

# **File System Unix**

Generalità

# File System: Caratteristiche

- Struttura gerarchica
- Files senza struttura (byte streams)
- Protezione da accessi non autorizzati
- Semplicità di struttura

"On a UNIX system, everything is a file; if something is not a file, it is a process."

# File Unix

I tipi principali di File sono:

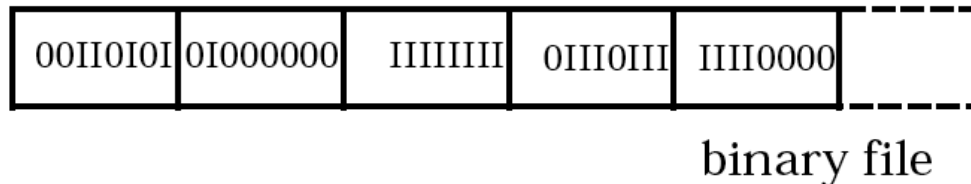
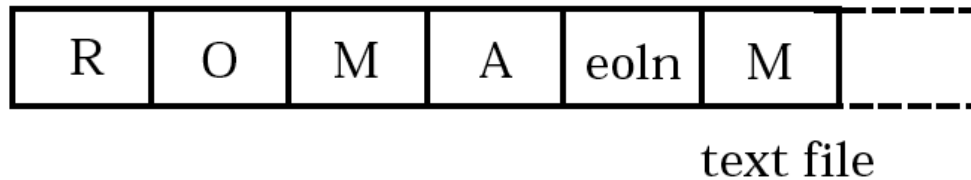
- **File ordinari**
- **Directory**
- **File Speciali**

Il sistema assegna biunivocamente a ciascun file un identificatore numerico, detto

***i-number*** ("index-number"), che gli permette di rintracciarlo nel file system.

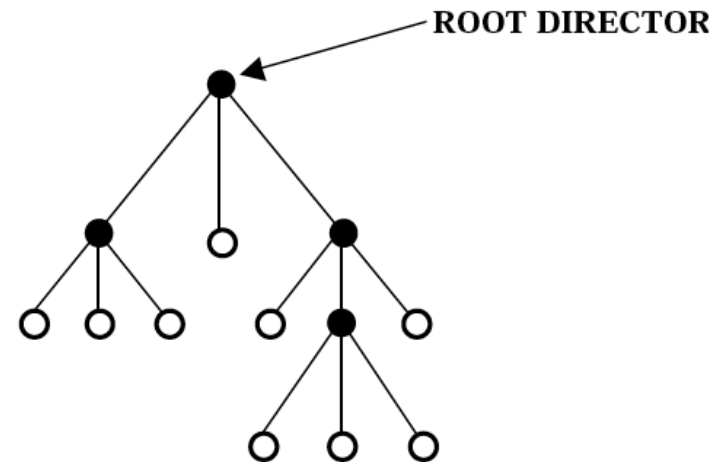
# File Ordinari

- Sono sequenze di byte (byte streams)
- Possono contenere informazioni qualsiasi (dati, programmi sorgente, programmi oggetto,...)
- Il sistema non impone alcuna struttura



# Organizzazione dei File

Per consentire all'utente di rintracciare facilmente i propri files, Unix permette di raggrupparli in cartelle, dette **Directories**, organizzate in una (unica) struttura gerarchica:

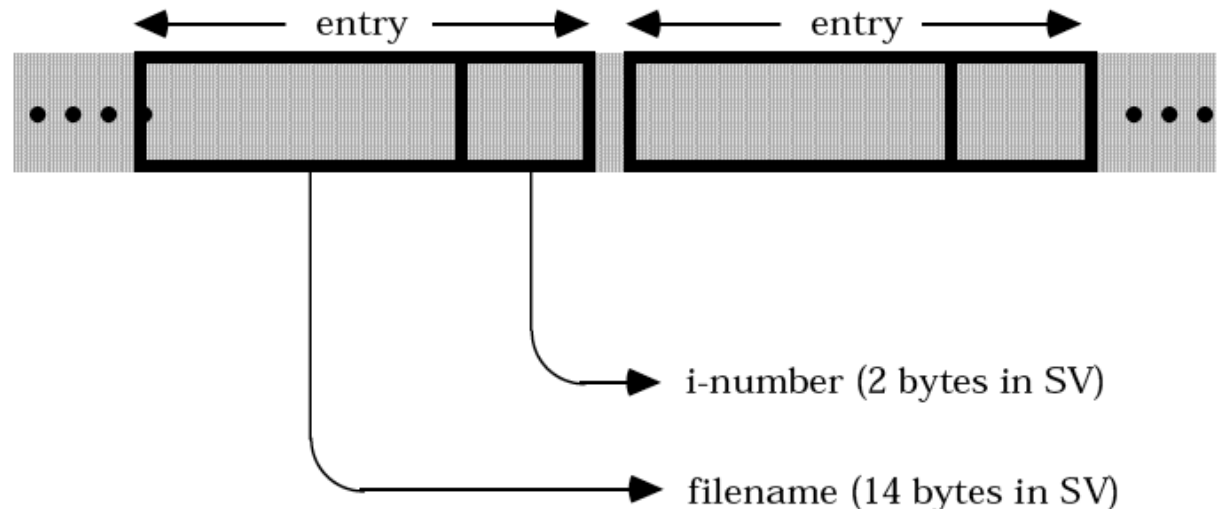


● : directory

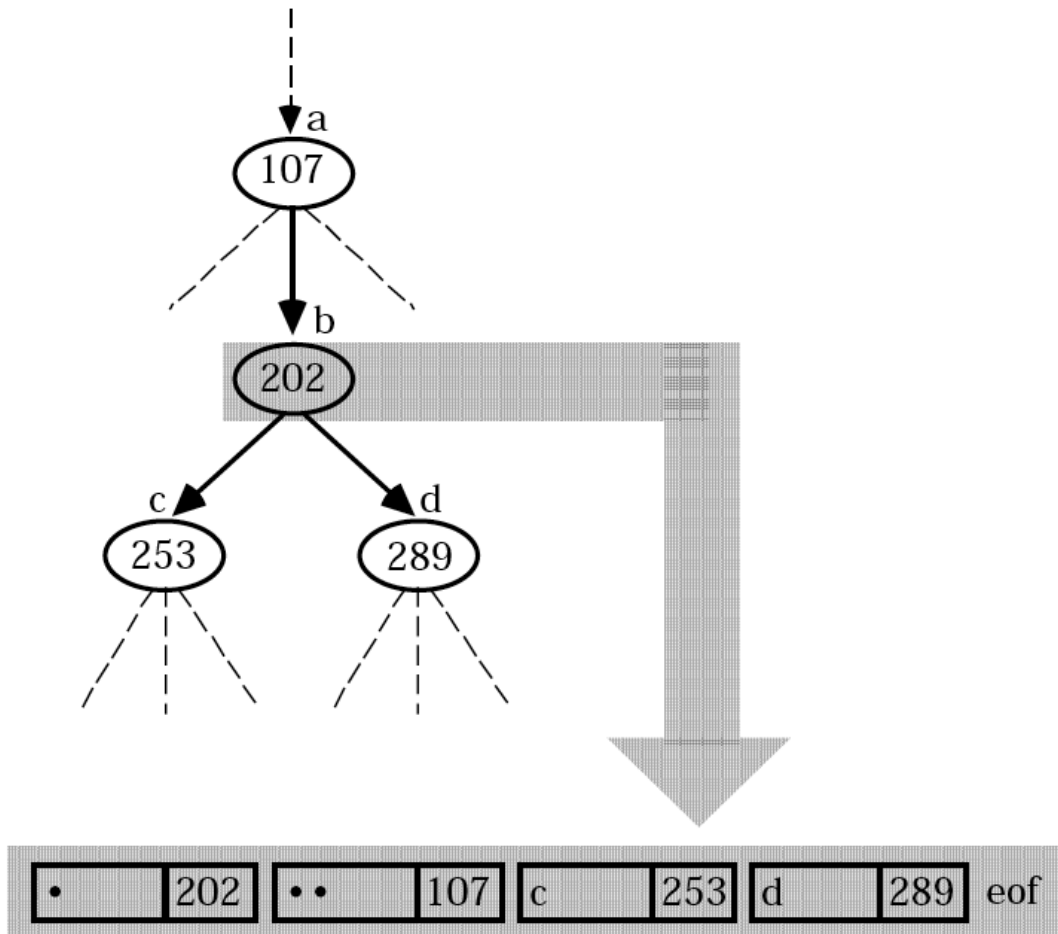
○ : file ordinario  
directory (vuota)  
file speciale

# Directories in Unix

- Sono sequenze di bytes come i file ordinari;
- Differiscono dai file ordinari solo perché non possono essere scritte da programmi ordinari
- Il loro contenuto è una serie di **directory entries**: associazione fra gli i-number (usati dal sistema) e i **filename** mnemonici (usati dall'utente):



# Esempio

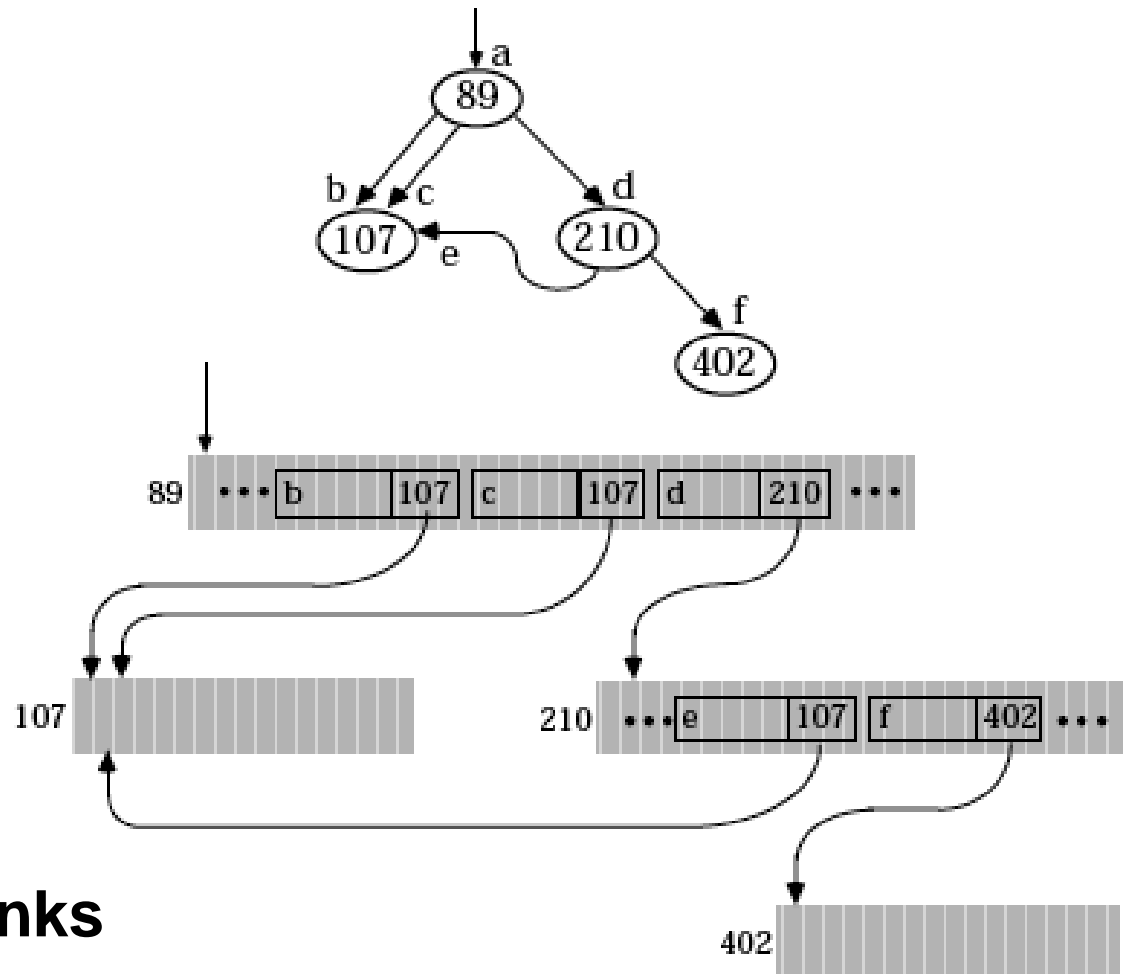


Almeno due entry: la directory stessa ".", la directory padre ".."

# Files Sinonimi

Un file può avere più filename (ma sempre un solo i-number)

Esempio:



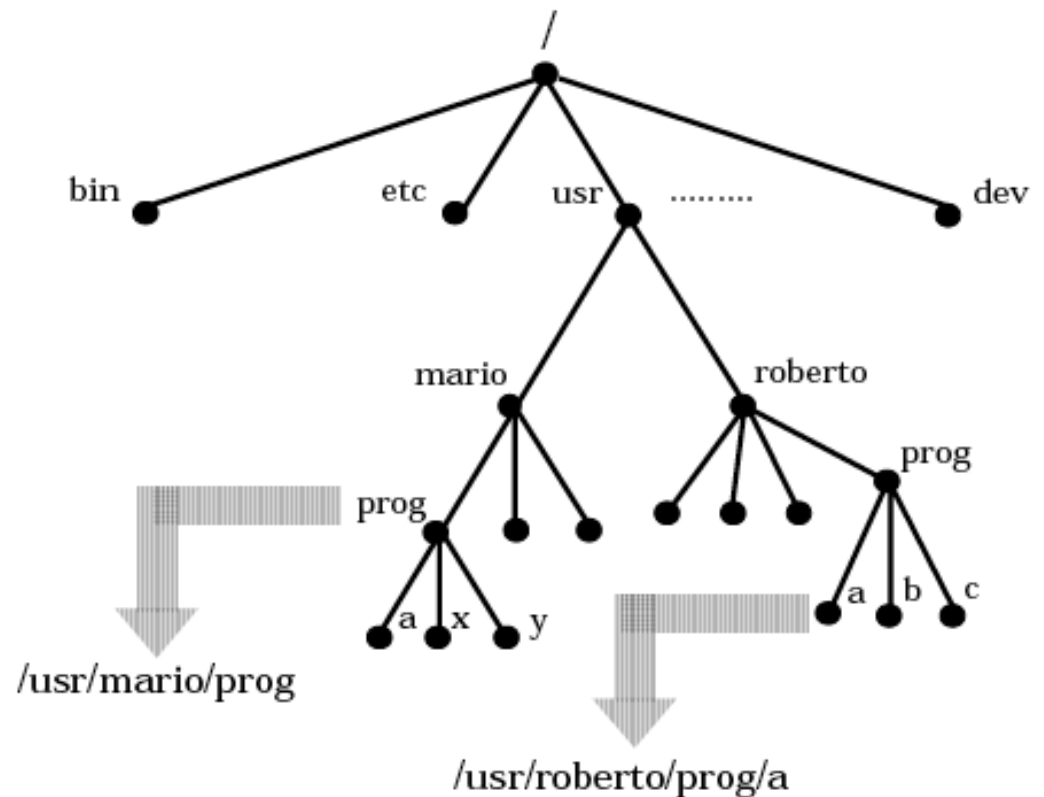
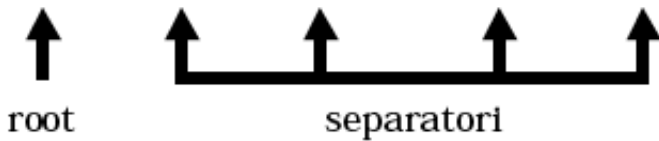
**Il file 107 ha 3 links**



# Pathnames

Ogni file viene identificato univocamente specificando il suo **pathname**, che individua il cammino dalla root-directory al file:

`/dir/dir/.../dir/filename`



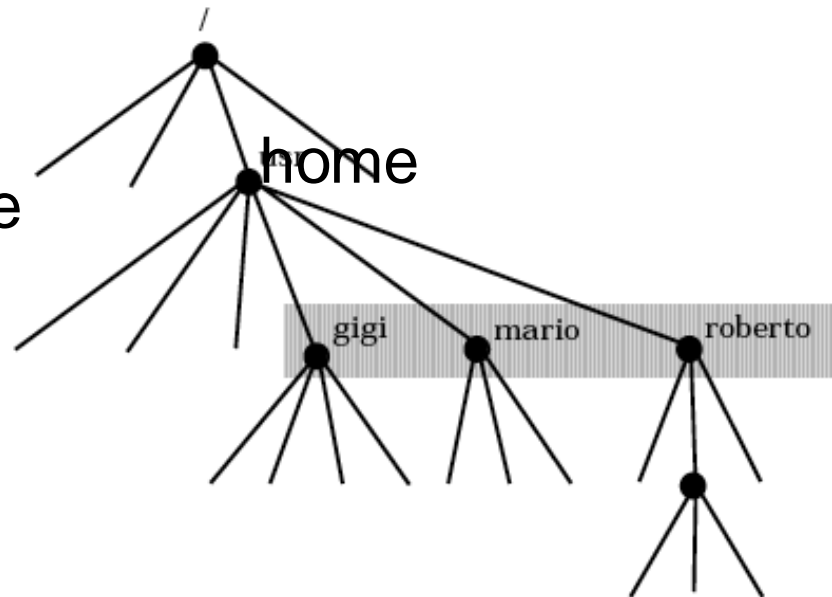
# Tipiche Directories

- /bin comandi eseguibili
- /dev file speciali (I/O devices)
- /etc file per l'amministrazione del sistema, ad esempio:  
    /etc/passwd
- /lib librerie di programmi
- /tmp area temporanea usata dal sistema
- /home home directories
- /usr Programmi, librerie, doc. etc. per i programmi user-related.

# Home Directory

- Ad ogni utente viene assegnata dal system administrator una directory proprietà (**home directory**) che ha come nome lo username del proprietario;
- Ad essa, l'utente potrà appendere files (o subdir):

Per denotare la propria home directory si può usare l'abbreviazione "~"



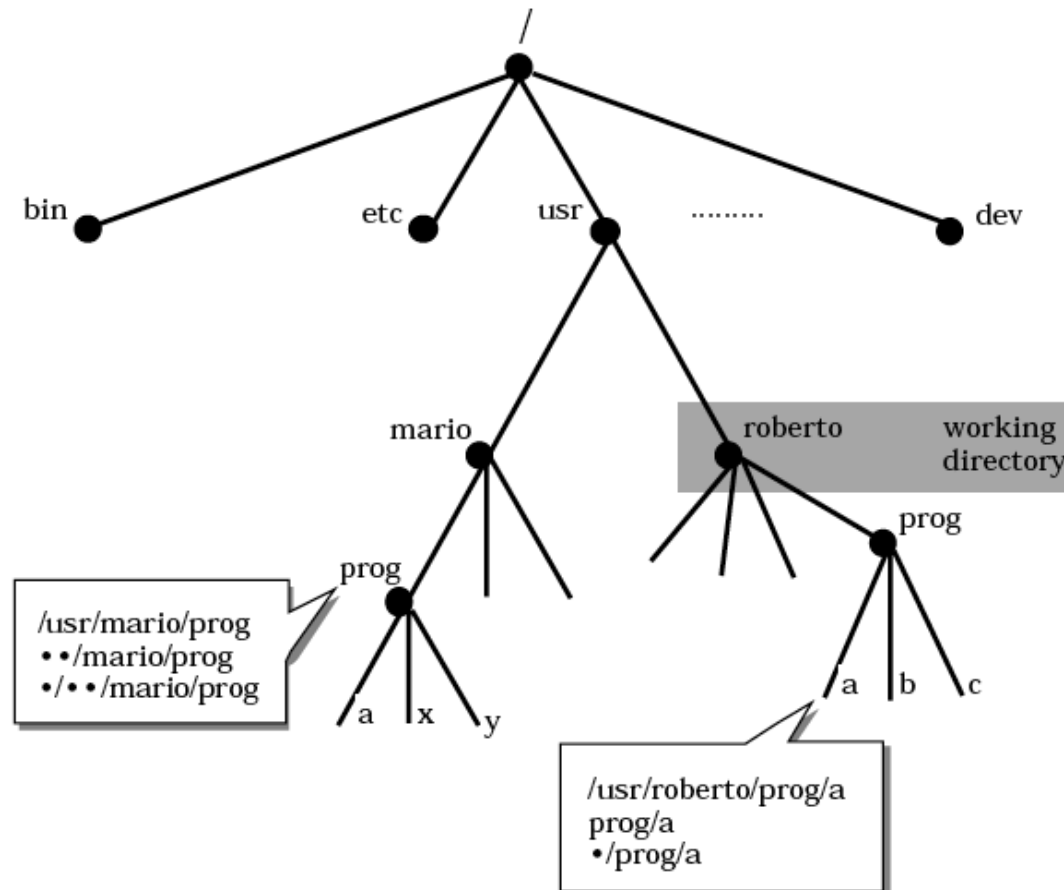
# Working Directory

- Ogni utente opera, ad ogni istante, su una directory corrente, o **working directory**
- Subito dopo il login, la working directory è la home directory dell'utente
- L'utente può cambiare la working directory con il comando (**cd** **change directory**)

# Pathnames Relativi

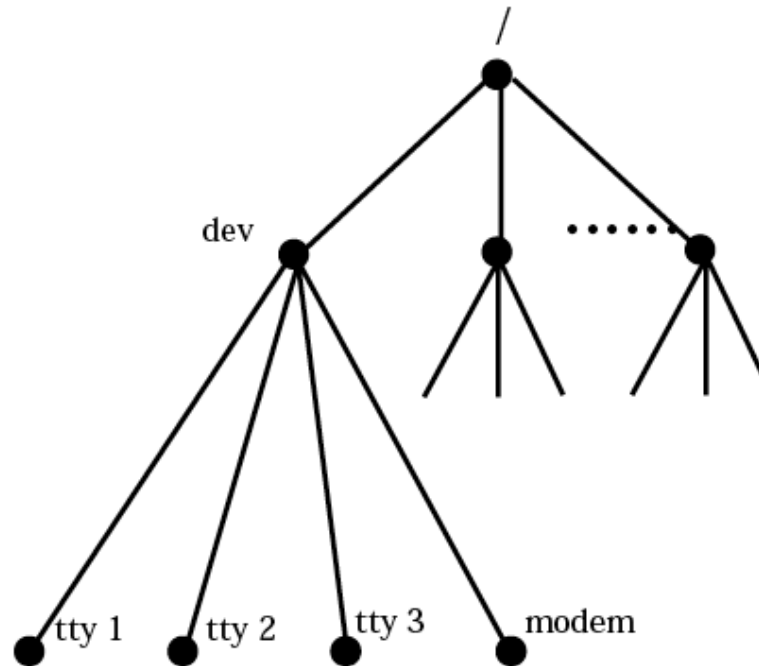
Ogni file può essere identificato univocamente specificando solamente il suo **pathname relativo alla working directory**

**Esempio:**



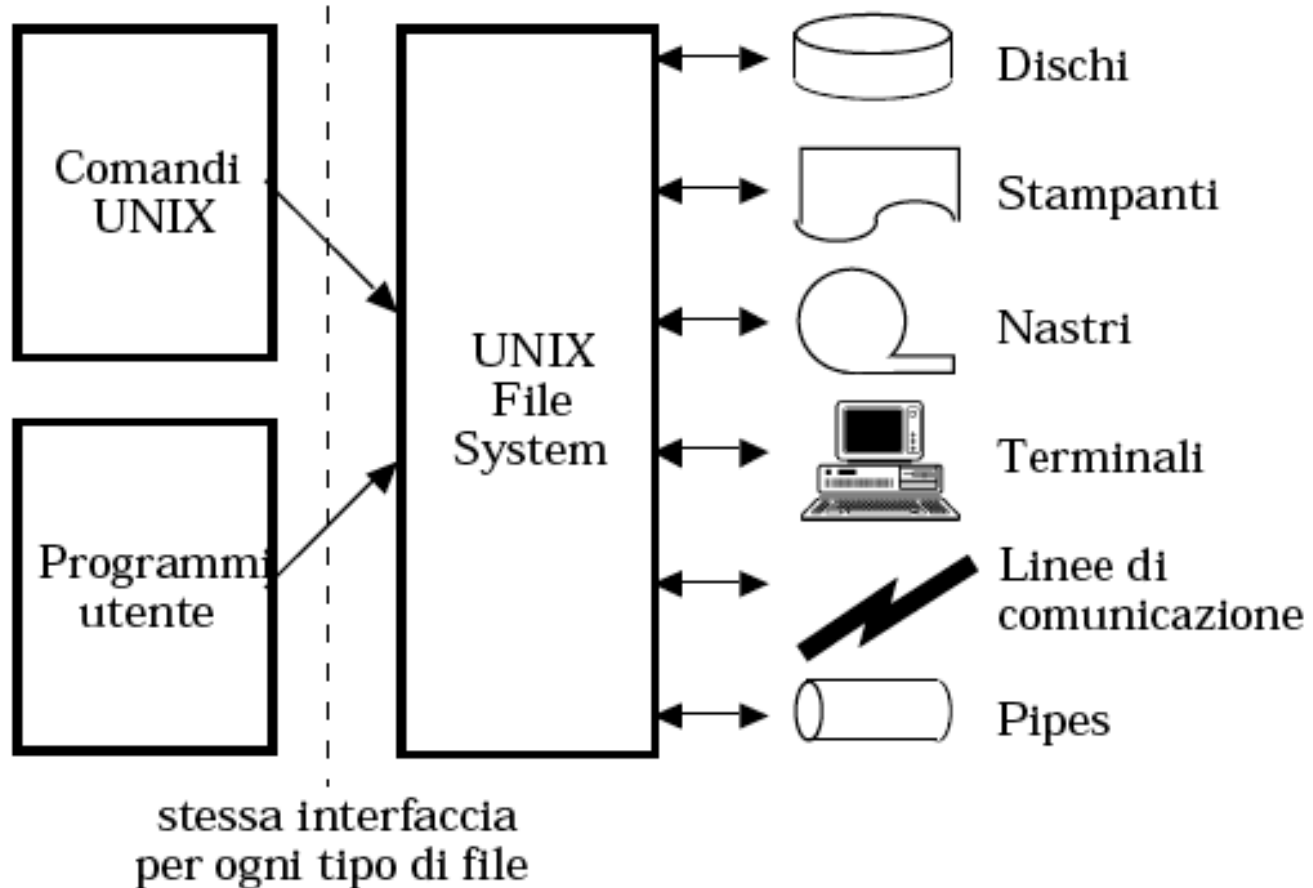
# Files Speciali

- Ogni device di I/O viene visto, a tutti gli effetti, come un file (**file speciale**)
- Richieste di lettura/scrittura da/a files speciali causano operazioni di input/output dai/ai devices associati

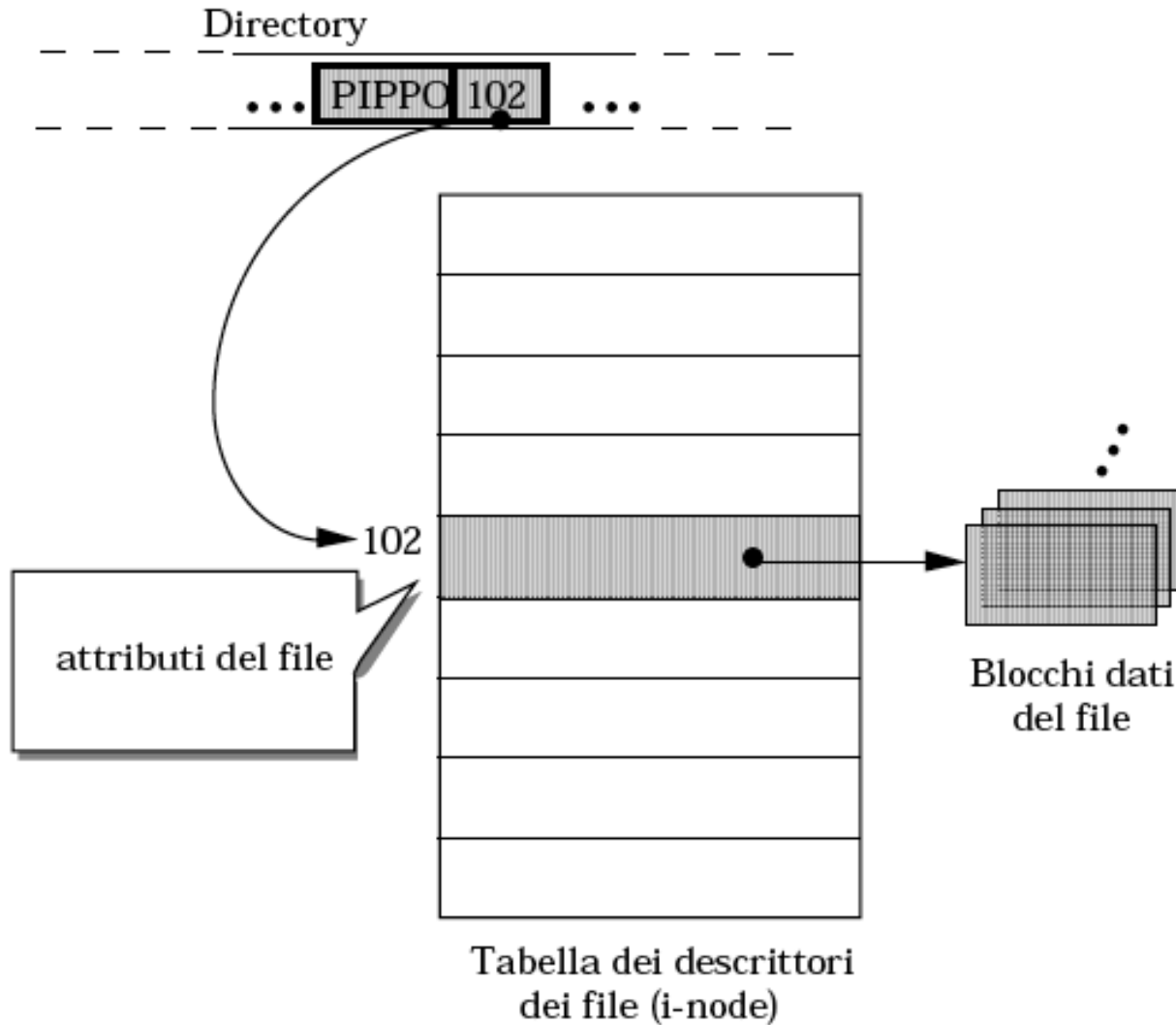


# Files Speciali: vantaggi

- Trattamento uniforme di files e devices
- In Unix i programmi non sanno se operano su un file o su un device



# Implementazione File





# Attributi di un File

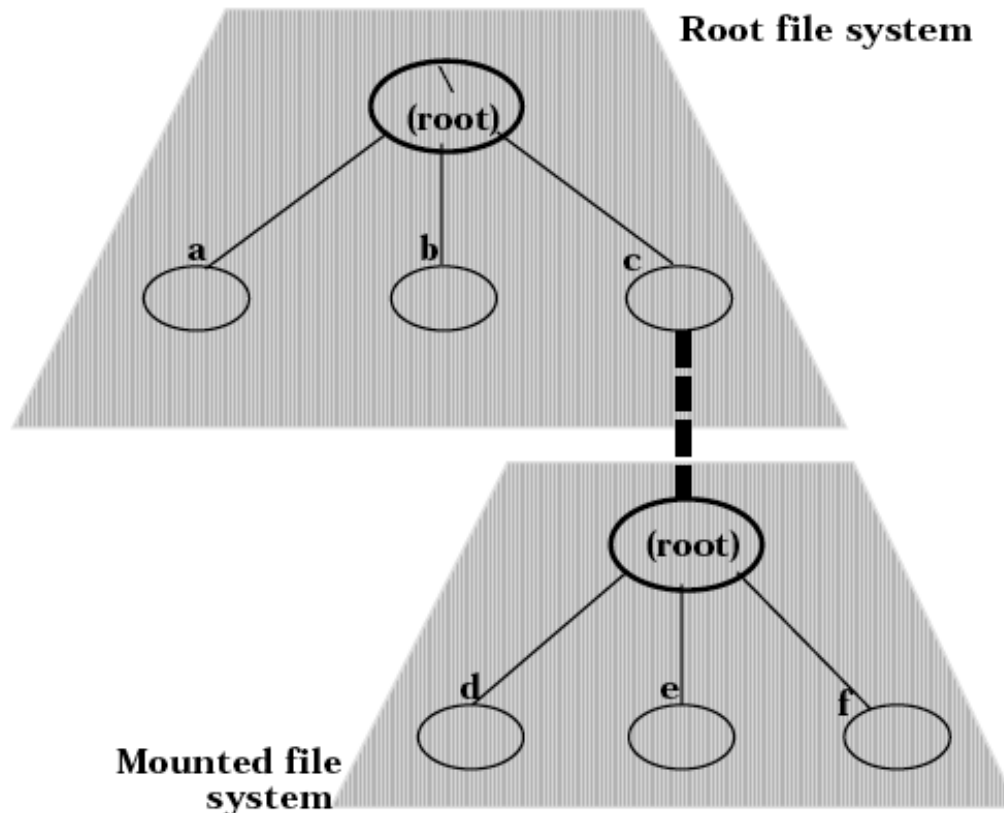
Per ogni file (ordinario, directory, speciale) Unix mantiene le seguenti informazioni nel descrittore del file:

<b>Tipo</b>	ordinario, directory, speciale?
<b>Posizione</b>	dove si trova?
<b>Dimensione</b>	quanto è grande?
<b>Numero di links</b>	quanti nomi ha?
<b>Proprietario</b>	chi lo possiede?
<b>Permessi</b>	chi può usarlo e come?
<b>Creazione</b>	quando è stato creato?
<b>Modifica</b>	quando è stato modificato più di recente?
<b>Accesso</b>	quando è stato l'accesso più recente?

# File System Montabile

Un file system Unix è sempre **unico**, ma può avere parti residenti su device rimovibili:

- "montate" prima di potervi accedere (mount)
- "smontate" prima di rimuovere il supporto (umount)



# Gestione delle Directories

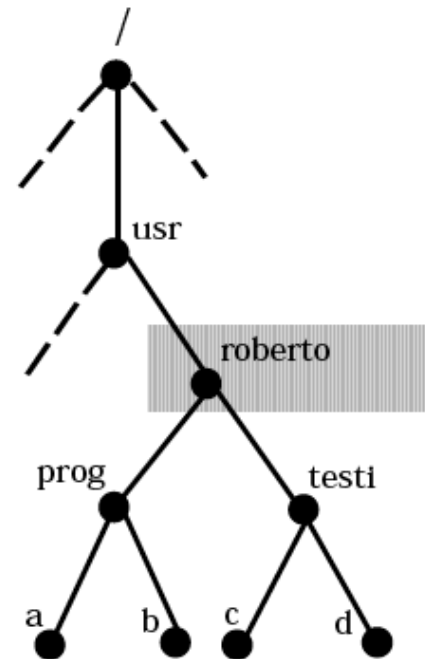
- **pwd**      **print working directory**
- **cd**        **change directory**
- **ls**        **list directory**
- **du**        **disk usage**
- **mkdir**    **make directory**
- **rmdir**    **remove directory**
- **ln**        **link**

# pwd (print working directory)

- Stampa pathname directory corrente

Esempio:

```
% pwd  
/usr/roberto  
%
```

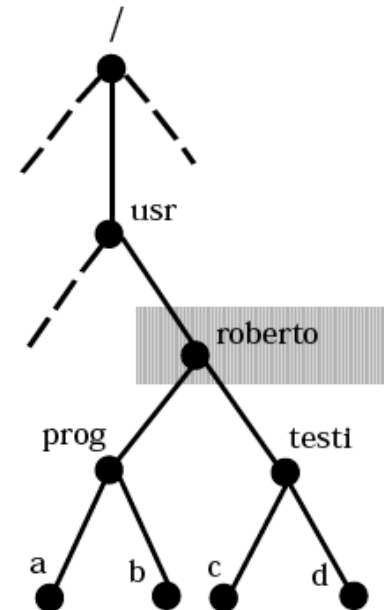


# cd change directory

- La directory specificata diviene la working directory
- se nessuna directory specificata, si "ritorna" alla home directory

Esempio:

```
%cd /usr
%pwd
/usr
%cd
%pwd
/usr/roberto
%
```



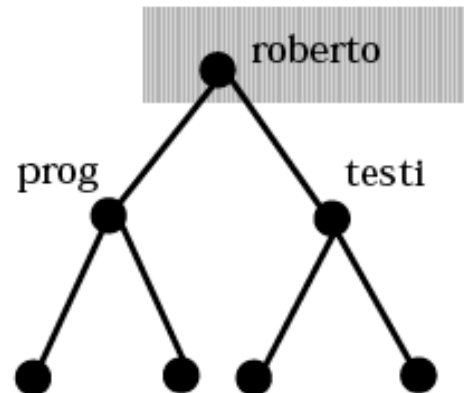
# Il comando ls

**ls** [options][directory...] "list directory"

- lista (in ordine alfabetico) il contenuto della o delle directories indicate
- se nessuna directory indicata, lista il contenuto della working directory
- possiede numerose opzioni

**Esempio:**

```
% ls  
prog testi  
%
```



# Alcune Opzioni

- **-s** fornisce la dimensione in blocchi (**size**)
- **-t** lista nell'ordine di modifica (prima il file modificato per ultimo) (**time**)
- **-1** un nome per ogni riga
- **-F** aggiunge / al nome delle directory e \* al nome dei files eseguibili
- **-R** si chiama ricorsivamente su ogni sottodirectory
- **-i** fornisce l'i-number del file
- ... e molte altre

# ls Esempi

```
% ls  
dir1 file1  
% ls -s  
total 4 2 dir1 2 file1  
% ls -t  
file1 dir1  
% ls -l  
dir1  
file1  
% ls -F  
dir1/ file1  
% ls -R  
dir1 file1  
./dir1:  
file1 file2 file3 file4  
% ls -i  
199742 dir1 51204 file1  
%
```



# File Nascosti (dotfiles)

- I files il cui nome inizia con "." vengono listati
- solo specificando l'opzione -a ("all")
- **Esempio:**

```
% ls -a
```

```
.. .cshrc .mailrc dir1
```

```
.. .login .sh_history file1
```

```
%
```

# ls – campi del formato esteso

Totale dimensione occupata (in blocchi)

Riferimenti al file

Dimensione (byte)

Nome

```
lso:~>ls -l
```

```
total 12
```

```
-rw-rw-r--
```

```
1 lso
```

```
lso
```

```
10 Mar 4 13:29
```

```
a
```

```
-rw-rw-r--
```

```
1 lso
```

```
lso
```

```
10 Mar 4 14:12
```

```
b
```

```
drwxrwxr-x
```

```
2 lso
```

```
lso
```

```
4096 Mar 4 14:29
```

```
c
```

Proprietario

Gruppo primario

Data ultima modifica

Tipo

Permessi

(r)ead, (w)rite, e(x)ecute

(d)irectory, (l)ink, (c)haracter special file, (b)lock special file, (-) ordinary file

# Protezioni di un File

A ciascun file (normale, speciale, directory) sono associati alcuni attributi:

- **Proprietario (owner):** l'utente che ha creato il file
- **Gruppo (group):** il gruppo a cui il proprietario appartiene
- **Permessi (permissions)** Il tipo di operazioni che il proprietario, i membri del suo gruppo o gli altri utenti possono compiere sul file

Proprietario, gruppo e permessi iniziali sono assegnati dal sistema al file al momento della sua creazione.

Il proprietario può successivamente modificare tali attributi con appositi comandi (**chown**, **chgrp**, **chmod**)

# Identificazione Utenti

- Ogni utente viene identificato da uno **user name** assegnato dall'amministratore del sistema. Ad esso corrisponde biunivocamente uno **userid** numerico, assegnato dal sistema
- User name e user-id sono **pubblici**

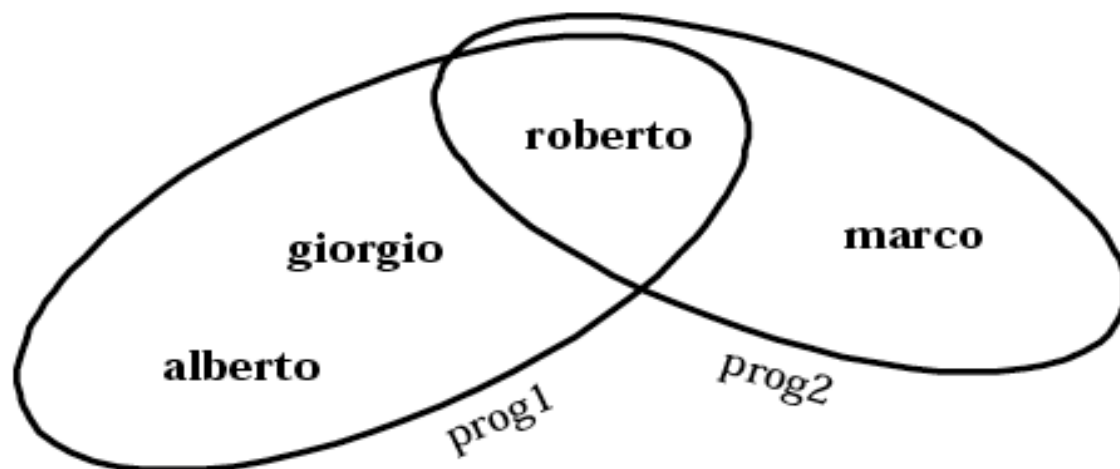
# GRUPPI

---

Ogni utente può far parte di uno o più **gruppi**, definiti dall'amministratore del sistema

Ogni gruppo è identificato da un **group name** di al più 8 caratteri, associato biunivocamente a un **group-id** numerico

**Esempio:**



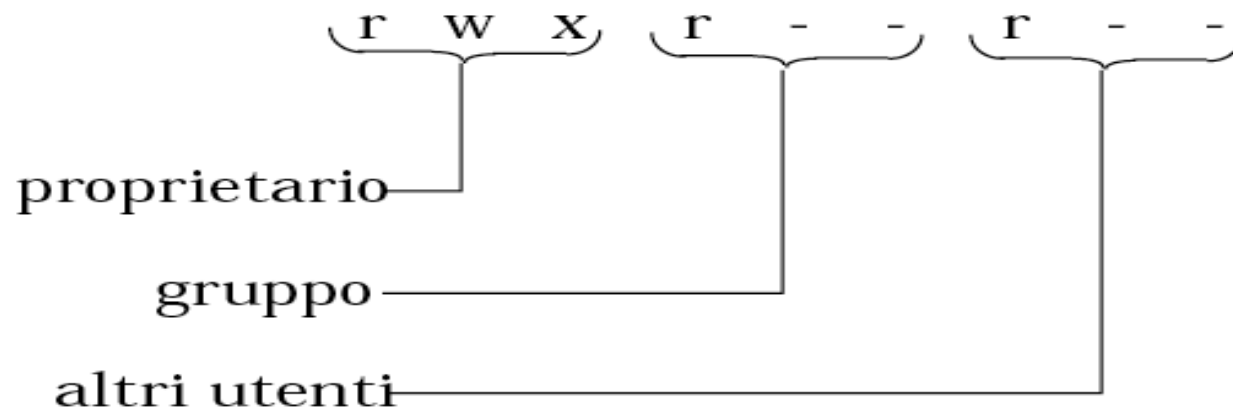
# Permessi

Ad un file possono essere attribuiti i seguenti permessi:

r : readable  
w : writable  
x : executable

} per { proprietario  
gruppo  
altri utenti

**Esempio:**



**In binario:** 1 1 1            1 0 0            1 0 0

**In ottale:**     7                    4                    4

# Permessi iniziali

- Alla creazione di un file, Unix assegna i seguenti permessi:
- Per i *files ordinari non eseguibili*:

rw-rw-rw

110 110 110

6 6 6

- Per i *files ordinari eseguibili* e per directories:

rwx rwx rwx

111 111 111

7 7 7

# Comando chmod

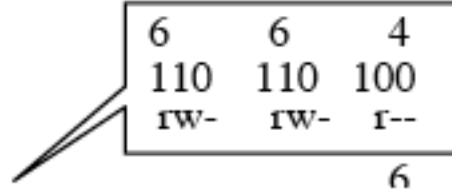
**chmod** *permissions filename...*

"change mode"

- attribuisce le *permissions a filename*  
(solo da parte del proprietario del file!)
- *permissions* può essere espresso in forma ottale o simbolica



## Permessi in forma ottale:



```
% chmod 664 file1 file2
```

```
% ls -l
```

```
total 4
```

```
-rw-rw-r-- 1 roberto  usrmail  35 Mar 11 16:34 file1
-rw-rw-r-- 1 roberto  usrmail  17 Mar 11 16:17 file2
```

```
%
```

## Permessi in forma simbolica:

```
% ls -l
```

```
total 4
```

```
-rw-rw-r-- 1 roberto  usrmail  35 Mar 11 16:34 file1
-rw-rw-r-- 1 roberto  usrmail  17 Mar 11 16:17 file2
```

```
% chmod ugo+x file1
```

```
% chmod o=rwx file2
```

```
% ls -l
```

```
total 4p
```

```
-rwxrwxr-x 1 roberto  usrmail  17 Mar 11 16:34 file1
-rw-rw-rwx 1 roberto  usrmail  17 Mar 11 16:17 file2
```

[ugoa] [+ - =] [rwx]

r read  
w write  
x execute

+ aggiungi  
- toglì  
= assegna

u proprietario (user)  
g gruppo (group)  
o altri utenti (other)  
a tutti (all)

# chown (change owner)

```
chown [options] [user][: [group]]  
file...
```

Cambia proprietario e/o gruppo primario per uno o più file.

Se dopo “:” non segue il nome del gruppo, viene attribuito il gruppo principale cui appartiene user.

Se prima di :group non viene indicato il nome dell'utente, viene cambiato solo il gruppo primario (chgrp)

## IL COMANDO `chgrp`

---

```
chgrp newgroupid file...
```

"change group"

- *newugroupid* diventa il nuovo gruppo dei *file...*
- il comando può essere eseguito solo dal proprietario (o dal superuser)

# mkdir e rmdir

- **mkdir** directory ... : Crea la/le directory

Esempio:

```
% mkdir dir1 dir2
% ls
dir1 dir2
```

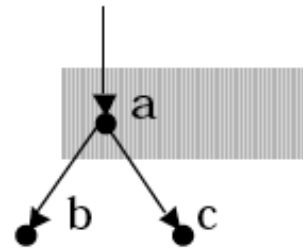
- **rmdir** directory ... : rimuove la/le directory (deve essere vuota)

```
% rmdir dir
rmdir: dir: Directory not empty
% ls dir
a
% rm dir/a
% rmdir dir
```

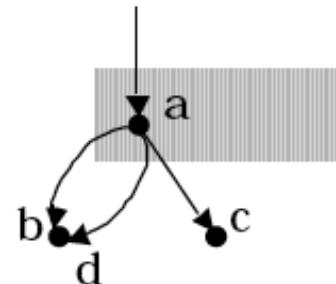
# Il comando “link”: ln

**ln name1 name2** "link"

associa il nuovo nome (link) name2 al file (esistente) name1, che non può essere una directory



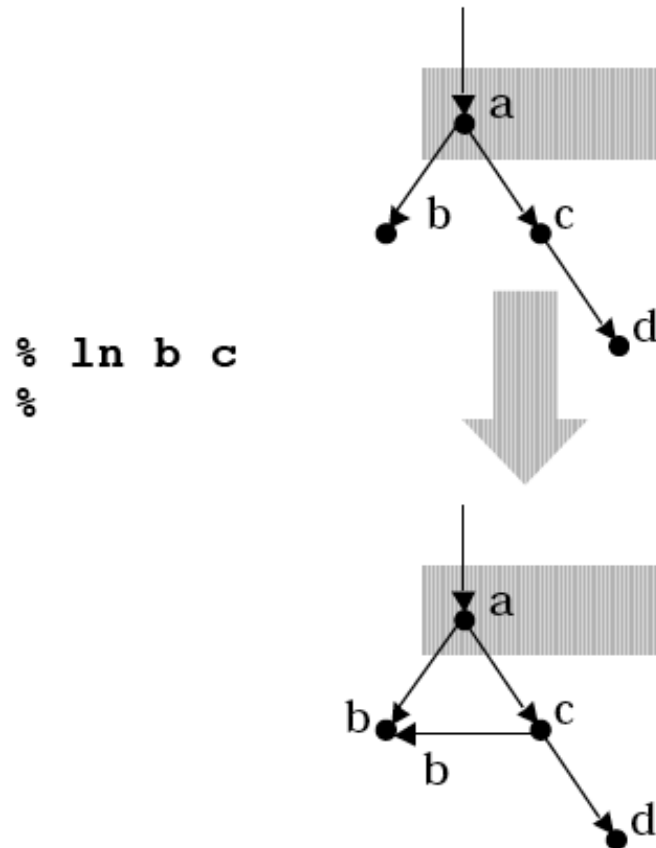
```
% ln b d  
%
```



# Il comando “link”: ln

**ln name1 name2** "link"

Se name2 è una directory, il nuovo nome è *name2/name1*

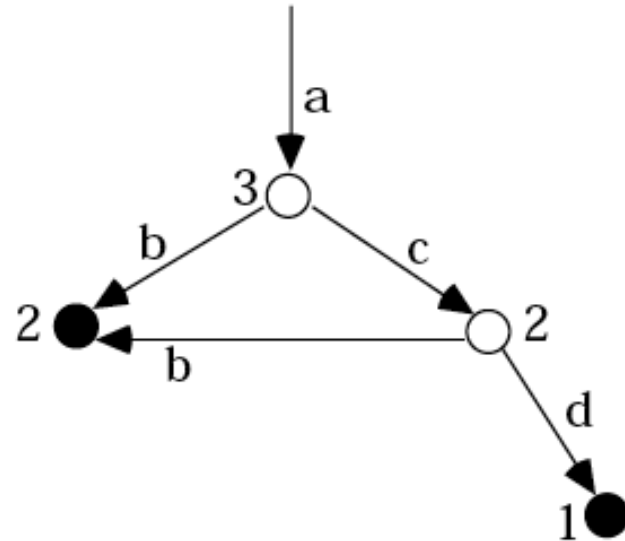


# Numero links

- Numero links e' un attributo gestito dal sistema

Per vedere:

ls -l



○ directory  
● file

# Esempio

```
% mkdir dir
% touch file
% ls -l
```

Crea file

```
total 2
drwxr-sr-x  2 roberto  usrmail  512   Mar 11 19:40 dir
-rw-r--r--  1 roberto  usrmail    0   Mar 11 19:40
file
```

```
% ln file nuovo
```

Crea link a file da nuovo

```
% ls -li
199742 dir  51204 file  51204 nuovo
```

```
% ls -l
```

```
total 2
drwxr-sr-x  2 roberto  usrmail  512   Mar 11 19:40 dir
-rw-r--r--  2 roberto  usrmail    0   Mar 11 19:40 file
-rw-r--r--  2 roberto  usrmail    0   Mar 11 19:40
nuovo
```



# Esempio

```
% ln file dir
```

link a file da directory

```
% ls -l dir
```

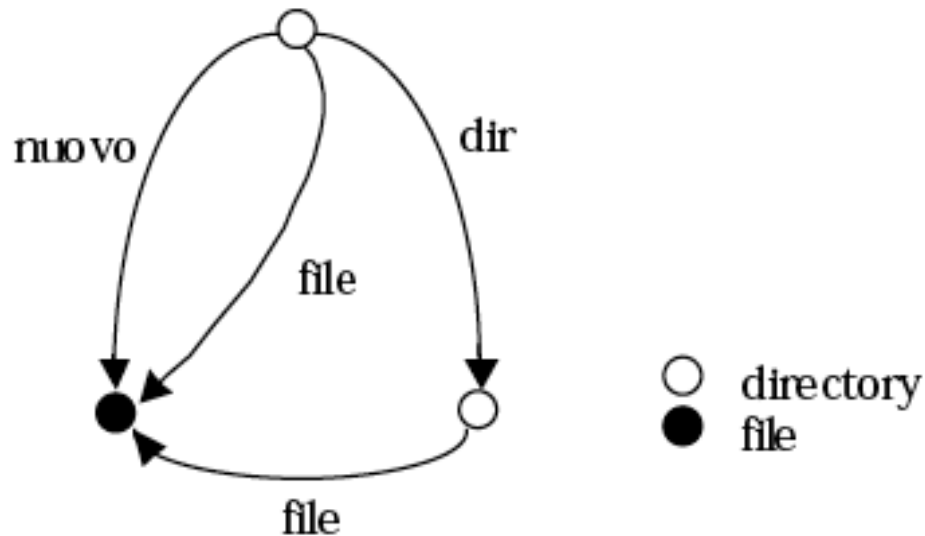
```
total 0
```

```
-rw-r--r--  3 roberto  usrmail      0 Mar 11 19:40 file
```

```
% ln dir nuovissimo
```

```
ln: dir is a directory
```

```
%
```

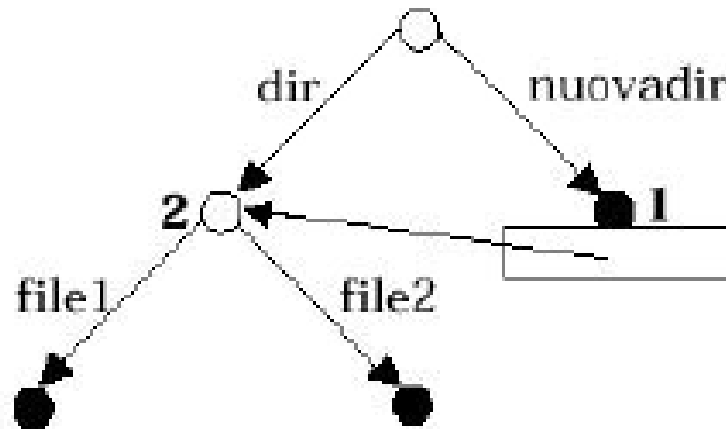


# Il comando “link”: In

- Tutti i link allo stesso file hanno identico status e caratteristiche
- Non è possibile distinguere la entry originaria dai nuovi link
- I link di questo tipo non possono essere fatti con file che stanno su file system diversi

# Link Simbolici

- In `-s name1 name2`
- Permette di creare link a directory;
- Permette di creare link fra file o directory che stanno su file system diversi;
- Viene creato un file name2 che contiene il link simbolico (i.e. il path di name1)



## Esempio:

```
% ls
```

```
dir
```

```
% ls dir
```

```
file1 file2
```

```
% ln -s dir nuovadir
```

```
% ls
```

```
dir nuovadir
```

```
% ls nuovadir
```

```
file1 file2
```

```
% ls -l
```

```
total 4
```

```
drwxr-sr-x  2 roberto  usrmail  512 Mar 11 19:24
```

```
dir
```

```
lrwxrwxrwx  1 roberto  usrmail    3 Mar 11 19:24
```

```
nuovadir -> dir
```

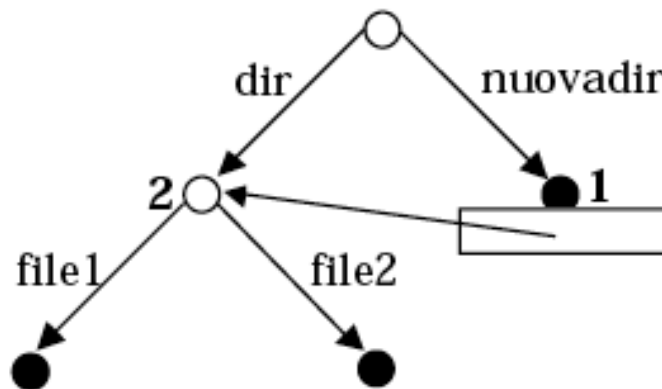
```
%
```

C'e' una directory ...

... contenuto della directory...

...link simbolico a **dir** da **nuovadir**

...**dir** con 2 rif. **nuovadir** con 1 rif.



○ directory  
● file

# Comando mv

- **mv** [options] name...target
1. muove il file o directory name sotto la directory target;
  2. se name e target non sono directories, il contenuto di target viene sostituito dal contenuto di name

## Se *target* è una directory:

- Caso1:

```
% ls
file1          file2          targetdir
% mv file1 file2 targetdir
% ls
targetdir
% ls targetdir
file1          file2
% mv targetdir/file1 targetdir/file2 .
% ls
file1          file2          targetdir
```

- Caso2:

## Se *target* è un file:

```
% ls
file1          file2          file3          targetfile
% mv file1 targetfile
% ls
file2          file3          targetfile
% mv file2 file3 targetfile
mv: Target targetfile must be a directory
Usage: mv [-f] [-i] f1 f2
        mv [-f] [-i] f1 ... fn d1
        mv [-f] [-i] d1 d2
```

- **Caso3: Se *target* non esiste:**

```
% ls
file1          file2
% mv file1 file2 target
mv: target not found
% mv file1 target
% cat target
contenuto di file1
%
```

# Comando **cp**

- **cp** [options][name...] target
- come **mv**, ma name viene copiato

```
% ls
file1 file2 targetdir
% cp file1 file2 targetdir
% ls . targetdir
.:
file1 file2 targetdir

targetdir:
file1 file2

% ls
file1 targetfile
% cp file1 targetfile
% ls
file1 targetfile
~
```



# Comando `rm`

- `rm [-r] name...`
- rimuove i files indicati
- se un file indicato è una directory: messaggio di errore, a meno che non sia specificata l'opzione `-r`  
... nel qual caso, rimuove ricorsivamente il contenuto della direttrice