Comandi

ls

opzioni
- -a | --all -> mostra anche file nascosti
- -n -> visualizzazione ID utente e ID gruppo invece di user esteso
- -l -> visualizzazione estesa, il timestamp riguarda l'ultima modifica del file (mtime)
  - -c -> timestampo dell'ultima modifica dei metadati (ctime)
  - -u -> timestamp dell'ultimo accesso (atime)

chmod mode[, mode...] filename
Si hanno due possibili modi:
- forma ottale
- forma simbolica

forma ottale (1->x, 2->w, 4->r)

6 = 110 = r w x

quando si usano 4 numeri, il primo fa riferimento ai bit speciale

1->sticky, 2->setgid, 4->setuid

forma simbolica
[+-=] [r,w,x] -> aggiunge/toglie a tutti il/i permesso/i scelti basandosi sulla umask (permessi alla creazione)

[ugoa] [+-=] [perm] -> per un determinato insieme (user, group, other, all). Non si basa su umask

perm può essere:
- r,w,x
- X -> aggiungi X se ha senso renderlo eseguibile
- u,g,o -> aggiunge all'insieme specificato i permessi di un altro insieme

L'opzione -R applica i permessi a tutte le sottodirectory (solo root)

umask [mode]
Diritti di accesso ai file/directory al momento della creazione
mode è in numeri ottali. Si parte da 666 per file o 777 per directory e si sottrae mode

cp [-r] [-i] [-a] [-u] {file_src} file_dest
opzioni
- -r -> ricorsivo, usato per directory (può perdere attributi e timestamp)
- -i -> interactive, avviso in caso di sovrascrittura
- -u -> sovrascrittura solo se mtime sorgente maggiore mtime destinazione
- -a -> copia ricorsiva che preserva attributi e metadati

mv [-i] [-u] [-f] {file_src} file_dest
opzioni
- -f -> force (di default)

rm [-f] [-i] [-r] {file}

ln [-s] src [dest]
Per creare symlink e hardlink. Con -s diventano soft

touch [-a] [-m] [-t timestamp] {file}
Creare file o modificare timestamp

opzioni:
-a -> modifica atime
-m -> modifica mtime
-t -> utilizza timestamp invece del corrente, formato [[CC][YY]MMDDhhmm[.ss]

du [-c] [-s] [-a] [-h] [--exclude=PATTERN] [files]

Questo comando stima lo spazio occupato del/dei file e/o directories dati in input. Vediamo ora le opzioni:

- `-c` → restituisce la somma del totale (di tutti i file passati in input)
- `-s` → non mostra eventuali sottodirectory o file contenuti
- `-a` → ricorsivo sulle directories in essa contenuta (valido solo per directory)
- `-h` → restituisce i risultati in formato leggibile (e.g., 1K 234M 2G)

df [-h] [-l] [-i] [file]

Restituisce un report sullo spazio libero. Vediamo le opzioni:

- `-h` → restituisce i risultati in formato leggibile (e.g., 1K 234M 2G)
- `-l` → mostra solo i filesystem locali (esclude quelli remoti come NFS o montaggi di rete)
- `-i` → restituisce le informazioni riguardanti gli inodi invece di quelle rispetto ai blocchi
- `file` → restituisce le informazioni rispetto al filesystem in cui è memorizzato il `file`

dd [opzioni]

Serve per copiare un file in modo elaborato e le opzioni sono una sequenza di `var=value`
Vediamo le opzioni:

- `bs=BYTES` → legge e scrive fino a `BYTES` bytes alla volta (default 512)
- `count=N` → copia solamente `N` blocchi di input
- `skip=N` → salta i primi `N` blocchi di input
- `seek=N` → salta i primi `N` blocchi di output
- `conv=CONVS` → converte il file come da comma-separated `CONVS` in input (es. `ucase`, `lcase`, `noerror`, ...)
- `if=FILE` → leggi da `FILE` invece di `stdin` (`cat file | dd of=output_file bs=1M`) viene usato `/dev/zero` quando bisogna creare un file con soli zero
- `of=FILE` → scrivi su `FILE` invece che su `stdout`

Oltre che per le conversioni si usa per copiare file speciali che non possono essere copiati con `cp`

mkfs [-t type fs-options] device

Questo è un comando fontend (non gestiamo davvero la creazione di un filesystem) serve per creare un filesystem Linux su un `device`, tipicamente una partizione su disco rigido.
Vediamo le opzioni:

- `-t type` → specifica il filesystem da usare tra questi (default `ext2`)
- `fs-options` → opzioni specifiche del filesystem da passare al vero costruttore del filesystem (es. `ro` read-only, `rw` read-write)
- `device` → può essere sia il nome del dispositivo (es. `/dev/hda1/`, `/dev/sdb2`) ma anche un file

jobs [-l] [-p]
Eleco dei job nella sessione corrente. Se un job è composto da più task la shell li gestisce come un unico job, ma ogni comando nella pipeline è un processo distinto
opzioni
-l -> info aggiuntive sui job
-p -> mostra solo il pid di ogni job (non lo stato)

bg e fg
il comando bg permette di portare un processo in background (lancio il processo; lo interrompo con ctrl+z; lo risveglio con bg)
il comando fg permette di portare un processo in foreground
in entrambi i comandi si può specificare un job tramite
- [prefix] -> parte iniziale del job desiderato
- [job_id]

- [+] oppure [%] → ultimo job mandato
- [-] → penultimo job mandato

ps [opzioni] [pid...]

Il comando `ps` mostra le informazioni riguardo una selezione dei processi attivi (se si vuole un aggiornamento continua della selezione e le informazioni mostrare usare `top`)
Legge le informazioni dai file virtuali in `/proc`

`ps` senza argomenti mostra i processi dell'utente attuale lanciati dalla shell corrente. Per ognuno di essi mostra `PID`, `TTY`, `TIME` (tempo totale di esecuzione) e `CMD`

Vediamo ora le opzioni disponibili:

- `-e` → tutti i processi di tutti gli utenti lanciati da tutte le shell o al boot (figli del processo 0)
- `-u {user}` → tutti i processi degli utenti nella lista in input
- `-p {pid}` → tutti i processi con i PID nella lista
- `-f` → restituisce in output delle colonne addizionali quali `UID`, `PPID`, `C` (fattore di utilizzo della CPU, da 0 a 99) e `STIME` (tempo di avvio)
- `-l` → altre colonne addizionali quali `F` (flag), `PRI` (priorità del processo, più basso → più alta priorità), `NI` (nice value, influenza la priorità), `ADDR` (indirizzo di memoria del processo), `SZ` (dimensione dell'immagine del processo in pagine), `WCHAN` (indirizza la funzione in cui il processo è in attesa, se dormiente)
- `-o {fields}` → per scegliere i campi da visualizzare
- `-C {cmds}` → mostra solo i processi il cui nome eseguibile è in `{cmds}`

Ci stanno anche i campi `RUSER` per il reale utente che ha avviato il processo e `EUSER` che corrisponde all'utente che ha eseguito il processo

ps -l

```
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   501     288       8  0  80   0 -  2648 do_wai pts/1    00:00:00 bash
0 R   501     310     288  0  80   0 -  2793 -      pts/1    00:00:00 ps
```

In ordine:

- `F` → flags associati al processo: 1 il processo e stato "forkato", ma ancora non eseguito; 4, ha usato privilegi da superutente; 5, entrambi i precedenti; 0, nessuno dei precedenti (`-y -l` elimina questo campo che e' poco utile)
- `S` → stato (modalita) del processo in una sola lettera
- `UID` → utente che ha lanciato il processo (se SetUID presente pottrebbe non essere chi ha dato il comando)
- `PID` → process id, identificatore del processo
- `PPID` → parent pid, pid del processo che ha creato questo processo
- `C` → parte intera della percentuale di uso della CPU
- `PRI` → attuale priorita del processo (piu il numero e alto, minore e la priorita)
- `NI` → valore di nice, da aggiungere alla priorita' (vedere piu avanti)
- `ADDR` → indirizzo in memoria del processo, ma e mostrato (senza valore) solo per compatibilita all'indietro (`-y -l` toglie questo campo e lo sostituisce con RSS - resident set size - dimensione del processo in memoria principale in KB - non tiene conto delle pagine su disco)
- `SZ` → dimensione totale attuale del processo in numero di pagine (tutte le 6 aree di memoria del processo) sia in memoria che su disco (memoria virtuale)
- `WCHAN` → se il processo e in attesa di un qualche segnale o comunque in sleep, qui c'e la funzione del kernel all'interno della quale si e fermato
- `TTY` → rappresenta il nome del terminale da cui è stato avviato il processo
- `TIME` → tempo di CPU usato finora
- `CMD` → comando con argomenti

ps -f

```
UID          PID    PPID  C STIME TTY          TIME CMD
fla          288       8  0 18:29 pts/1    00:00:00 -bash
fla          322     288  0 18:43 pts/1    00:00:00 ps -f
```

L'unica differenza è data da `STIME` (o `START`) che indica l'ora in cui e stato invocato il comando, oppure la data, se e stato fatto partire da più di un giorno

top [-b] [-n num] [-p {pids}]

Permette di avere ps, ma sempre aggiornato. Una volta aperto si usa ? per vedere i comandi disponibili

opzioni:
-b → non accetta comandi interattivi, si mantiene aggiornato (batch)
-n num → fa solo num refresh
-p {pids} → come in ps

kill [-l [signal]] [-signal] [pid ...]
Invia un segnale a un processo. Il segnale è identificato da un numero o da un nome (con o senza SIG)
opzioni
-l [signal] → lista i segnali; specificando signal converte il numero in nome o viceversa
es kill -9 pid, kill -KILL pid, kill -SIGKILL pid
I segnali verranno presi in considerazione solo se il real user del processo è lo stesso che invia il segnale (o su)
Al segnale verrà eseguita un'azione predefinita oppure se ne può definire una personalizzata (non su SIGKILL e SIGSTOP)
È possibile usare la notazione con % (di bg e fg) per indicare i job destinatari del messaggio

alcuni segnali
- SIGSTOP → sospensione del processo (ctrl+z)
- SIGCONT → continuazione dei processi stoppati (bg, fg)
- SIGKILL, SIGINT → terminazione dei processi (ctrl+c)

SIGUSR1 e SIGUSR2
Consentono una semplice forma di comunicazione tra processi

nice [-n num] [command]
da solo restituisce la niceness di partenza; ovvero indica quanto "gentile" sarà il processo nel contendersi la CPU: più il numero è alto, più cede volentieri tempo-CPU (da -19 a +20)
opzioni:
- [command] → esegue command con niceness uguale a num (0 se non specificato)

renice priority {pid}
Permette di modificare la priorità di un processo in esecuzione

strace [-p pid] [command] [-o file]
Esegue command mostrando le syscall oppure visualizza le syscall del processo pid
È possibile specificare un file di output

Permessi di speciali

Se uno dei bit speciali è attivo ma non si ha il corrispondente permesso di esecuzione, la lettera (s o t) sarà maiuscola

Sticky bit (t nella sezione other)

Si può cancellare un file solo se si hanno i permessi di scrittura sul file. Senza è sufficiente avere i permessi di scrittura sulla directory (con sticky necessari entrambi). Viene impostato sulla directory che contiene il file (non sul file stesso)

Setuid bit (solo per file eseguibili) (s nella sezione user)

File viene eseguito con i permessi del proprietario del file, anche se chi lo esegue ha meno permessi

Setgid bit (s nella sezione group)

Uguale a setuid ma per i gruppi. Può anche essere applicato ad una directory, così ogni file creato all'interno ha il gruppo della directory invece che quello di chi ha creato il file