

La shell di Unix

Sistemi di Elaborazione – Programmazione di sistema in ambiente Unix

Anno Accademico 2023/2024

Ing. Giovanni Nardini



UNIVERSITÀ DI PISA

Comandi di base della shell

Ing. Giovanni Nardini - University of Pisa - All rights reserved

Prima o poi, vi troverete a muovervi all'interno della shell di un sistema Unix o Unix-like:

- Alcune funzionalità del sistema non sono accessibili dalla GUI
- La GUI non è disponibile (e.g. accesso a un server remoto)
- Più veloce e rende le operazioni più efficienti

Accesso al sistema

- La shell (o la GUI) presenta il prompt per effettuare il login
 - Inserimento di username e password
- Ci si trova nella directory HOME dell'utente
 - e.g., `/home/studente`

\$ logout

- Esce dalla sessione corrente
- Presenta di nuovo il prompt per effettuare il login

\$ shutdown -h now Arresta il sistema

\$ shutdown -r now Riavvia il sistema

La directory HOME dell'utente è la porzione del file system dedicata ai file personali dell'utente, dentro la quale di solito l'utente si limita a lavorare.

Il comando shutdown permette di spegnere (-h) o riavviare (-r) la macchina a un tempo prefissato. Può essere un'orario specifico (e.g., 10:18), un intervallo di tempo a partire dall'istante corrente (e.g., +10, per spegnere fra 10 minuti), oppure la stringa now (per spegnere subito)

Prompt della shell Bash

The diagram shows a Bash prompt string: `studente@debian-SdE:~/Documents$`. Below the prompt, four colored horizontal bars are positioned: a green bar under 'studente', a yellow bar under '@debian-SdE', a grey bar under '~/Documents', and a red bar under '\$'. Arrows point from these bars to labels below: 'username' (green), 'hostname' (yellow), 'directory corrente' (grey), and 'cursore' (red).

Il prompt ci mostra diverse informazioni utili:

- Nome utente attualmente loggato nel sistema;
- Il nome del calcolatore, utile soprattutto quando il calcolatore è connesso a una rete;
- Il percorso della directory corrente (ricorda: ~ indica la home dell'utente loggato);
- Il cursore, dopo il carattere \$, indica dove viene digitato il comando.

Utilizzo della shell

- Sintassi generale

\$ comando [opzione] [argomento_opzionale] argomento

- Non tutti i comandi hanno bisogno di opzioni e/o argomento
- Nel seguito (così come nel manuale del S.O.) le parentesi quadre [] indicano un'opzione o un argomento facoltativo

cd

cd [directory]

- **change directory**: permette di spostarsi nella directory specificata dall'argomento

\$ **cd /home/studente/Documents** Usando percorso assoluto

\$ **cd Documents** oppure **cd ../Documents** Usando percorso relativo

\$ **cd** Senza argomenti, ci si sposta nella home dell'utente → equivalente a **cd ~**

\$ **cd ..** Ci si sposta nella directory parent, rispetto a quella corrente

\$ **cd ../../** Ci si sposta nella directory parent della directory parent

Come detto nel blocco precedente, i percorsi di un file si specificano con i nome delle directory separate da /, iniziano dalla root

Ci sono dei caratteri speciali che indicano:

- La HOME
- La directory precedente
- La directory corrente

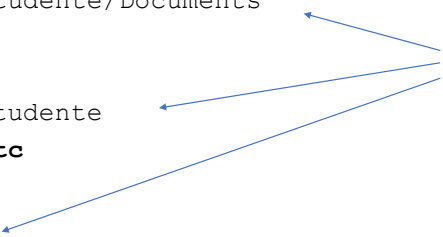
pwd

pwd

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

```
$ pwd
/home/studente/Documents
$ cd ..
$ pwd
/home/studente
$ cd /etc
$ pwd
/etc
```

Output



ls

ls [opzioni] [directory1] [directory2] ...

- **list**: stampa l'elenco dei file contenuti nelle directory specificate
- Se non si specifica alcuna directory, elenca il contenuto della directory corrente

```
$ ls
```

```
$ ls /home/studente/Documents
```

```
$ ls Documents
```

```
$ ls ./Documents ./Pictures
```


ls

```
ls [opzioni] [directory1] [directory2] ...
```

Opzioni:

```
$ ls -l
```

- Mostra i dettagli di ogni file (proprietario, ultima modifica, ecc.)

```
$ ls -a
```

- Mostra anche i file nascosti (il cui nome inizia con .)

```
$ ls -l -a oppure $ ls -la
```

- Le opzioni possono essere cumulate

Operazioni su file e directory

\$ mkdir test_dir

- Crea una nuova directory di nome `test_dir`

\$ rmdir test_dir

- Elimina la directory `test_dir`
 - Solo se vuota!

Operazioni su file e directory

\$ touch file1

- Aggiorna il tempo di ultimo accesso al file di nome `file1`
- Se il file non esiste, viene creato

\$ rm file1

- Rimuove `file1`
- Se `file1` è una directory, non viene rimossa

\$ rm -r file1

- Se `file1` è una directory, la rimuove insieme a tutto il suo contenuto

Eseguendo `touch` più volte, si vede che cambia il tempo di ultimo accesso al file

Operazioni su file e directory

\$ cp oldfile newfile

- Crea una copia di `oldfile` con nome `newfile`

\$ cp file1 file2 dir1

- Copia `file1` e `file2` nella directory `dir1`

\$ mv oldfile newfile

- Rinomina `oldfile` in `newfile`

\$ mv file1 file2 dir1

- Sposta `file1` e `file2` nella directory `dir1`

Wildcards

- Nella shell è possibile utilizzare dei *metacaratteri* (wildcards) per indicare un insieme di file o directory

- `?` sostituisce uno e un solo carattere
- `*` sostituisce zero o più caratteri
- `[aeiou]` sostituisce uno dei caratteri dell'insieme specificato
- `[a-z]` sostituisce uno dei caratteri all'interno dell'intervallo specificato

```
$ ls ing*.txt
```

- Elenca tutti i file nella directory corrente il cui nome inizia con `ing` e termina con `.txt`

```
$ rm ~/Documents/*.txt
```

- Rimuove tutti i file nella directory `Documents` che terminano con `.txt`

```
$ rm file?
```

- Rimuove tutti i file con nome `file`, seguito da *un* carattere qualsiasi

```
$ mv file[1-9] dest_dir
```

- Sposta tutti i file con nome `file`, seguito da *una* cifra compresa tra 1 e 9, in `dest_dir`

Strumento molto potente che velocizza le operazioni

Useful tips

- **Funzione di autocompletamento**
 - Durante la digitazione di un comando o del nome di un file, premere il tasto `TAB`
 - In caso di ambiguità, mostra le opzioni disponibili
- **Comandi precedenti**
 - Con i tasti freccia `↑` / `↓`, scorre la lista dei comandi digitati precedentemente
- **\$ history**
 - Mostra l'elenco dei comandi precedenti, ognuno con un numero di sequenza
 - Digitando `$!<numero_comando>` riesegue il comando scelto
- **\$ clear**
 - Pulisce la shell

Dalla directory `HOME`, prova a digitare i seguenti comandi:

- `ls Desk<TAB>` # completa il percorso
- `ls D<TAB><TAB>` # ambiguità da risolvere
- `ls d<TAB>` # aiuta fino a un certo punto

man

man [sezione] comando

- Probabilmente è il comando più importante di Unix!
- Visualizza la pagina del manuale di Unix relativa al comando specificato
- Stampa a video la descrizione del comando e specifica il significato di tutte le opzioni e gli argomenti di quel comando
- Esiste una pagina del manuale per spiegare sé stesso
 - `$ man man`

**Contiene anche la descrizione
di system calls e funzioni di
libreria!**



```
The table below shows the section numbers of the manual followed by the types of
pages they contain.
1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
```

Immaginate di dover programmare senza connessione a Internet, i.e. senza Google, Stackoverflow... il manuale di Unix vi dà una grossa mano

man

man [sezione] comando

\$ man ls

- Mostra la pagina del manuale per il comando `ls`

\$ man printf

- Mostra la pagina del manuale per il comando `printf`

\$ man 3 printf

- Mostra la pagina del manuale per la funzione C `printf` (sezione 3)

\$ whatis ls

- Mostra una descrizione breve del comando `ls`

\$ apropos zip

- Mostra tutti i comandi la cui pagina del manuale contiene la parola `zip`
 - Quali comandi posso usare per comprimere un file?

Operazioni di lettura e scrittura

Ing. Giovanni Nardini - University of Pisa - All rights reserved

Lettura di file

\$ cat file1

- Stampa sulla shell il contenuto di `file1`

\$ cat file1 file2

- Stampa sulla shell il contenuto di `file1`, seguito da `file2` (concatenazione)

\$ less file1

- Se `file1` è troppo lungo per essere visualizzato interamente sulla shell, questo comando ne mostra una parte, dando la possibilità di spostarsi interattivamente coi tasti freccia

\$ head file1

- Stampa le prime righe di `file1`
 - Default: 10 righe
 - Si può specificare il numero di righe con l'opzione `-n`

\$ tail file1

- Stampa le ultime righe di `file1`
 - Default: 10 righe
 - Si può specificare il numero di righe con l'opzione `-n`

Input/output

Ogni comando (e ogni processo) ha tre stream standard per input e output

- **stdin** → input da tastiera
- **stdout** → output su schermo
- **stderr** → messaggi di errore su schermo

È possibile:

- Prendere l'input da un'altra sorgente, e.g. un file
- Redirigere l'output verso un'altra destinazione, e.g. un file
- Redirigere i messaggi di errore verso un'altra destinazione, e.g. un file

Redirezione dell'output

```
$ ls /home/studente/Documents > doclist.txt
```

- Esegue il comando `ls` e scrive l'output sul file `doclist.txt`
 - Il contenuto del file `doclist.txt` viene sovrascritto!
 - Se il file non esiste, viene creato

```
$ ls /home/studente/Documents >> doclist.txt
```

- Uguale a `>`, ma l'output del comando viene scritto *in append* al contenuto del file

Redirezione dei messaggi di errore

```
$ ls /home/studente/Documents_err 2> errorlog.txt
```

- Esegue il comando `ls` e scrive l'output sul file `errorlog.txt`
 - Il contenuto del file `errorlog.txt` viene sovrascritto!
 - Se il file non esiste, viene creato

```
$ ls /home/studente/Documents_err 2>> errorlog.txt
```

- Uguale a `2>`, ma i messaggi di errore vengono scritti *in append* al contenuto del file

```
$ ls /home/studente/Documents &> log.txt
```

```
$ ls /home/studente/Documents &>> log.txt
```

- Redirige l'output e i messaggi di errore

Esempio di err

Redirezione dell'input

```
$ less < doclist.txt
```

- Prende l'input dal file, invece che dall'argomento del comando
- In questo caso, l'effetto è uguale a **\$ less doclist.txt**
- Tornerà utile nel caso di programmi più complessi
- Esempio:
 - Il programma deve ricevere l'input dell'utente da tastiera
 - Con la redirezione dell'input, il programma prende l'input da un file invece che attendere l'inserimento da tastiera dell'utente

Esempio: un programma che fa la somma di due numeri deve ricevere in input due numeri interi, inseriti dall'utente tramite la tastiera.

Con la redirezione dell'input, il programma legge i due numeri interi da un file invece di attendere l'inserimento da tastiera dell'utente.

input.txt:

2

3

```
$ ./somma < input.txt
```

5

Dispositivo /dev/null

- Talvolta, può essere utile «scartare» l'output o i messaggi di errore di un comando o di un processo
- `/dev/null` è un dispositivo fittizio
- Tutto è un file → possibile redirigere l'output o i messaggi di errore verso `/dev/null`

```
$ ls /home/studente/Documents > /dev/null
```

```
$ ls /home/studente/Documents 2> /dev/null
```

- `/dev/null` è come un «buco nero»
- Tutto ciò che viene rediretto verso `/dev/null` viene perso

`/dev` è la directory del file system dove sono contenuti i file che rappresentano i dispositivi e le periferiche connessi al calcolatore (e.g. disco rigido). Tali file sono astrazioni dei dispositivi con cui un utente può fare delle operazioni di I/O. Sono file speciali, su cui non posso effettuare operazioni di lettura e scrittura.

`/dev/null` è un dispositivo che non esiste fisicamente.

Esiste anche `/dev/zero`: può essere usato come sorgente illimitata di bytes con valore 0

Pipeline

- È possibile «concatenare» due o più comandi in modo che l'output del primo comando sia l'input del secondo, e così via

```
$ ls /home/studente/Documents | sort -r
```

- Il risultato di `ls` viene «inviato» al comando `sort -r`, che stampa in ordine decrescente il contenuto del file

- Alternativa:

```
$ ls /home/studente/Documents > doclist.txt
```

```
$ sort -r doclist.txt
```

L'alternativa richiede una scrittura sul disco e una lettura dal disco

echo

echo [stringa1] [stringa2] ...

- Stampa sulla shell il contenuto delle stringhe specificate

```
$ echo Hello world!
```

```
Hello world!
```

- A cosa serve?? → usato con redirectione dell'output, permette di scrivere/appendere dati in un file

```
$ echo Hello world! > helloworld.txt
```

```
$ cat helloworld.txt
```

```
Hello world!
```

```
$ echo This is a new line. >> helloworld.txt
```

```
$ cat helloworld.txt
```

```
Hello world!
```

```
This is a new line.
```

Il comando echo è un modo piuttosto grezzo di scrivere su un file, non permette nemmeno di aggiungere linee in un punto qualsiasi del file stesso

Editor di testo

- In base al tipo di ambiente grafico, ci sono diversi editor di testo
 - GNOME → gedit
 - KDE → kate
 - ...

\$ gedit

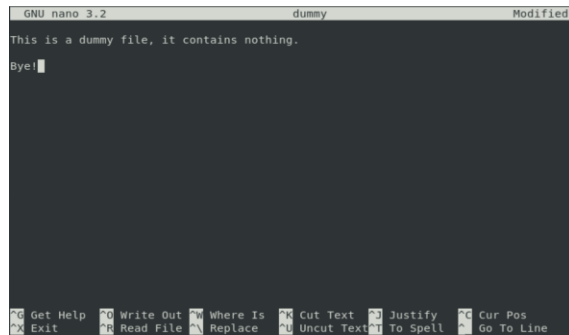
- Si apre una finestra (interfaccia grafica) dove poter scrivere il contenuto del file
- Esistono anche editor di testo da linea di comando
- Utili nel caso in cui il sistema operativo non abbia installato un ambiente grafico
 - nano
 - emacs
 - vi
 - vim

L'editor di testo è esso stesso un programma, e come tutti i programmi può essere lanciato dalla shell.

nano

\$ nano dummy

- Semplice
- Comandi elencati in fondo allo schermo
 - CTRL+O → Write file (save)
 - CTRL+X → Exit
 - ...

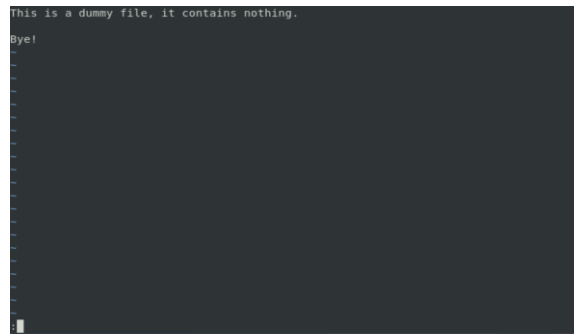


```
GNU nano 3.2 dummy Modified
This is a dummy file, it contains nothing.
Bye!
Ctrl+G Get Help  Ctrl+O Write Out  Ctrl+W Where Is  Ctrl+K Cut Text  Ctrl+J Justify  Ctrl+P Cur Pos
Ctrl+X Exit      Ctrl+R Read File  Ctrl+N Replace   Ctrl+U Uncut Text Ctrl+T To Spell  Ctrl+_ Go To Line
```

vi

\$ vi dummy

- Più funzionalità di nano
- Due modi di funzionamento
 - *Modalità inserimento* → come un qualsiasi editor di testo
 - *Modalità comandi* → permette di digitare e eseguire comandi
- Per passare alla modalità inserimento, premere **i**
- Per passare alla modalità comandi, premere **ESC**



vi

Alcuni comandi:

- `:w` salva modifiche
- `:wq` salva modifiche ed esci
- `:q` esci (solo se non sono state fatte modifiche)
- `:q!` esci senza salvare le modifiche
- `o` inserisce una nuova linea dopo la linea corrente
- `dd` cancella la linea corrente
- `yy` copia la linea corrente
- `p` incolla il contenuto copiato
- `/test` cerca nel testo la parola test

Archiviazione di file

tar [options] [files]



- **tape archive:** archivia/comprime/estrae file un insieme di file e/o directory
- Le opzioni specificano
 - La modalità in cui il comando deve operare:
 - archiviare o comprimere dei file
 - estrarre i file da un archivio esistente
 - mostrare il contenuto di un archivio esistente
 - Ulteriori dettagli su come eseguire l'operazione:
 - quale algoritmo di compressione utilizzare
 - livello di verbosità dell'output
 - nome dell'archivio

Archiviazione di file

```
$ tar -c -v -f archivio.tar ~/Documents/*
```

- Crea un archivio di nome `archivio.tar` contenente i file della directory `~/Documents`

```
$ tar -c -v -z -f archivio.tar.gz ~/Documents/*
```

- Crea un archivio di nome compresso `archivio.tar.gz` contenente i file della directory `~/Documents`
- Usa l'algoritmo *gzip* → estensione `tar.gz`

```
$ tar -c -v -j -f archivio.tar.bz2 ~/Documents/*
```

- Crea un archivio di nome compresso `archivio.tar.bz2` contenente i file della directory `~/Documents`
- Usa l'algoritmo *bzip2* → estensione `tar.bz2`

```
$ tar -x -v -f archivio.tar
```

- Estrai il contenuto di `archivio.tar`

- Possibile la forma abbreviata:

```
$ tar cvfz archivio.tar.gz ~/Documents/*
```

Gestione di utenti e gruppi

Ing. Giovanni Nardini - University of Pisa - All rights reserved

Utenti e gruppi

- Un sistema Unix è un sistema multiutente
- Ogni utente è caratterizzato da:
 - Username
 - Password
 - User ID (`uid`)
- I gruppi sono insiemi di utenti e sono caratterizzati da:
 - Group name
 - Password
 - Group ID (`gid`)
- Un utente *deve* appartenere ad almeno un gruppo → *primary group*

Utenti e gruppi

\$ id [username]

- Mostra uid, gid (del primary group) e gruppi a cui l'utente specificato appartiene
- Se username non è specificato, mostra le informazioni per l'utente corrente

```
studente@debian-SdE:~$ id root
uid=0(root) gid=0(root) groups=0(root)
studente@debian-SdE:~$ id studente
uid=1000(studente) gid=1000(studente) groups=1000(studente),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),113(bluetooth),117(lpadmin),120(scanner)
```

\$ groups [username]

- Mostra i gruppi a cui l'utente specificato appartiene
- Se username non è specificato, mostra le informazioni per l'utente corrente

```
studente@debian-SdE:~$ groups studente
studente : studente cdrom floppy sudo audio dip video plugdev netdev bluetooth lpadmin scanner
studente@debian-SdE:~$
```

Privilegi di root

- Utenti 'normali'
- Utente *root*
 - Amministratore del sistema
 - Ha privilegi speciali per compiere operazioni non permesse agli utenti normali

\$ **su username**

- Switch User
- Accedere al terminale dell'utente specificato
 - Se non specificato, accede al terminale dell'utente *root*
- Chiede di inserire la password dell'utente specificato

\$ **sudo command**

- Esegue il comando specificato con i privilegi di root
- Chiede di inserire la password dell'utente corrente (deve appartenere al gruppo *sudoers*)



Gestione degli utenti

\$ **adduser username**

- Crea un nuovo utente con lo username specificato
- Il comando richiede di inserire le informazioni per il nuovo utente

\$ **deluser username**

- Rimuove l'utente specificato


\$ **passwd**

- Cambia la password dell'utente corrente

\$ **passwd username**

- Cambia la password dell'utente specificato
- Solo l'utente con privilegi di root può cambiare la password di un altro utente

Solo utente con
privilegi di root



Gestione dei gruppi

\$ **addgroup groupname**


- Crea un nuovo gruppo con il nome specificato

\$ **delgroup groupname**

- Rimuove il gruppo specificato

\$ **newgrp groupname**

- Il gruppo specificato diventa temporaneamente il primary group per l'utente
- Per la sola sessione corrente (e.g., fino al logout)
- Necessario conoscere la password del gruppo



Solo utente con privilegi di root

Gestione dei gruppi

\$ gpasswd groupname

- Imposta la password del gruppo specificato
- Con password impostata, i membri di altri gruppi possono ottenerne temporaneamente i privilegi con il comando `newgrp`

\$ gpasswd -r groupname

- Rimuove la password per il gruppo specificato
- Con password non impostata, solo i membri del gruppo possono averne i privilegi

\$ gpasswd -a username groupname

- Aggiunge l'utente al gruppo specificato

\$ gpasswd -d username groupname

- Rimuove l'utente dal gruppo specificato

\$ gpasswd -M username1,username2,... groupname

- Specifica la lista dei membri del gruppo specificato

\$ gpasswd -A username1,username2,... groupname

- Specifica la lista degli amministratori del gruppo specificato
- Solo gli amministratori possono aggiungere/rimuovere utenti dal gruppo

Gestione dei permessi

- Il file system gestisce i *permessi di accesso* di utenti e gruppi a ogni singolo file

Read – r	Leggere il contenuto del file
Write – w	Modificare il contenuto del file
Execute – x	Eeguire il file

- Ogni file è caratterizzato da:
 - Utente proprietario del file
 - Gruppo proprietario del file
- I permessi sono differenziati per tre categorie di utenti:
 - Utente proprietario del file
 - Utenti appartenenti al gruppo proprietario del file
 - Tutti gli altri utenti

A cosa servono utenti e gruppi? A definire chi può fare determinate operazioni e accedere a un determinato file.

Per ciascuna categoria di utenti, possiamo definire se possono accedere al file, e in che modo (lettura/scrittura/esecuzione).

Permessi su un file

\$ ls -l

Utente
proprietario

Gruppo
proprietario

```
studente@debian-SdE:~$ ls -l dummy
-rwxr--r-- 1 studente studente 84 Sep 27 13:57 dummy
```

Permessi degli altri utenti

Permessi degli utenti del gruppo proprietario

Permessi dell'utente proprietario

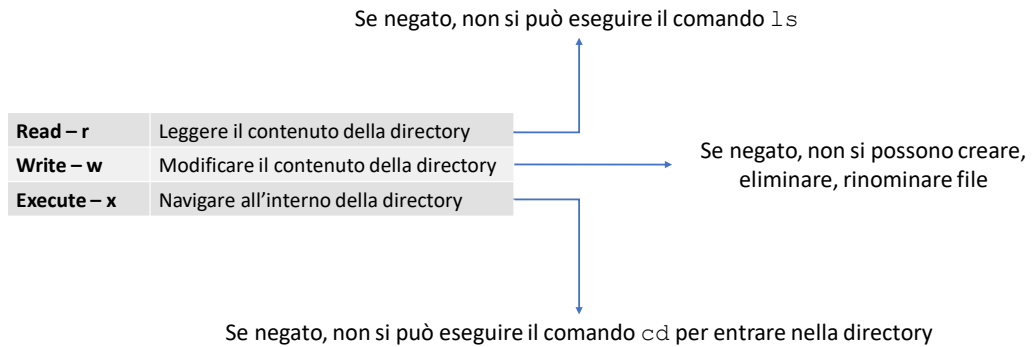
Nome file

Alcune shell mostrano i file di colore
diverso a seconda che siano file
eseguibili, directory, ecc.

Specifica il tipo di file → - = regular file; d = directory)

Permessi su una directory

- Le directory sono file → si possono specificare permessi per le directory



Manipolare i permessi

chmod [u|g|o] [+|-|=] [r|w|x] file

- **change file mode** bits: permette di cambiare i permessi di una o più classi di utenti relativi a un file
- [u|g|o]
 - Specifica la categoria di utenti a cui applicare il comando
 - u (*user*): utente proprietario; g (*group*): utenti del gruppo proprietario; o (*others*): tutti gli altri utenti
- [+|-|=]
 - Specifica se aggiungere (+), rimuovere (-) o assegnare (=) il permesso
- [r|w|x]
 - Specifica a quale permesso si riferisce il comando

Manipolare i permessi

\$ chmod u+rw file1

- Aggiunge tutti i permessi al proprietario su `file1`

\$ chmod go-w file1

- Rimuovi il permesso di modifica agli utenti del gruppo proprietario e a tutti gli altri
 - Gli altri permessi di `g` e `o` rimangono invariati
- Solo il proprietario potrà modificare `file1`

\$ chmod o=r file1

- Assegna solo permesso di lettura a utenti non proprietari
- Permessi di modifica e esecuzione vengono rimossi

\$ chmod -x privatedir

- Rimuovi a tutti il permesso di navigazione nella directory `privatedir`

```
studente@debian-SdE:~$ chmod -x privatedir/
studente@debian-SdE:~$ ls -l privatedir/
ls: cannot access 'privatedir/file1': Permission denied
ls: cannot access 'privatedir/file2': Permission denied
total 0
-???????? ? ? ? ? ? ? file1
-???????? ? ? ? ? ? ? file2
```

Rappresentazione ottale dei permessi

Osservazione

- Ogni gruppo di permessi è identificato da tre valori
- Ciascuno valore è on/off

Posso usare 3 bit per rappresentare i permessi di una classe di utenti

- --- 000 = 0
- --x 001 = 1
- ...
- r-x 101 = 5
- ...
- rwx 111 = 7



Una cifra in base 8

Metodo alternativo:

- r vale 4
- w vale 2
- x vale 1

La cifra in base 8 si ottiene sommando:

lettura + esecuzione $\rightarrow 4 + 1 = 5 \rightarrow 101$

I permessi di un file sono rappresentabili con *tre cifre in base otto*

Rappresentazione ottale dei permessi

```
$ chmod 744 file1
```

- Assegna tutti i permessi al proprietario, mentre assegna solo lettura ai membri del gruppo e a tutti gli altri

```
$ chmod 777 file1
```

- Assegna tutti i permessi a tutti su `file1`

Cambiare il proprietario di un file

\$ chown username file

- **change owner:** rende l'utente `<username>` proprietario del file specificato
- Solo con privilegi di root

\$ chgrp groupname file

- **change group:** rende il gruppo `<groupname>` proprietario del file specificato
- Solo con privilegi di root se l'utente corrente non appartiene a `<groupname>`

Permessi aggiuntivi

SUID: set user identification

- Permette al processo di acquisire i permessi del proprietario
- Nel campo relativo al permesso di esecuzione del proprietario, si indica *s* (invece che *x*)

SGID: set group identification

- Permette al processo di acquisire i permessi del gruppo proprietario
- Nel campo relativo al permesso di esecuzione del gruppo proprietario, si indica *s* (invece che *x*)

```
$ ls -l /usr/bin/passwd  
-rwsr-xr-x
```

Il programma *progA* deve accedere a un file *fileA* in lettura, *nel sistema ci sono due* utenti: Alice e Bob. Supponiamo anche che:

- *progA* e *fileA* abbiano come proprietario Alice
- sia Alice che Bob possano leggere ed eseguire il programma
- solo Alice possa leggere *fileA*

Se Bob esegue il programma, il programma non è in grado di accedere al file in lettura.

Con il permesso SUID su *progA*, quando Bob esegue il programma ottiene i permessi di Alice, dunque il programma può leggere il file.

Permessi aggiuntivi

- Si impostano con la rappresentazione ottale

```
$ chmod 4755 file1
```



Cifra aggiuntiva **prima** delle tre relative alle classi di utenti:

- | | | | |
|---------------------------|------|---|------------|
| • 4 per SUID | 4755 | → | -rwsr-xr-x |
| • 2 per SGID | 2755 | → | -rwxr-sr-x |
| • 6 per SUID e SGID (4+2) | 6755 | → | -rwsr-sr-x |

File di configurazione degli utenti

`/etc/passwd` → informazioni pubbliche sugli utenti del sistema

Diagram illustrating the fields of a user entry in the `/etc/passwd` file:

- username
- UID
- password
- GID
- dati aggiuntivi utente
- home directory
- shell

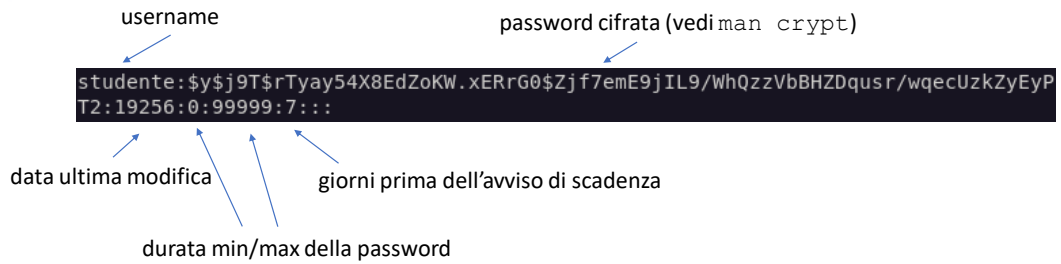
Example entry: `studente:x:1000:1000:studente,,,:/home/studente:/bin/bash`

- Il campo 'shell' può essere impostato al valore `/sbin/nologin` per inibire il login all'utente
- Il file `/etc/passwd` viene compilato quando si esegue il comando `adduser`
- Nota: `/etc` è la directory che contiene i file di configurazione del sistema

File di configurazione degli utenti

`/etc/shadow`

→ informazioni sensibili sugli utenti del sistema (e.g. password)



- Il file `/etc/shadow` viene modificato dal comando `passwd`

File di configurazione dei gruppi

/etc/group

→ informazioni pubbliche su gruppi del sistema

group name →

```
cdrom:x:24:studente
floppy:x:25:studente
tape:x:26:
sudo:x:27:studente
```

 ← membri del gruppo

password ← GID

/etc/gshadow

→ informazioni sensibili sui gruppi (e.g. password, amministratori)

group name →

```
cdrom:!:studente
floppy:!:studente
tape:!:
sudo:!:studente
```

 ← amministratori ← membri del gruppo

← Password cifrata (se * o !, non impostata)