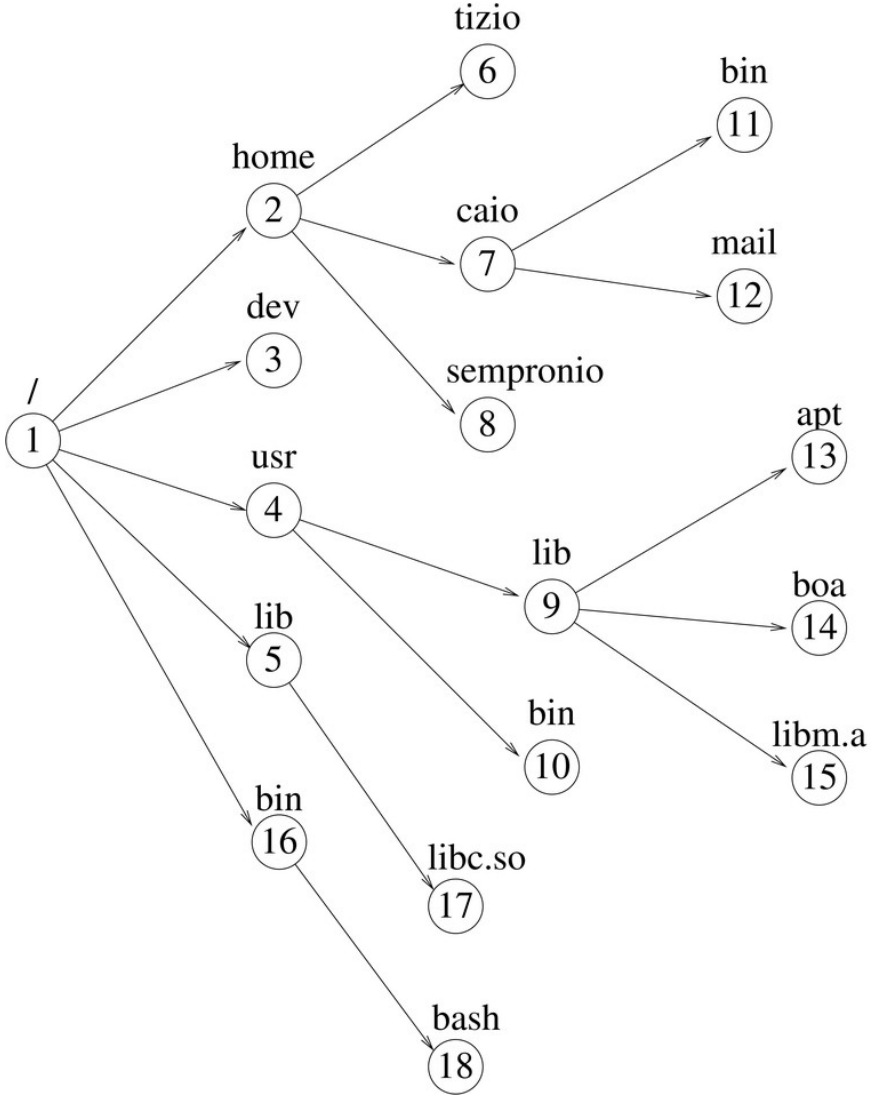
Informazioni utili

Per gli approfondimenti sui vari punti affrontati oggi verrete indirizzati alle pagine degli “Appunti di informatica libera” ospitati nel sito:

http://a2.informaticalibera.net/

Filesystem di Linux



I nomi di alcune directory hanno un significato ben preciso

Dubbi frequenti

- *Dove sono i dischi?*
- *I files sono sul disco?*
- *Posso mettere i files dove voglio?*

Linux Filesystem Hierarchy

Directory	Descrizione
/	La radice del filesystem;
/bin/	binari essenziali (cp, mv, bash);
/boot	file statici per l'avvio del sistema (lilo, grub);
/dev/	file di dispositivo (anche i devices sono trattati come files);
/etc/	configurazione del sistema e dei programmi;
/home/	directory personali degli utenti;
/lib/	librerie essenziali e moduli del kernel;
/mnt/	punto di innesto temporaneo di dischi, filesystem remoti, ecc;
/media/	unità rimovibili (gestite prevalentemente dai desktop manager)
/opt/ e /srv/	applicativi aggiuntivi (e relativi dati);
/root/	directory personale dell'utente root;
/tmp/	file e directory temporanei;
/usr/	file e directory dei programmi installati sul sistema;
/var/	dati variabili;
/proc/ /sys/	Informazioni realtime prodotte dal kernel;

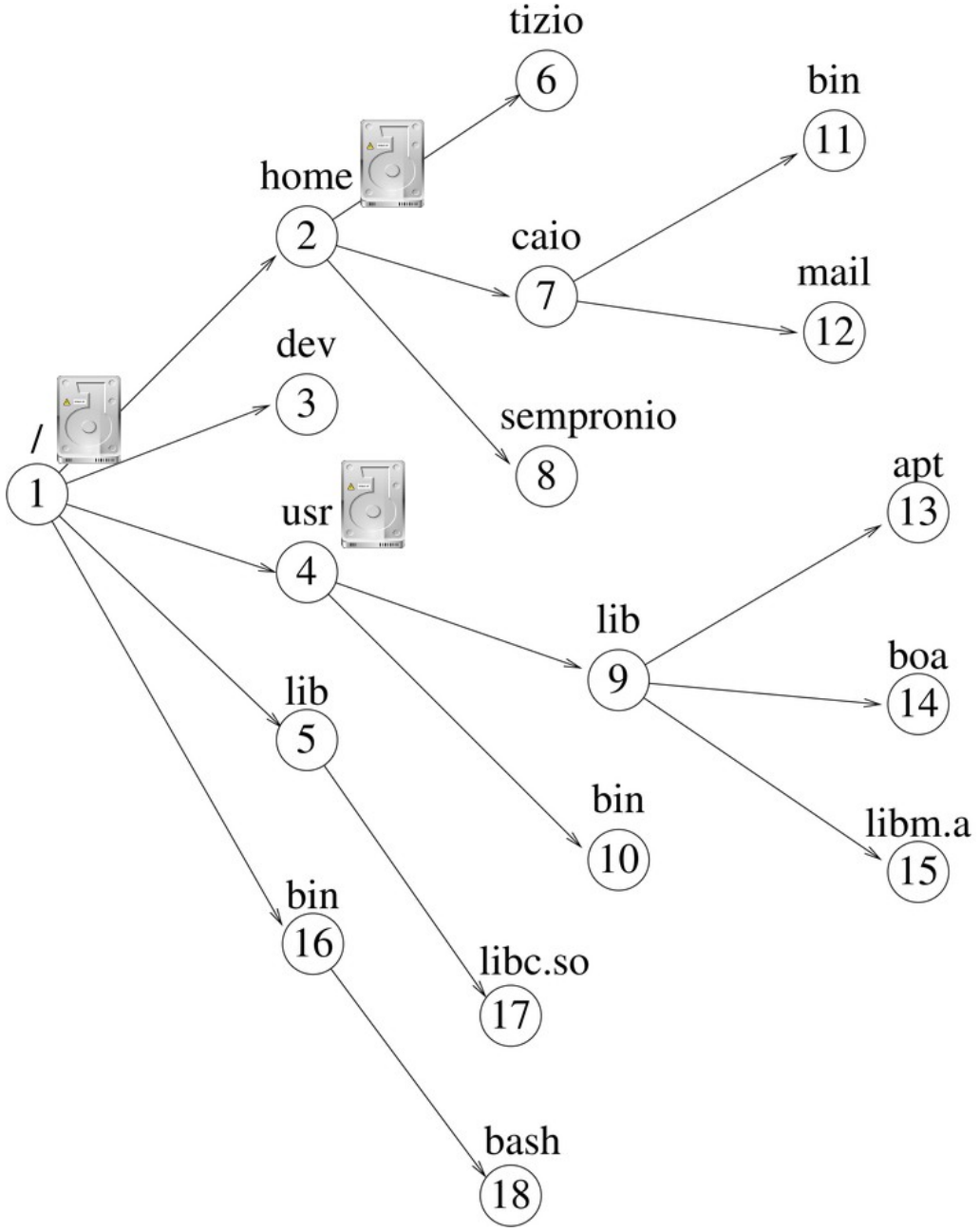
Linux Filesystem Hierarchy

Domanda: dobbiamo rispettare la gerarchia?

SI

- **Facilità di migrazione e recupero dati**
- **Mantenimento dell'integrità della distribuzione**
- **Sappiamo dove guardare quando siamo in cerca di qualcosa.**

Dischi e filesystem di Linux



- Ogni disco rappresenta una partizione.
- Una partizione risiede su disco
- Dividere il filesystem in più partizioni rende più robusto il sistema
- Le dimensioni delle partizioni che compongono il filesystem dipendono dal ruolo che deve svolgere il computer/server.

L'operazione di mount

- Il mount mette in collegamento un filesystem reale in una directory del filesystem di Linux

```
mount -t ext4 /dev/sda2 /home
```

- I filesystem possono essere virtuali, non realmente presenti su un dispositivo di memorizzazione di massa. (Esempio /sys).

Attenzione!!!

- Per filesystem Linux si intende la struttura virtuale di sistema che ingloba tutti i filesystem reali e virtuali su dischi e in memoria
- Ogni dispositivo dispone di un suo filesystem (esempio ext4, ntfs, nfs)
- Alcuni filesystem sono più o meno compatibili con lo standard posix (esempio supporto a proprietari e permessi)

Navigare nel filesystem

- cd: change directory → cambia la directory
  - cd /var/www
  - cd ../lib

  “ . e .. sono directory speciali e identificano la directory corrente e la directory padre”

- pwd: print work directory → mostra la directory corrente
  - pwd

Navigare nel filesystem

- ls: list → mostra i files in una directory
  - ls
  - ls /usr/sbin
- mkdir: make dir → crea una directory
  - mkdir prova
- cp: copy → copia uno o più files/directory
  - cp sorgente destinazione

    “se destinazione è una directory, il file verrà copiato con lo stesso nome in essa.”

Opzioni e manuale

Ogni comando può accettare delle opzioni e in genere è fornito di un aiuto in linea e di una pagina di manuale

copia di una directory:
cp **-r** /home/utente1/backup /var/backup/users/utente1

cp –**help**

man cp

Spesso le pagine man di un programma/comando sono ricche di esempi e illustrano in dettaglio il funzionamento.

Navigare nel filesystem

- mv: move → sposta/rinomina un file o una directory
  - mv file nuovonome
  - mv file directory
  - mv directory directory
- rm: remove → cancella un file o una directory
  - rm file
  - rm -rf directory

“Molti programmi accettano una lista di files e directory”

cp -r file1 file2 directory3 directory

mv directory1 file2 directory

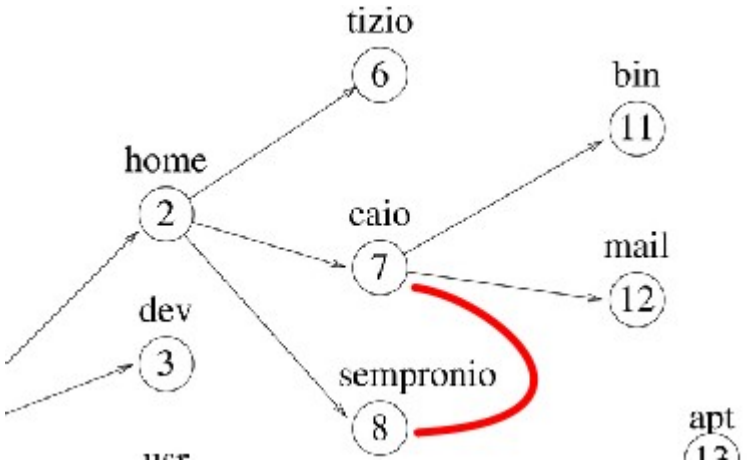
Navigare nel filesystem

Per agevolare le operazioni su gruppi di files è possibile utilizzare i wildcards

- '*':
  - mv prova* directory → tutti i files che iniziano per 'prova'
  - mv *.txt directory → tutti i files che finiscono per '.txt'
- '?':
  - mv prova?.txt directory → es: prova**1**.txt prova**a**.txt
- '[caratteri]':
  - mv prova[234].txt directory → prova**2**.txt prova**3**.txt prova**4**.txt

Link simbolici

A volte ci capiterà di incontrare dei link simbolici. Un link simbolico è un riferimento logico ad un file o ad una directory.



Utilizzando ls li riconoscete per la diversa colorazione, in genere di colore celeste.

Eliminare un link simbolico non eliminerà le directory ed i files originali. Fa eccezione la modifica dei files.

Link simbolici

- ln: link → crea un link
  - ln -s percorsooriginale nomelink
  - ln -s ../directory/file nomelink
  - ln -s /etc/directory directory
  - ln -s /var/directory → se non si specifica il secondo parametro verrà usato il nome del file o della directory originaria

  ATTENZIONE: non dimenticate il parametro "-s"!!! Senza questo parametro creerete un diverso tipo di link, chiamato hard link, che non ha le stesse caratteristiche del link simbolico.

Permessi di accesso

Elementi base del sistema di privilegi di accesso del filesystem di Linux:

Utenti

Gruppi

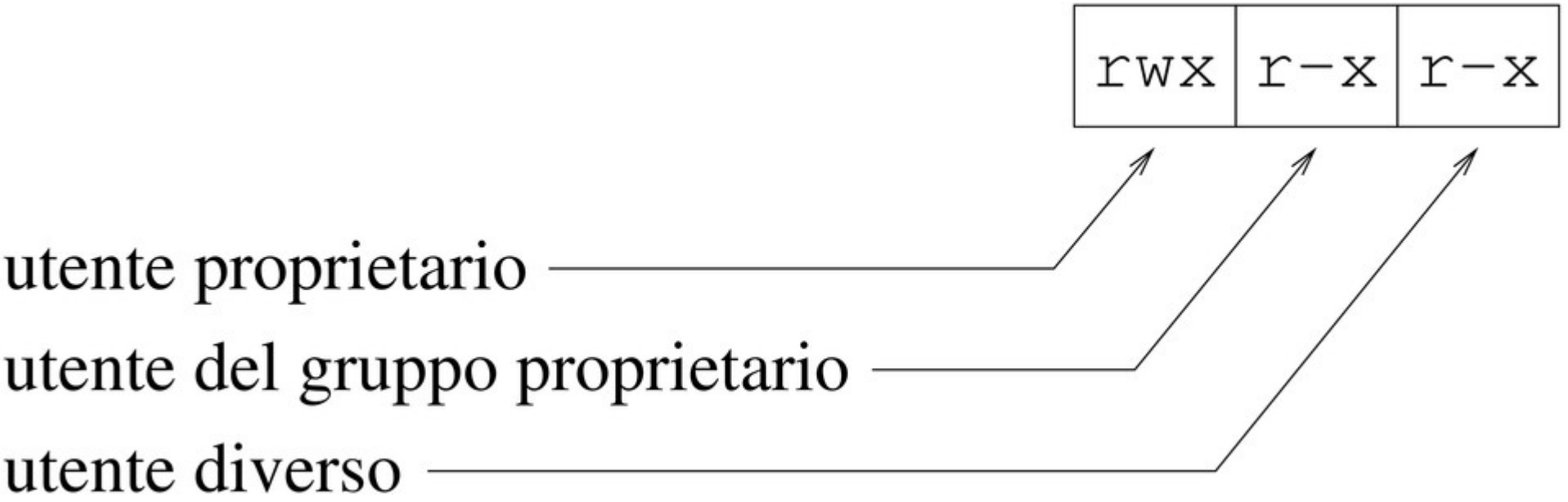
Accesso lettura/scrittura/esecuzione

In realtà questo sistema è nato con Unix: Linux è solo conforme allo standard POSIX

Permessi

$ ls -l

```
drwxr-xr-x 11 kbyte kbyte    4096 16 gen  2008 immagini
drwxr-xr-x 12 kbyte kbyte    4096 16 ago  2009 informatica
drwxr-xr-x  2 kbyte kbyte    4096  4 nov  2007 kaffeine
```



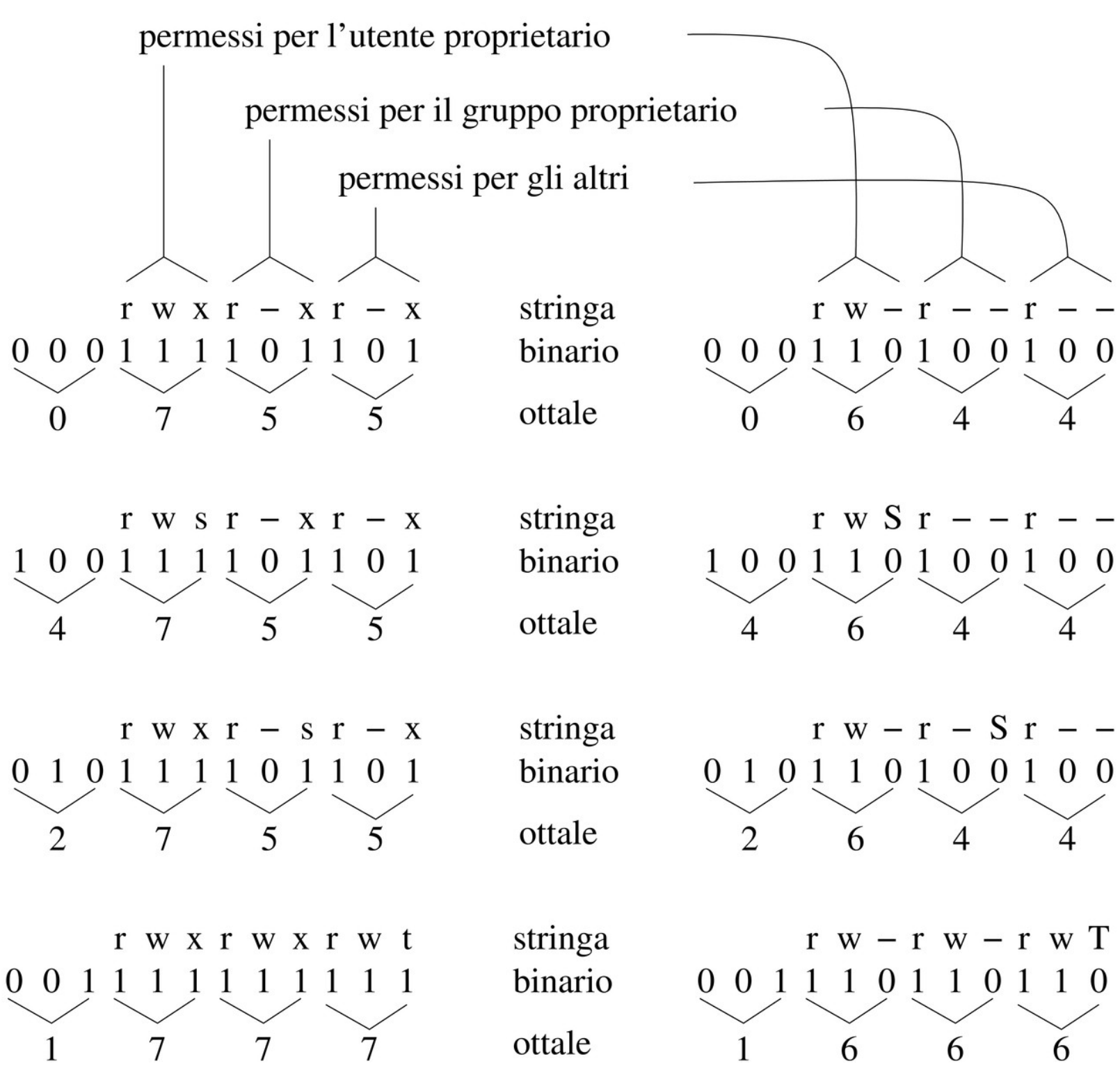
Permessi

- chown: change owner → cambia utente proprietario
  - chown utente directory
  - chown utente:gruppo directory
  - chown -R utente directory

- chgrp: change group → cambia gruppo proprietario
  - chgrp gruppo directory
  - chgrp -R gruppo directory

- chmod: change mode → cambia permessi lettura, scrittura esecuzione
  - chmod g-w file → toglie al gruppo proprietario il permesso di scrittura
  - chmod 755 file → imposta i permessi rwx r-x r-x

“A volte per cambiare proprietari e permessi bisogna essere amministratori”


permessi per l'utente proprietario
permessi per il gruppo proprietario
permessi per gli altri
r w x r – x r – x
stringa
0 0 0 1 1 1 1 0 1 1 0 1
binario
0 7 5 5
ottale
r w – r – – r – –
0 0 0 1 1 0 1 0 0 1 0 0
0 6 4 4
r w s r – x r – x
1 0 0 1 1 1 1 0 1 1 0 1
4 7 5 5
r w S r – – r – –
1 0 0 1 1 0 1 0 0 1 0 0
4 6 4 4
r w x r – s r – x
0 1 0 1 1 1 1 0 1 1 0 1
2 7 5 5
r w – r – S r – –
0 1 0 1 1 0 1 0 0 1 0 0
2 6 4 4
r w x r w x r w t
0 0 1 1 1 1 1 1 1 1 1 1
1 7 7 7
r w – r w – r w T
0 0 1 1 1 0 1 1 0 1 1 0
1 6 6 6

Chi può cambiare cosa

- Un utente proprietario può fare chmod in ogni caso.
- Un utente proprietario può cambiare il gruppo proprietario in uno in cui l'utente fa parte.
- Root può cambiare proprietari e permessi anche senza essere proprietario.

Attenzione!!!

Il flag X applicato in una directory non governa l'esecuzione (è pur sempre una directory), ma l'accesso.

Una directory senza il flag di esecuzione non sarà accessibile e non è possibile accedere ai files nel suo interno.

Attenzione!!!

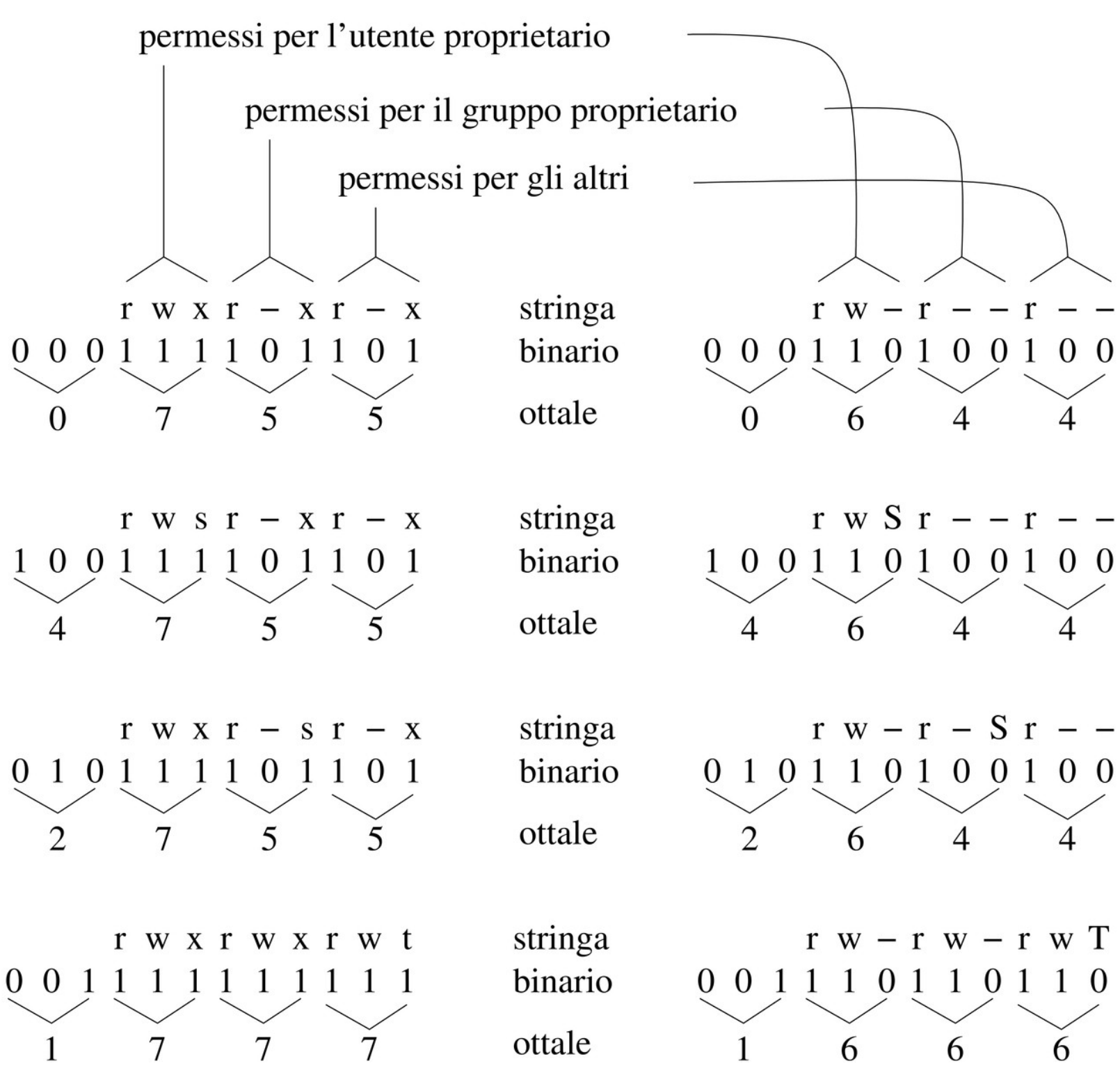
Eliminando il flag R (lettura) di una directory non potremo leggere il suo contenuto con il comando ls e simili.

Tuttavia possiamo accedere ai files al suo interno conoscendo i loro nomi.

Attenzione!!!

Eliminando il flag W (scrittura) di una directory disattiviamo la creazione di nuovi files. Quelli già presenti rimangono modificabili.

Un gruppo o un utente possono creare nuovi files in una directory solo se posseggono il flag di scrittura.


permessi per l'utente proprietario
permessi per il gruppo proprietario
permessi per gli altri
r w x r – x r – x
stringa
0 0 0 1 1 1 1 0 1 1 0 1
binario
0 7 5 5
ottale
r w – r – – r – –
0 0 0 1 1 0 1 0 0 1 0 0
0 6 4 4
r w s r – x r – x
1 0 0 1 1 1 1 0 1 1 0 1
4 7 5 5
r w S r – – r – –
1 0 0 1 1 0 1 0 0 1 0 0
4 6 4 4
r w x r – s r – x
0 1 0 1 1 1 1 0 1 1 0 1
2 7 5 5
r w – r – S r – –
0 1 0 1 1 0 1 0 0 1 0 0
2 6 4 4
r w x r w x r w t
0 0 1 1 1 1 1 1 1 1 1 1
1 7 7 7
r w – r w – r w T
0 0 1 1 1 0 1 1 0 1 1 0
1 6 6 6

Suid (chmod u+s nome)

- Applicato ad un file eseguibile esegue il comando con i privilegi del proprietario

  -rwsr-xr-x root root fileeseguibile

- Applicato ad una directory fa si che i nuovi files in essa abbiano lo stesso utente proprietario

  drwsr-xr-x pippo pippo miadirectory

Sgid (chmod g+s nome)

- Applicato ad un file eseguibile esegue il comando con i privilegi del gruppo proprietario

  -rwxr-sr-x root root fileeseguibile

- Applicato ad una directory fa si che i nuovi files in essa abbiano lo stesso gruppo proprietario

  drwxr-sr-x pippo pippo miadirectory

Sticky bit (chmod +t nome)

- E' usato generalmente per le aree dedicate ai files temporanei

  drwxrwxrwt root root /tmp

- Solo il proprietario di un file (e root) può cancellare o rinominare i files creati.

- Questa protezione serve per rendere più sicuro il sistema da attacchi basati sull'alterazione/corruzione dei files temporanei.

Filesystem e privilegi di accesso

Filesystem Posix

- ext2/3/4
- reiserfs
- brfs
- xfs
- nfs

Filesystem no-Posix

- vfat
- ntfs
- iso9660
- udf

Filesystem Posix compliant

- I permessi sono memorizzati nel dispositivo e vengono caricati nel filesystem di Linux.
- Tutti i cambiamenti di proprietario, gruppo e bit di accesso sono permanenti.
- Nel caso di dispositivi rimovibili, potrebbe capitare un'incoerenza tra il reale utente/gruppo proprietario e quello visualizzato sugli altri computer.

Filesystem di altri tipi

- Il filesystem di Linux non sa come ottenere le informazioni native relative ai permessi.
- Proprietari, gruppi e privilegi possono essere simulati e stabiliti al momento del mount (esempio apertura di un dvd dal desktop)
- In genere i cambiamenti di privilegi vengono negati o assegnati senza diventare permanenti.

Permessi e parametri di mount

Alcune opzioni di mount di un dispositivo possono alterare i permessi di un filesystem:

- **noexec:** impedisce l'esecuzione dei file con il permesso di esecuzione.
- **nosuid:** impedisce che i bit SUID (Set user ID) e SGID (Set group ID) abbiano effetto.
- **user:** scorciatoia per noexec, nosuid e nodev

Esempio: mount /dev/sda6 -o noexec /mnt/test

Permessi e parametri di mount

Per i filesystem no-posix possiamo usare queste opzioni:

- **uid:** Permette di stabilire il proprietario dei file e delle directory.
- **gid:** Permette di stabilire il gruppo proprietario dei file e delle directory.
- **umask=maschera:** permette di stabilire quali permessi inibire nel file system

I permessi di base sono quelli del mount point.

Utilizzo di umask

Prendiamo ad esempio il seguente umask:

umask=0037 → ----wxrwx

Condieriamo un file con questi permessi:

miofile rwxrwxr-x → 775

I permessi risultanti saranno:

```
rwxrwxr-x -
----wxrwx =
-----------------
rwxr-----
```

Cambio di attributi con chattr

Con alcuni filesystem possiamo assegnare uno o più attributi ad un file con il comando chattr.

- **A:** Non aggiorna la data di accesso (atime). Può essere utile se si vuole ridurre l'attività a carico del disco
- **i:** Fa in modo che il file non sia modificabile, né cancellabile, né sia possibile cambiargli nome, né sia possibile creare un collegamento fisico verso di esso

Esempio: chattr +A miofile

Con “lsattr” vediamo quali attributi ha un file.

Sistema dei permessi inadeguato?

Il sistema dei permessi Posix/Unix/Linux è molto semplice, ma oggi può essere considerato obsoleto?

- In molti sistemi un utente e un gruppo bastano
- Il sistema è stato studiato negli anni '70: le capacità di memorizzazione erano esigue e si doveva risparmiare ogni singolo bit.

16bit utente + 16bit gruppo + 12 bit flags = 54bit

ACL: Access Control List

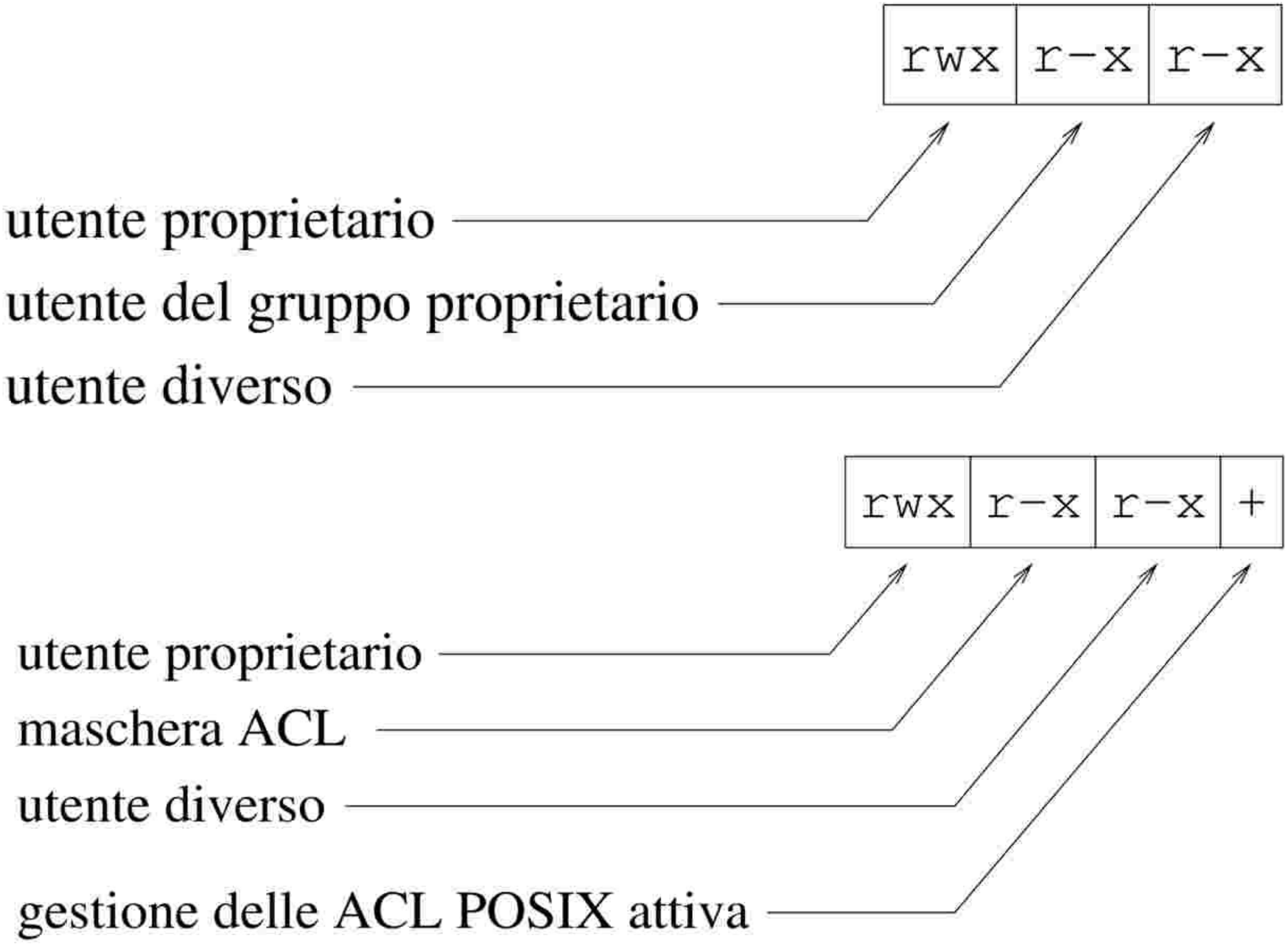
Le Acl nascono con la necessità di poter assegnare più di un utente e più di un gruppo ad un file o una directory.

- {Filesystems}
  - {Second extended fs support}
    - {Ext2 extended attributes}
      - {Ext2 POSIX Access Control Lists}
  - {Ext3 journalling file system support}
    - {Ext3 extended attributes}
      - {Ext3 POSIX Access Control Lists}
  - {The Extended 4 (ext4) filesystem}
    - {Ext4 extended attributes}
      - {Ext4 POSIX Access Control Lists}

Le ACL sfruttano le estensioni del filesystem

Al mount dobbiamo usare l'opzione acl o modificare l'fstab:

mount -o acl /dev/sda3 /mnt/test


rwx r-x r-x
utente proprietario
utente del gruppo proprietario
utente diverso
rwx r-x r-x +
utente proprietario
maschera ACL
utente diverso
gestione delle ACL POSIX attiva

ACL: getfacl

Con il comando getfacl possiamo esaminare le acl di un file o una directory.

$ getfacl primo[Invio]

 # file: primo

 # owner: tizio

 # group: tizio

user::rw-

user:caio:rw-

group::r--

mask::rw-

other::r--

ACL: Tipi di voci

Voce	Descrizione
user::permessi	Dichiarazione dei permessi associati all'utente proprietario (questi permessi non sono filtrati dalla maschera ACL).
group::permessi	Dichiarazione dei permessi associati agli utenti appartenenti al gruppo proprietario.
user:utente:permessi	Dichiarazione dei permessi associati a un utente particolare.
user:gruppo:permessi	Dichiarazione dei permessi associati agli utenti di un gruppo particolare.
mask::permessi	Dichiarazione della maschera dei permessi concessi a tutte le classi di utenti, escluso il proprietario e gli utenti che non ricadono in alcuna categoria specificata.
other::permessi	Dichiarazione dei permessi associati agli utenti che non vengono individuati in alcuna categoria particolare (questi permessi non sono filtrati

ACL: setfacl

Con setfacl possiamo modificare le acl di un file o di una directory.

- **-m: modifica**

  setfacl -m user:caio:rwx miofile

- **-x: elimina**

  setfacl -m user:caio miofile

- **-b: elimina tutte le acl**

  setfacl -b miofile

- **-R: ricorsivo**

  setfacl -R -m user:caio:rwx miadirectory

ACL: importazione ed esportazione

Possiamo importare ed esportare le acl utilizzando i comandi getfacl e setfacl

getfacl miofile > acl.txt

setfacl –set-file=acl.txt miofile2

Possiamo copiare le acl al volo usando un pipe:

getfacl file1 | setfacl --set-file=- file2

ACL: valori di predefiniti

Il parametro -d di setfacl o la parola “default” in una voce dell'acl di una directory faranno si che i nuovi files creati ereditino le acl predefinite.

setfacl -m default:user:www-data:rwx miadir

I nuovi files creati in “miadir” avranno nelle loro acl:

user:www-data:rwx

Approfondimenti e letture

- Appunti di informatica libera
  - 19.12 Gerarchia del file system
  - 19.4 Attivazione e utilizzo
  - 3.18 File e directory in un sistema Unix
  - 3.28 ABC dei comandi Unix
  - 3.21 Permessi
  - 20.6 Proprietà
  - 20.7 Modalità dei permessi
  - 20.8 Attributi speciali
  - 20.9 ACL POSIX con i sistemi GNU/Linux