

UNIX FILE SYSTEM

Per iniziare a conoscere le caratteristiche del File System di UNIX e dei S.O. che ad esso si rifanno (LINUX, Free BSD, AIX, AUX, ULTRIX, ecc.) si raccomanda vivamente di leggere il par. III (**THE FILE SYSTEM**) ed il par. IV (**IMPLEMENTATION OF THE FILE SYSTEM**) dell'articolo originale che ne illustrano le caratteristiche **The UNIX Time-Sharing System**, Communications of the ACM, 17(7), 365-375, 1974.

File system di UNIX: caratteristiche generali

- Struttura gerarchica
- File senza struttura ("byte stream")
- Protezione da accessi non autorizzati
- File & device independence

Tipi di file

- Files ordinari
- Directory
- Files speciali

File ordinari

- Sono sequenze di byte ("byte stream")
- Possono contenere informazioni qualsiasi
 - (dati, programmi sorgente, programmi oggetto,...)
 - File di tipo testo (formati da linee di caratteri ASCII)
 - File binari: sequenze di codici binari
- Il sistema non impone nessuna struttura alla sequenza di byte

| | | | | | | |
|---|---|---|---|-----|---|--|
| U | N | I | X | eol | M | |
|---|---|---|---|-----|---|--|

 File testo

| | | | |
|---------|---------|---------|--|
| 0010010 | 0110010 | 1010110 | |
|---------|---------|---------|--|

 File binario

Tipi di file

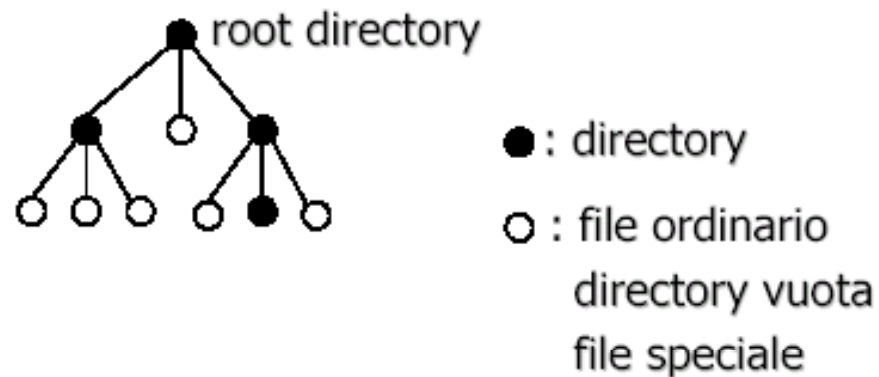
- Files ordinari
- Directory
- Files speciali

Directory

- Sono sequenze di byte, come i file ordinari
- A differenza dei file ordinari
 - Non contengono dati ma un elenco di nomi di file e relativi riferimenti ad altre strutture dati del file system
 - Non possono essere scritte da programmi ordinari
- Una directory è un indice contenente i riferimenti (i-number) di tutti i file memorizzati nella directory stessa

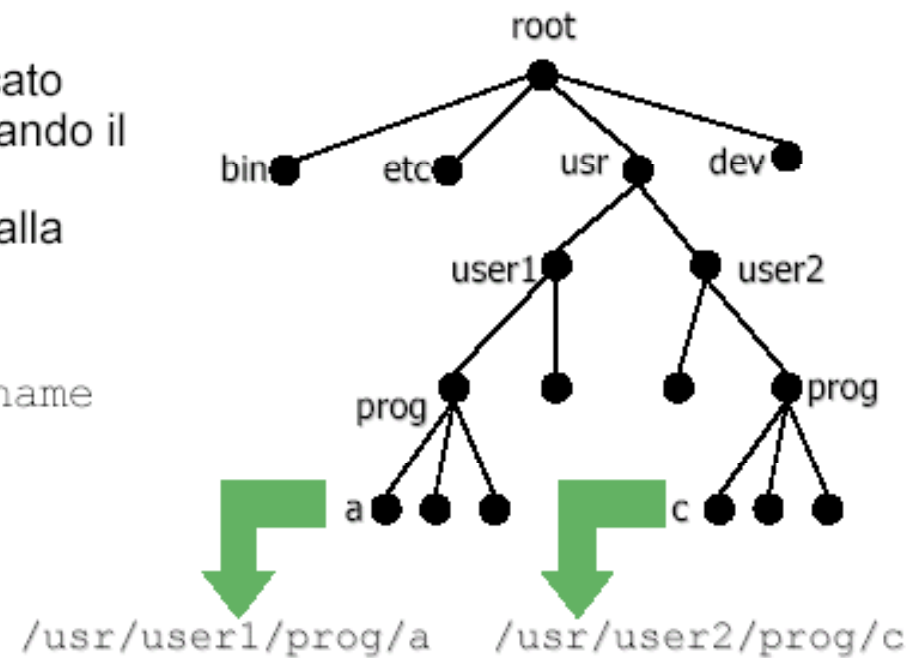
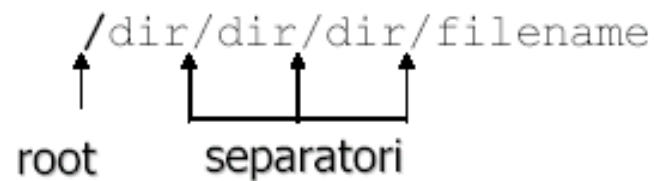
Organizzazione gerarchica dei file

- Per consentire all'utente di rintracciare facilmente i propri file, Unix permette di raggrupparli in **directory**, organizzate in una (unica) struttura gerarchica:



Pathname

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



Tipiche directory di sistema

`/bin` comandi eseguibili

`/dev` files speciali (I/O devices)

`/etc` files per l'amministrazione del sistema, ad esempio:

`/etc/passwd`

`/etc/termcap`

`/lib` librerie di programmi

`/tmp` area temporanea

`/usr` home directory degli utenti

N.B. La struttura può variare da versione a versione

Home directory

- Ad ogni utente viene assegnata, da parte del system administrator, una directory di sua proprietà (**home directory**) che ha come nome lo username dell'utente stesso
- In essa, l'utente potrà creare tutti i file (o subdirectory) che desidera
- Spesso, ma non sempre, le home directory sono sotto la directory di sistema `/usr`
- Per denotare la propria home directory si può usare l'abbreviazione "`~`"

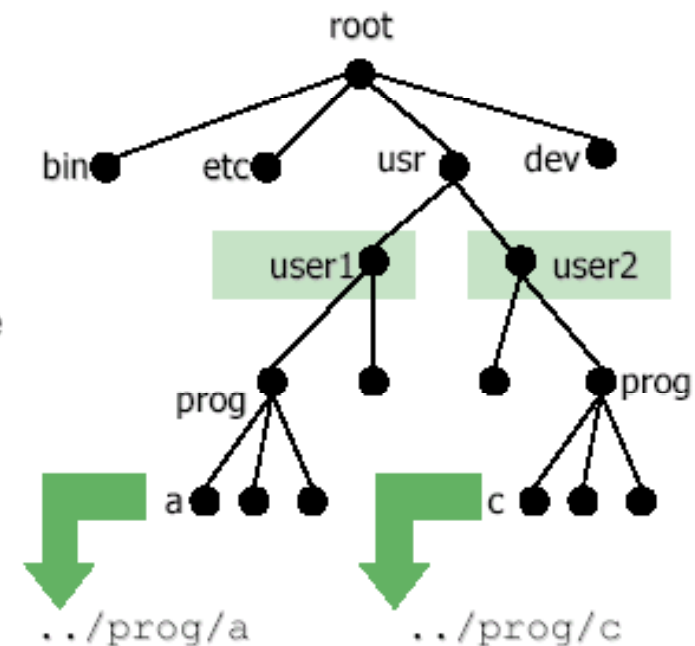
Working directory



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

Pathname relativi

- Ogni file può essere identificato univocamente specificando solamente il suo **pathname relativo alla working directory**
 - non è possibile avere due file con lo stesso nome nella stessa directory (pathname identico), ma è possibile avere due file con lo stesso nome in directory diverse (pathname diverso)

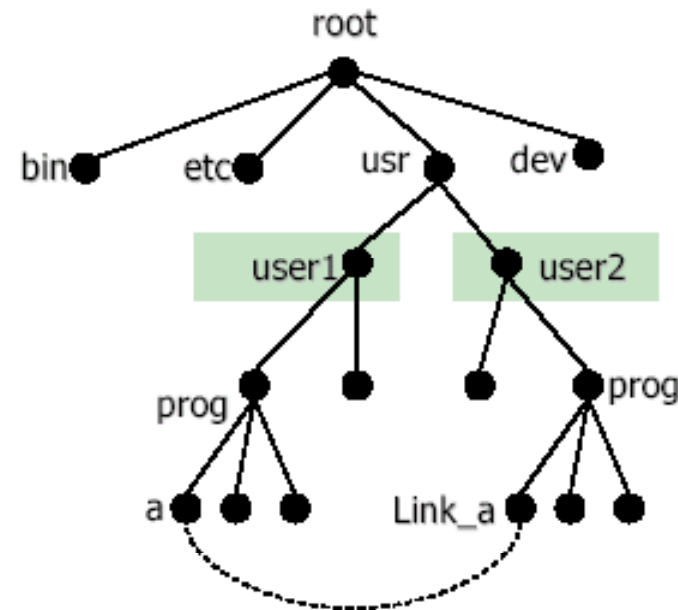


Link

- I link sono particolari file ordinari che puntano ad altri file o directory
 - Lo scopo è rendere più flessibile la struttura gerarchica del file system
 - Consentono la condivisione di un file tra directory diverse, evitandone la duplicazione
- Con l'utilizzo dei link, la struttura del file system diviene un grafo aciclico

La condivisione di un file è ottenuta mediante la primitiva LINK oppure tramite il comando `ln`

```
ln /usr/user1/prog/a link_a
```



Il file `a` è identificato mediante due cammini differenti:

```
/usr/user1/prog/a
```

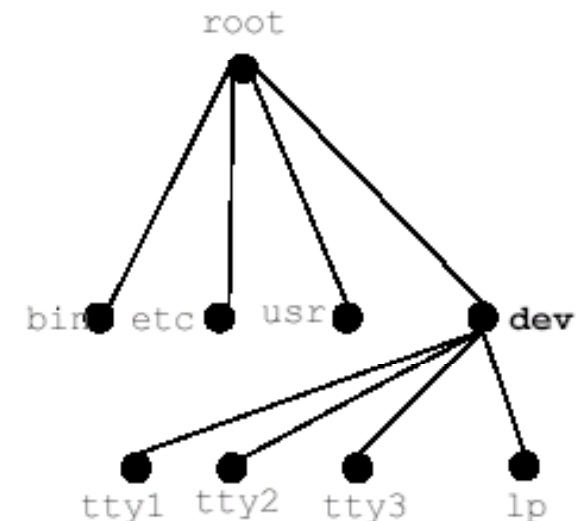
```
/usr/user2/prog/link_a
```

Tipi di file

- Files ordinari
- Directory
- Files speciali

File speciali

- Ogni device di I/O viene visto, a tutti gli effetti, come un file (**file speciale**)
 - **A blocchi**: associati a dispositivi che presentano blocchi di informazione accessibili direttamente (es. dischi)
 - **A caratteri**: associati a dispositivi che presentano un flusso di caratteri in ingresso o uscita (es. terminali, stampanti)
- Richieste di lettura/scrittura da/a file speciali causano operazioni di input/output dai/ai device associati
 - Tutte le operazioni di I/O relative ai dispositivi fisici vengono effettuate applicando le normali primitive definite per file normali, sui corrispondenti file speciali



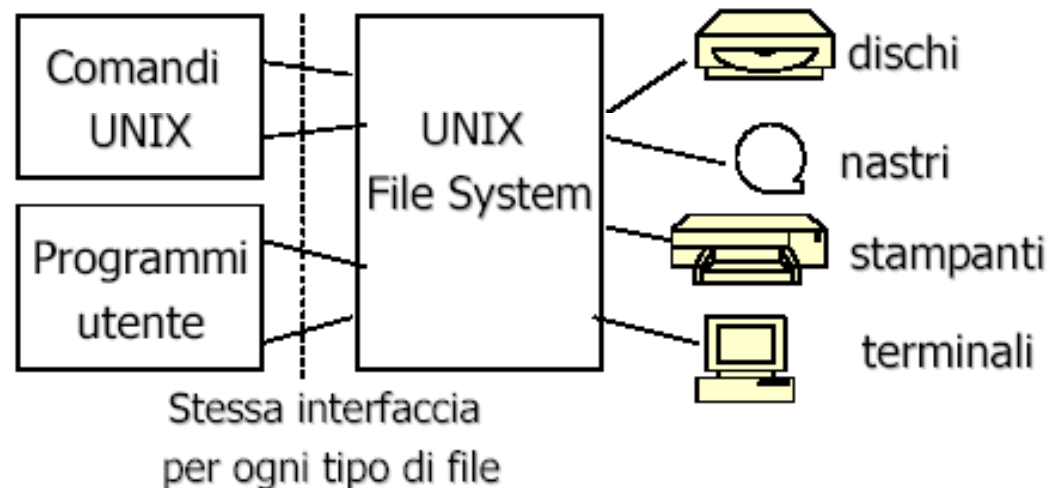
```
cp file /dev/lp
```

File speciali: vantaggi

Trattamento uniforme di file e device



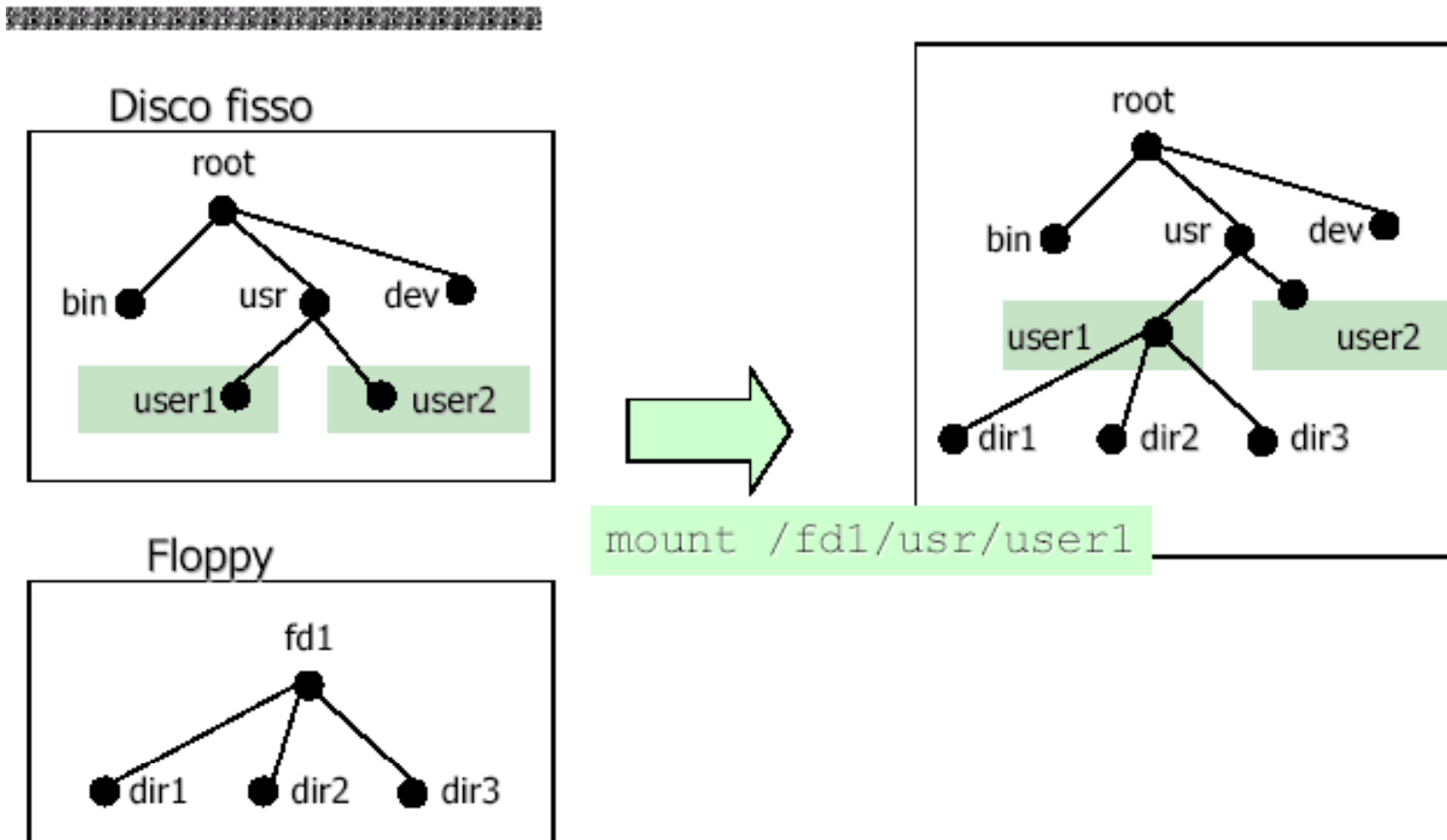
File & device independence: programmi portabili e facilmente interfacciabili con ogni tipo di device



MOUNT: File system removibili

- Un file system Unix è sempre **unico**, ma può avere parti residenti su device rimovibili (es. dischetti)
- Queste parti devono essere:
 - "montate" prima di potervi accedere (**mount**)
 - "smontate" prima di rimuovere il supporto (**umount**)
- La system call **mount** ha due argomenti:
 - nome di un file ordinario
 - nome di un file speciale relativo ad un device removibile contenente un file system (fd1, fd2)
- Dopo la mount non c'è più distinzione tra il file system presente sul disco rimovibile e quello presente sul disco fisso
- La stessa tecnica si usa per suddividere il file system fra diversi device, anche se non rimovibili

Effetto della MOUNT

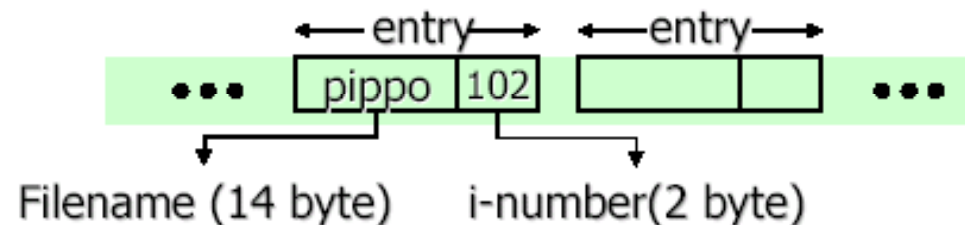


Implementazione del File system (cenni)

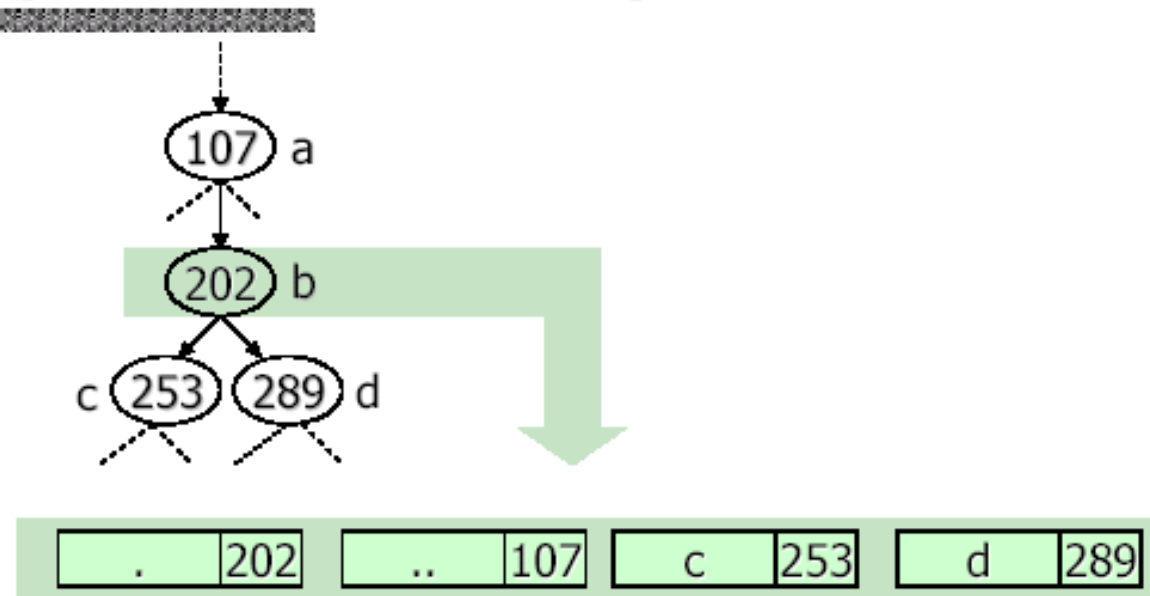
- Ad ogni file è associata una piccola tabella, detta **i-node** ("index-node"), contenente
 - gli attributi del file
 - gli indirizzi dei primi blocchi del disco su cui è memorizzato il file
 - L'indirizzo di un blocco a singola indirazione contenente gli indirizzi di ulteriori blocchi di dati su disco
 - L'indirizzo di un blocco a doppia indirazione contenente gli indirizzi di blocchi a singola indirazione
- Attraverso l'i-node, il sistema operativo può rintracciare il file nel file system
- Ogni i-node è identificato da un **i-number**

Implementazione del File system (cenni)

- Ogni directory è implementata come una serie di **directory entry** che definiscono l'associazione fra gli i-node (usati dal sistema) e i **filename** mnemonici (usati dall'utente)
- Ogni entry (16 byte) contiene solo il nome del file e il numero dell'i-node del file (gli attributi del file sono contenuti nell'i-node)

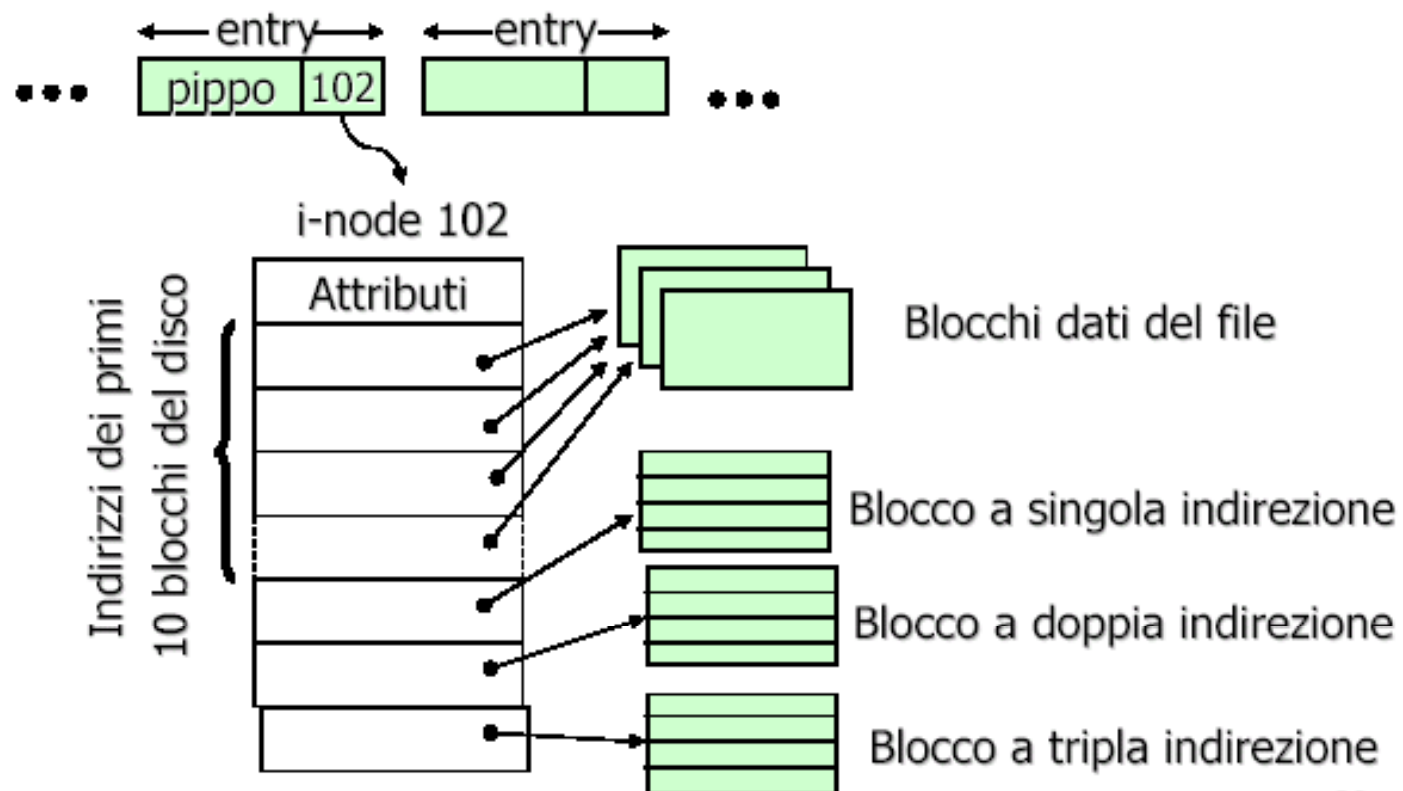


Esempio di directory



- Ogni directory ha almeno 2 entry:
- "." la directory stessa
- ".." la directory padre

Implementazione del file system (cenni)



Attributi di un file

- Gli attributi contenuti nell'i-node di un file sono:
 - Tipo: ordinario, directory, speciale
 - Posizione: dove si trova
 - Dimensione: quanto è grande
 - Numero di links: quanti nomi ha
 - Permessi: chi può usarlo e come
 - Creazione: quando è stato creato
 - Modifica: quando è stato modificato di recente
 - Accesso: quando è stato l'accesso più recente

Nomi di file

Generalmente le regole di formazione dei nomi di file e directory sono le stesse

- UNIX è "case-sensitive": distingue tra maiuscole e minuscole
- Sono consentiti tutti i caratteri ASCII
 - Tuttavia è opportuno evitare i metacaratteri \$, ;, \, &, !, *, | (a meno che non vengano racchiusi tra apici, in tal modo si evita che lo shell li interpreti come operatori)
 - Il carattere / (slash) non può mai essere usato, in quanto impiegato per definire il pathname
- Il nome di un file può essere lungo fino a 256 caratteri (ma in alcune directory è limitata a 14 caratteri)
- Opzionalmente, il nome di un file può presentare una estensione (separata dal .) che generalmente indica il formato del file o il tipo di informazione in esso contenuta
 - Estensioni più comuni:
 - .c**: file sorgente in C
 - .tar**: file compresso con il comando *tar*
 - .gz**: file compresso con il comando *gzip*
 - .tar.gz** : file compresso prima con il comando *tar* e poi con *gzip*

Nomi di file



- | | |
|---|---|
| <ul style="list-style-type: none">■ Nomi di file riservati:<ul style="list-style-type: none">/ (slash) root directory. (period) directory attuale.. (double period) directory padre~ (tilde): home directory | <ul style="list-style-type: none">■ File il cui nome inizia con "." sono detti file nascosti■ Esempi:<ul style="list-style-type: none">– ".login": è un file che lo shell esegue al momento del login al sistema; contiene le definizioni delle variabili di ambiente e dei comandi– ".logout": usato per cancellare file di lavoro, eseguire operazioni di manutenzione del SW, fornire l'indicazione della durata della sessione |
|---|---|

Alcuni attributi di file

| Attributo | Significato |
|----------------------|--|
| Protezione | Chi può accedere al file e in che modo |
| Creatore | Identità della persona che ha creato il file |
| Proprietario | Proprietario attuale |
| Flag di sola lettura | 0 per lettura/scrittura, 1 per sola lettura |
| Flag nascosto | 0 per normale, 1 per non visualizzare nel listato |
| Flag ASCII/binario | 0 per file ASCII, 1 per file binario |
| Tempo di creazione | Data e ora del momento in cui è stato creato il file |
| Ultimo accesso | Data e ora dell'ultima modifica del file |
| Dimensione attuale | Numero di byte nel file |

Un file, al momento della creazione, è di proprietà dell'utente che lo ha creato e del suo gruppo. Il proprietario può cambiare mediante opportuni comandi (`chown`, `chgrp`)

Permessi di accesso a file e directory



- Ad un file possono essere attribuiti i seguenti permessi:
 - **Lettura** (`r``ead`)
 - **Scrittura** (`w``rite`)
 - **Esecuzione** (`x``ecute`)
- I permessi sono definiti per
 - utente proprietario
 - gruppo (a cui appartiene il proprietario)
 - altri

Significato dei permessi

Per i file ordinari:

- r: leggere il contenuto
- w: modificare il contenuto
- x: eseguire il file (ha senso solo se il file contiene un programma)

Per i file speciali:

- r: leggere dal device (input)
- w: scrivere sul device (output)
- x: non significativo


Per le directory:

- r: leggere il contenuto directory (es.: ls, con x abilitato)
- w: modificare la directory, cioè aggiungere, rinominare, rimuovere files (con x abilitato)
- x: accesso (scansione) della directory (per leggere, modificare, eseguire un file in essa contenuto)

NB: i permessi definiti su un file dipendono dai permessi della directory che contiene il file

Permessi di accesso a file e directory

- Ad ogni file (e directory) è associata una stringa di 10 caratteri, detta **lista di controllo degli accessi** o **Access Control List (ACL)**, contenente i diritti di accesso al file

| | owner | gruppo | altri | | | | | | | | | |
|---|---|--------|-------|---|---|---|---|---|---|---|---|---|
|  | <table border="1"><tr><td>r</td><td>w</td><td>x</td></tr></table> | r | w | x | <table border="1"><tr><td>r</td><td>w</td><td>x</td></tr></table> | r | w | x | <table border="1"><tr><td>r</td><td>w</td><td>x</td></tr></table> | r | w | x |
| r | w | x | | | | | | | | | | |
| r | w | x | | | | | | | | | | |
| r | w | x | | | | | | | | | | |

d: directory

- :file normale

b :file speciale a blocchi

c :file speciale a caratteri

Permessi di accesso a file e directory

- Un file (directory), al momento della creazione, è per default reso accessibile in lettura e scrittura all'utente proprietario, mentre ne è vietato qualunque accesso a tutti gli altri utenti, compresi quelli del gruppo di appartenenza
 - La sua ACL sarà: - rw- --- --- (file)
 - d rwx --- --- (directory)
- L'utente proprietario può cambiare la ACL in qualunque momento mediante il comando `chmod`
- Set usuale di permessi
 - Per file: - rv- r- r—
 - Per directory: d rwx r-x r-x

Permessi di accesso a file speciali



- Le regole di protezione dei file normali si applicano anche ai dispositivi di I/O attraverso i file speciali
 - Esempio: se i bit di protezione per la directory `/dev` sono settati in modo da proibire l'accesso a tutti tranne che al superuser, allora agli utenti non è permesso l'accesso all'I/O in modo autonomo
 - L'accesso ad alcuni dispositivi di I/O è consentito con l'installazione di particolari programmi ai quali è concesso di leggere e scrivere i file `/dev` in modo limitato

Principali comandi di UNIX



Gestione di file e directory

Il comando `ls`

.....

ls[- opzioni] name

- **list directory:** Senza opzioni, visualizza (in ordine alfabetico) l'elenco di file e directory presenti nella directory corrente
- **Possiede numerose opzioni**
- **ls -l:** elenca i dettagli dei file (ACL, owner)

Indica che c'è una ACL associata al file

```
%ls -l
```

```
-rwxrwxrwx+ 1 user11 dev 10876 March 16 9:42 miofile
```

0%

ACL

Numero di link
al file

Username

Group owner

n. byte

te Data e ora N
dell'ultima modifica

Nome del file

31

Il comando `ls`: alcune opzioni



- s fornisce la dimensione in blocchi (`size`)
 - t lista nell'ordine di modifica (prima il file modificato per ultimo) (`time`)
 - l un nome per ogni riga
 - C lista su più colonne i nomi dei file
 - 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
 - a lista i file nascosti (quelli il cui nome inizia con `.`)
- ... e molte altre

Il comando `du`



```
du [ options ] [ name... ]
```

`disk usage`: stampa il numero di blocchi contenuti in tutti i files e (ricorsivamente) nelle directory specificate

- se `name` non è specificato, si intende la directory corrente
- `-s`: solo il totale

```
% du
2    ./dir1
2    ./dir2
14
% du -s
198812
```


Il comando `cd`

~~~~~

**cd** [pathname]

- **change dir:** permette di cambiare la directory corrente
  - la directory specificata diviene la working directory
- Senza argomenti, porta alla home directory

```
% pwd
/usr/user11/mia
% cd prog
% pwd
/usr/user11/mia/prog
% cd
% pwd
/usr/user11
```

```
% pwd
/usr/user11/mia
% cd ..
% pwd
/usr/user11
% cd /
% pwd
/
```

# Il comando `mkdir`

```
mkdir [- opzioni] dir ...
```

- **make dir**: permette di creare una nuova directory
- **mkdir -m mode**: consente di creare la directory definendone i permessi di accesso (vedi `chmod`)

```
% pwd
/usr/user11/mia
mkdir A B
% ls
A
B
```

# Il comando `rmdir`

=====

**`rmdir`** [- opzioni] dir ...

- **remove** **dir**: elimina una directory

```
% pwd
/usr/user11/mia
% ls
A
B
% rmdir A
% ls
B
```

# Il comando `mv`

```
mv [ options] filename... target_dir
```

- **move:** muove un file (o directory) sotto la directory target
- se `filename` e `target_dir` non sono nomi di directory, il contenuto di `target_dir` viene sostituito dal contenuto di `filename`

**Se target è una directory:**

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

# Il comando `mv`

=====

## 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
%
```

## Se target non esiste:

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

# Il comando `cp`

```
cp [ options] filename... target_dir
```

- **copy**: copia un file (o directory) sotto la directory target

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

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

# Il comando `rm`

`rm [opzioni] file ...`

- `rm` **remove file**: elimina un file
- `rm -f` : rimuove tutti i file nella directory senza interazione con l'utente
- `rm -i`; permette di eliminare interattivamente un file
- Se il file è un link, `rm` rimuove il link ma non il file puntato

```
% rm -i miofile  
rm: remove miofile ?(Y/N)
```

# Il comando `cat`

```
cat [opzioni] file ...
```

- **Concatenate**: concatena, copia e visualizza file
- Legge ogni file in sequenza e scrive su standard output

```
file1
```

```
Questo è il primo file
```

```
file2
```

```
Questo è il secondo file
```

```
% cat file1 file2
```

```
Questo è il primo file
```

```
Questo è il secondo file
```



# Il comando `chmod`

**`chmod`** `[ugoa][+-][rwx]` `filename`

- Permette di cambiare i permessi di accesso (lettura, scrittura, esecuzione) su un file
- il comando può essere eseguito solo dall'utente proprietario (o dal superuser)

```
% ls -l
-rwxrwxrwx+ 1 user11 dev 10876 March 16 9:42 miofile
% chmod go -w miofile
% ls -l
-rwxr-xr-x+ 1 user11 dev 10876 March 16 9:42 miofile
```

# I comandi `chown` e `chgrp`

=====

`chown newuserid file...`

- **change owner:** cambia l'utente proprietario di un file
- L'utente identificato da `newuserid` diventa il nuovo proprietario dei file
- il comando può essere eseguito solo dal proprietario "cedente" (o dal superuser)

`chgrp newgroupid file...`

- **change group:** cambia l'owner group del file
- Il gruppo identificato da `newgroupid` diventa il nuovo gruppo
- il comando può essere eseguito solo dal proprietario (o dal superuser)

# I metacaratteri

- I metacaratteri rappresentano un meccanismo, fornito dallo shell, che permette di specificare una lista di nomi di file mediante una singola espressione sintetica
  - \* qualsiasi stringa
  - ? qualsiasi carattere
- NB: per sopprimere il significato speciale dei metacaratteri, occorre anteporre \

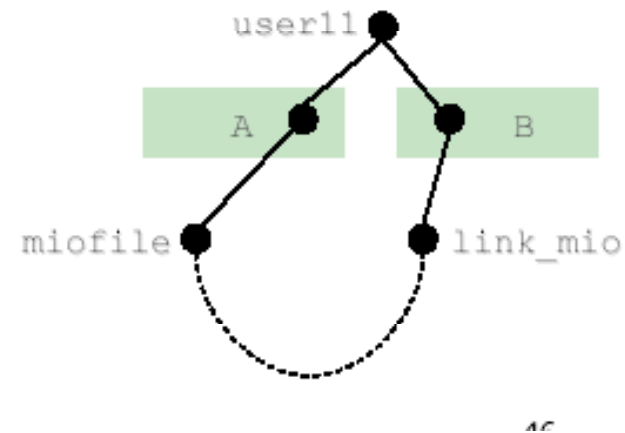
```
% ls -l mi*
-rwxrwxrwx+ 1 user11 dev 10876 March 16  9:42 miofile
-rwxrw-rw-+ 1 user11 dev 10876 March 16 10:40 mioprogram
-rw-rw-rw-+ 1 user11 dev 10876 March 16 19:22 mio
% ls -l mi?
-rw-rw-rw-+ 1 user11 dev 10876 March 16 19:22 mio
```

# Il comando `ln`

`ln file_esistente nuovo`

- associa un nuovo nome (link) al file (esistente), che non può essere una directory
- Consente di condividere un file, creando un link ad esso
  - Non distingue fra maiuscole e minuscole
- Incrementa di 1 il contatore di link

```
% ls -l
-rwxrwxrwx 1 . . . . . miofile
% cd B
% ln ../A/miofile link_mio
-rw-rw-rw- 2 . . . . . link_mio
% cd ../A
% ls -l
-rwxrwxrwx 2 . . . . . miofile
```



# Il comando `touch`

```
touch [ options ][ time ] filename...
```

- aggiorna la data e l'ora dell'ultimo accesso (opzione `-a`) o dell'ultima modifica (opzione `-m`) di filename (default: `-am`)
- se `time` non è specificato, usa la data e l'ora corrente
- se il file non esiste, lo crea

```
% touch 01281738 file1
% ls -l
total 0
----r--r-- 1 user11 usrm1 0 Jan 28 17:38
file1
%
```

# Il comando `find`

```
find pathname... [ expression]
```

- discende ricorsivamente le directories specificate ( `pathname...`), cercando tutti i files che rendono vera la `expression` booleana, costituita da specificati attributi (filename, tipo, permessi, proprietario, gruppo, numero di links, dimensione, tempo dell'ultima modifica/accesso ...)
  - and, or, not di attributi

```
% find . -name a.out -atime +7
```

Trova tutti i file nella working directory (o più sotto) di nome `a.out` il cui ultimo accesso è anteriore a 7 giorni

# Uso combinato del comando `find`: esempio

Eliminare tutti i file nella working directory (o più sotto) di nome `a.out` il cui ultimo accesso è anteriore a 7 giorni

