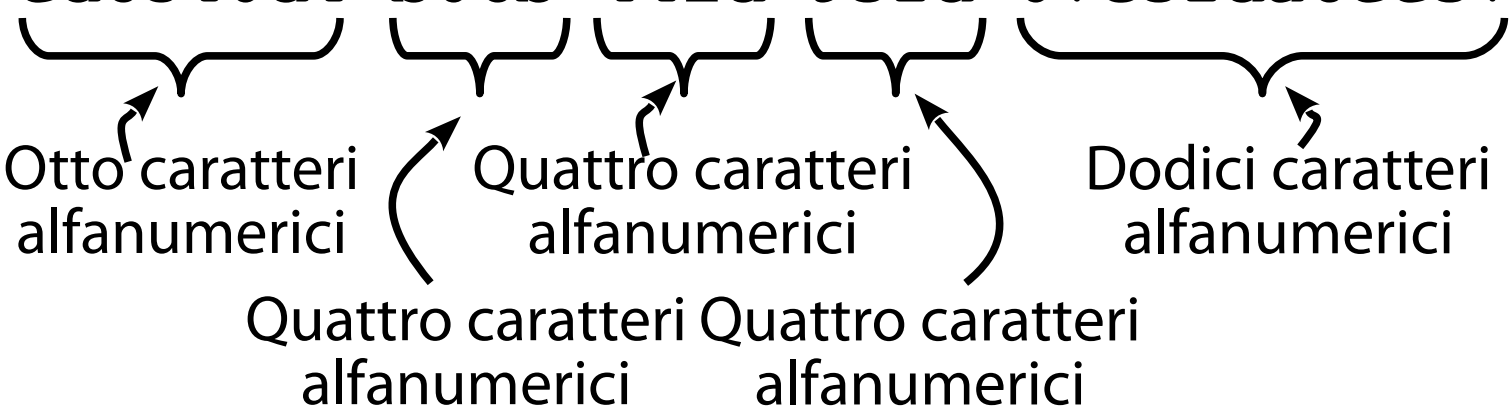
# Studio del file `/etc/fstab`

Si stampa innanzitutto il contenuto del file **`/etc/fstab`**, al fine di capire il formato delle stringhe UUID.

```
cat /etc/fstab
```

Una stringa UUID ha il formato seguente:

**`UUID=ed6540d4-b06b-442d-93fd-07c3fda6ee37`**



# Scrittura della espressione regolare

L'espressione regolare seguente effettua il match del primo blocco di caratteri alfanumerici:

```
^UUID=[[:alnum:]]{8}
```

L'espressione regolare seguente effettua anche il match del secondo blocco di caratteri alfanumerici:

```
^UUID=[[:alnum:]]{8}-[[:alnum:]]{4}
```

Esistono tre blocchi di quattro caratteri consecutivi, separati da –; l'espressione regolare seguente li cattura:

```
^UUID=[[:alnum:]]{8}-([[:alnum:]]{4}-){3}
```

# Scrittura della espressione regolare

L'espressione regolare seguente cattura anche l'ultimo blocco di dodici caratteri alfanumerici:

```
^UUID=[[:alnum:]]{8}-([[:alnum:]]{4}-){3}
[[:alnum:]]{12}
```

# Identificazione UUID con **grep**

Per identificare e stampare contenuti in un file si può usare il comando **grep**.
Poiché è richiesta l'individuazione di contenuti tramite una espressione regolare estesa, si può usare l'opzione -E **"REGEX"**. Si può rimuovere la stringa **UUID=** dalla espressione regolare precedente per stampare solo il valore di UUID.
Per stampare solo il match, si può usare l'opzione **-o**.
In definitiva, il comando richiesto è il seguente:

```
grep -o -E "[[:alnum:]]{8}-([[:alnum:]]{4}-){3}[[:alnum:]]{12}" /etc/fstab
```

# Esercizio 23 (2 min.)

Create un collegamento fisico di nome **`passwd`** al file **`/etc/passwd`**. Ci riuscite?
Create un collegamento simbolico di nome **`passwd`** al file **`/etc/passwd`**. Ci riuscite?

# Creazione collegamento fisico

Si crea un collegamento fisico al file **`/etc/passwd`**:

```
ln /etc/passwd
```

Si ottiene un messaggio di errore del tipo “permesso negato”. Non è possibile creare collegamenti fisici a file posseduti da altri utenti.

# Creazione collegamento simbolico

Si crea un collegamento simbolico al file **`/etc/passwd`**:

```
ln -s /etc/passwd
```

Il collegamento simbolico è stato creato correttamente; lo si può visualizzare con il comando seguente:

```
ls -l passwd
```

.

# Esercizio 24 (4 min.)

Create tre archivi della vostra home directory:

- **`/tmp/home.tar.gz`**
- **`/tmp/home.tar.bz2`**
- **`/tmp/home.tar.xz`**

Come variano le dimensioni degli archivi?

# Creazione primo archivio con **`tar`**

Per creare un archivio si usa il comando **`tar`** con le opzioni **`-f ARCHIVE_NAME`** e **`-c`**.
Per comprimere l'archivio in formato GZIP si usa l'opzione **`-z`**.

In definitiva, il comando richiesto è il seguente:

```
tar -z -f /tmp/home.tar.gz -c /home/studente
```

# Creazione secondo archivio con **`tar`**

Per creare un archivio si usa il comando **`tar`** con le opzioni **`-f ARCHIVE_NAME`** e **`-c`**.
Per comprimere l'archivio in formato BZIP2 si usa l'opzione **`-j`**.

In definitiva, il comando richiesto è il seguente:

```
tar -j -f /tmp/home.tar.bz2 -c /home/studente
```

# Creazione secondo archivio con **`tar`**

Per creare un archivio si usa il comando **`tar`** con le opzioni **`-f ARCHIVE_NAME`** e **`-c`**.
Per comprimere l'archivio in formato XZ si usa l'opzione **`-J`**.

In definitiva, il comando richiesto è il seguente:

```
tar -J -f /tmp/home.tar.xz -c /home/studente
```

# Analisi dimensioni archivi

Le dimensioni degli archivi sono recuperabili con il comando **`ls -l`**:

```
ls -l /tmp/home.tar.gz
ls -l /tmp/home.tar.bz2
ls -l /tmp/home.tar.xz
```

In generale XZ comprime di più, GZ comprime di meno, BZIP2 è una via di mezzo.

# Esercizio 25 (2 min.)

Create un archivio della vostra home directory:

**`/home/studente/home.7z`**

Cifrate l'archivio con la chiave simmetrica **`secret`**.

(Nella speranza che l'archivio non sia gigantesco) Estraete l'archivio nella directory **`/tmp`**.

# Creazione archivio cifrato con **7z**

Per creare un archivio in formato 7-ZIP si usa il comando **7z** con il sottocomando **a**.
Per cifrare l'archivio si abilita la cifratura con l'opzione **-mhe=on**.
Per specificare la chiave di cifratura **secret** si usa l'opzione **-psecret**.

In definitiva, il primo comando richiesto è il seguente:

```
7z -mhe=on -psecret a ~/home.7z ~
```

# Estrazione archivio cifrato con **7z**

Per estrarre l'archivio nella directory **/tmp** ci si sposta preventivamente lì:

```
cd /tmp
```

Per estrarre l'archivio in formato 7-ZIP si usa il comando **7z** con il sottocomando **x**.
Si immette la chiave **secret** al prompt.

In definitiva, il primo comando richiesto è il seguente:

```
7z x ~/home.7z
```

# Esercizio 26 (1 min.)

Testate la leggibilità e la scrivibilità del file seguente:

**`/etc/shadow`**

Riuscite ad accedere al file in qualche modo?

# Test di leggibilità con il builtin **`test`**

Si verifica la leggibilità di **`/etc/shadow`**:

```
test -r /etc/shadow
```

Si stampa l'esito del test:

```
echo $?
```

Si ottiene l'output seguente:

```
1
```

→ Il file **`/etc/shadow`** non è leggibile da **`studente`**.

# Test di scrivibilità con il builtin **`test`**

Si verifica la scrivibilità di **`/etc/shadow`**:

```
test -w /etc/shadow
```

Si stampa l'esito del test:

```
echo $?
```

Si ottiene l'output seguente:

```
1
```

→ Il file **`/etc/shadow`** non è scrivibile da **`studente`**.

# Esercizio 27 (2 min.)

Producete un elenco di tutti i file nel sistema con relativa dimensione:

```
/file1 dimensione1
/file2 dimensione2
...
```

Salvate l'output nel comando **`out.txt`**.
Scartate i messaggi di errore.

# Identificazione dei file con **`find`**

Per identificare i file a partire dalla loro dimensione si può usare il comando **`find`**.
Poiché è richiesta la stampa di un insieme specifico di metadati, si può usare l'opzione **`-printf`**. Nel caso specifico, si richiedono la dimensione e il nome del file; pertanto, la stringa di formato richiesta è **`"%s %p\n"`**.
Per trovare tutti i file nel sistema (compatibilmente con i diritti di accesso agli stessi), si può far partire la ricerca dalla directory radice **`/`**.

```
find / -printf "%s %p\n"
```

# Redirezione di STDOUT e STDERR

Per salvare l'output del comando nel file **`out.txt`**, si redireziona il canale STDOUT su tale file.

```
find / -printf "%s %p\n" > out.txt
```

Per scartare i messaggi di errore, si redireziona il canale STDERR nel file speciale di dispositivo **`/dev/null`**.

```
find / -printf "%s %p\n" > out.txt
  2> /dev/null
```

# Esercizio 28 (3 min.)

Usando i comandi interni **`exec`** e **`read`**, copiate le prime cinque righe di **`/etc/passwd`** nel file **`passwd-top5.txt`**.

# Apertura dei file

Per poter leggere dal file **`/etc/passwd`**, bisogna aprire tale file in lettura:

```
exec 3< /etc/passwd
```

Per poter scrivere sul file **`passwd-top5.txt`**, bisogna aprire tale file in scrittura, nella modalità "append":

```
exec 4>> passwd-top5.txt
```

# Lettura e scrittura delle righe

Per scrivere le prime cinque righe di **`/etc/passwd`** nel file si può usare un ciclo for con i builtin **`read`** e **`echo`**:

```
for i in 1 2 3 4 5; do
   read -u 3 line
   echo $line >&4
done
```

# Chiusura dei file

Al termine delle operazioni di lettura e scrittura, si chiudono i file **`/etc/passwd`** e **`passwd-top5.txt`**:

```
exec 3<&-
exec 4>&-
```

# Esercizio 29 (5 min.)

Individuate i dieci file più grandi all'interno della vostra home directory.

# Scomposizione del task in sotto-task

L'esercizio si può scomporre in tre parti:

- creazione di un elenco di file con relative dimensioni;
- ordinamento decrescente delle righe nell'elenco tramite la chiave "dimensione";
- stampa dei primi dieci file dall'elenco ordinato.

La prima parte si risolve come nell'Esercizio 27.
La seconda parte si risolve tramite il comando **`sort`**.
La terza parte si risolve tramite il comando **`head`**.

# Soluzione primo task

Occorre produrre un elenco di file e directory all'interno della propria home directory, con le relative dimensioni.

```
find $HOME -printf "%s %p\n"
```

Si dirotta l'output su un file **tmp1.txt**.

```
find $HOME -printf "%s %p\n" > tmp1.txt
```

Si scartano gli errori, dirottandoli su **/dev/null**.

```
find $HOME -printf "%s %p\n" > tmp1.txt
2>/dev/null
```

# Soluzione secondo task

Occorre ordinare numericamente ed in senso decrescente l'elenco prodotto nel task precedente, usando come chiave il primo campo (dimensione file).

```
sort -n -r -k 1 tmp1.txt
```

Si dirotta l'output su un file **`tmp2.txt`**.

```
sort -n -r -k 1 tmp1.txt > tmp2.txt
```

Di solito **`sort`** non genera errori; si può decidere di non scartare **`STDERR`**.

# Soluzione terzo task

Occorre stampare le prime dieci righe dell'elenco prodotto nel task precedente:

```
head tmp2.txt
```

# Esercizio 30 (1 min.)

Individuate tutti i file in **`/etc`** che terminano con l'estensione **`.txt`**.

# Identificazione dei file con **`find`**

Si usa l'opzione **`-name`** di **`find`** per identificare i file in base al pattern **`*.txt`**.

Il pattern **`*.txt`** va quotato per evitare l'espansione nei file terminanti con **`.txt`** contenuti nella directory corrente.

```
find /etc -name "*.txt"
```