

Il file system di Unix

Caratteristiche generali dei FS
comunemente usati da Unix/Linux

Il file system di Unix

- Il file system è la parte del SO che si occupa di mantenere i dati/programmi in modo persistente
- Astrazioni fornite :
 - *File* : unità di informazione memorizzata in modo persistente
 - *Directory* : astrazione che permette di raggruppare assieme più file

I file di Unix

- Tipi di file Unix :
 - *regular* (-): collezione di byte non strutturata
 - *directory* (d) : directory
 - *buffered special file* (b) : file che rappresenta una periferica con interfaccia a blocchi
 - *unbuffered special file* (b) : file che rappresenta una periferica con interfaccia a caratteri
 - *link simbolico* (l) : file che rappresenta un nome alternativo per un altro file X, ogni accesso a questo file viene ridiretto in un accesso a X

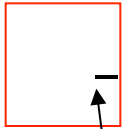
I file di Unix (2)

- Tipi di file Unix (cont.):
 - *pipe* (p): file che rappresenta una pipe
 - *socket* (s) : file che rappresenta un socket

Attributi di un file Unix

- File = nome + dati + attributi
- Alcuni attributi dei file unix :

– es. `ls -l pippo.c`



```
-rw-r--r-- 1 tizio users 1064 Feb 6 2002 pippo.c
```

Tipo del file
(regolare, -)

Attributi di un file Unix (2)

- File = nome + dati + attributi
- Alcuni attributi dei file unix :

```
- es. ls -l pippo.c  
-rw-r--r-- 1 tizio users 1064 Feb 6 2002 pippo.c
```



Protezione

r - permesso di lettura (directory, listing)

w- permesso di scrittura (directory, aggiungere file)

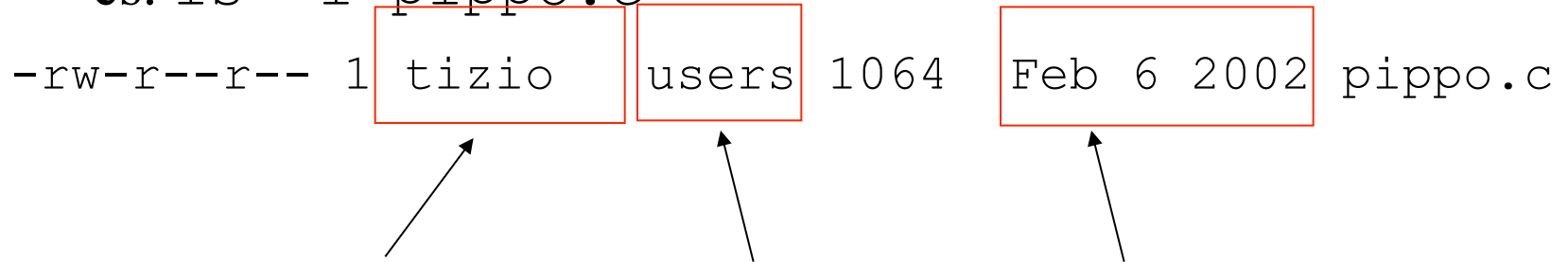
x - permesso di esecuzione (directory, accesso)

Attributi di un file Unix (3)

- File = nome + dati + attributi
- Alcuni attributi dei file unix :

– es. `ls -l pippo.c`

```
-rw-r--r-- 1 tizio users 1064 Feb 6 2002 pippo.c
```



Proprietario del file

Gruppo

Data ultima modifica

Attributi di un file Unix (4)

- File = nome + dati + attributi
- Alcuni attributi dei file unix :

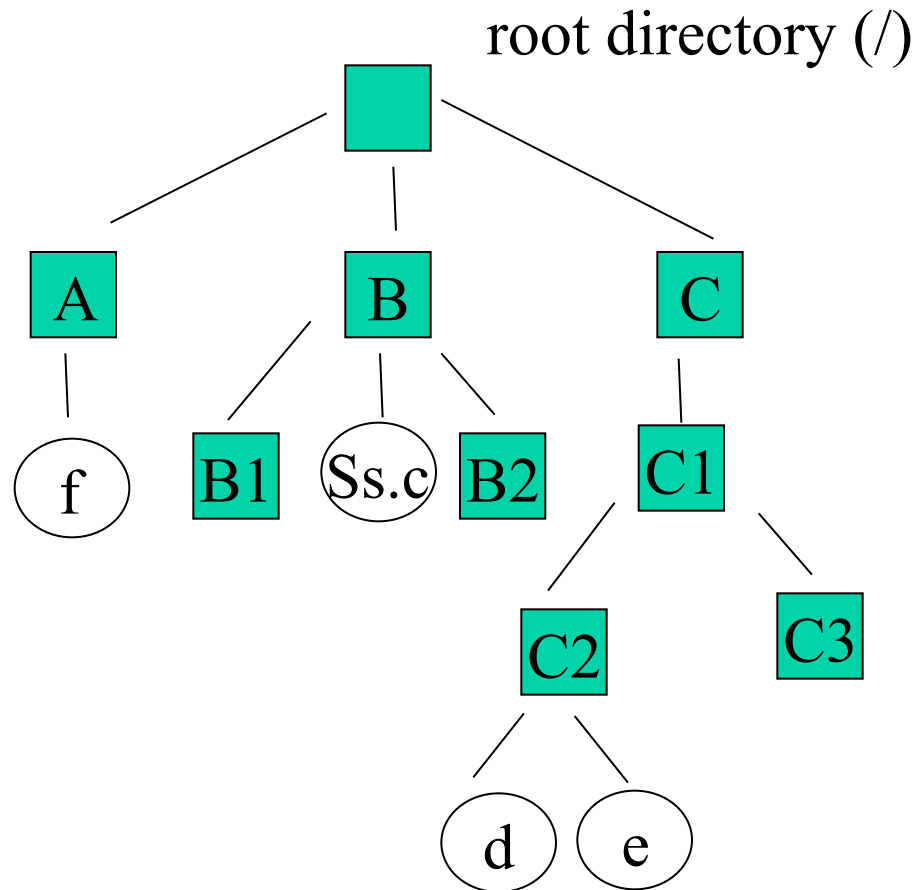
– es. `ls -l pippo.c`

```
-rw-r--r-- 1 susanna users 1064 Feb 6 2002 pippo.c
```

Numero di blocchi su disco utilizzati

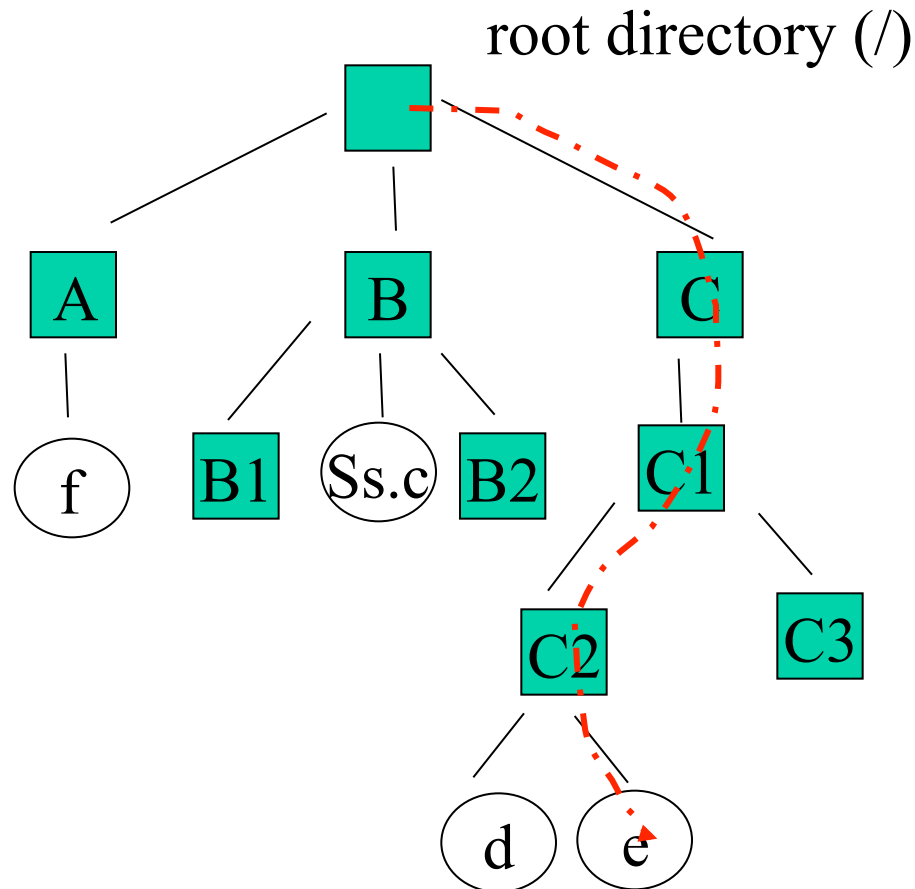
Lunghezza in byte
del file

Il FS di Unix è gerarchico



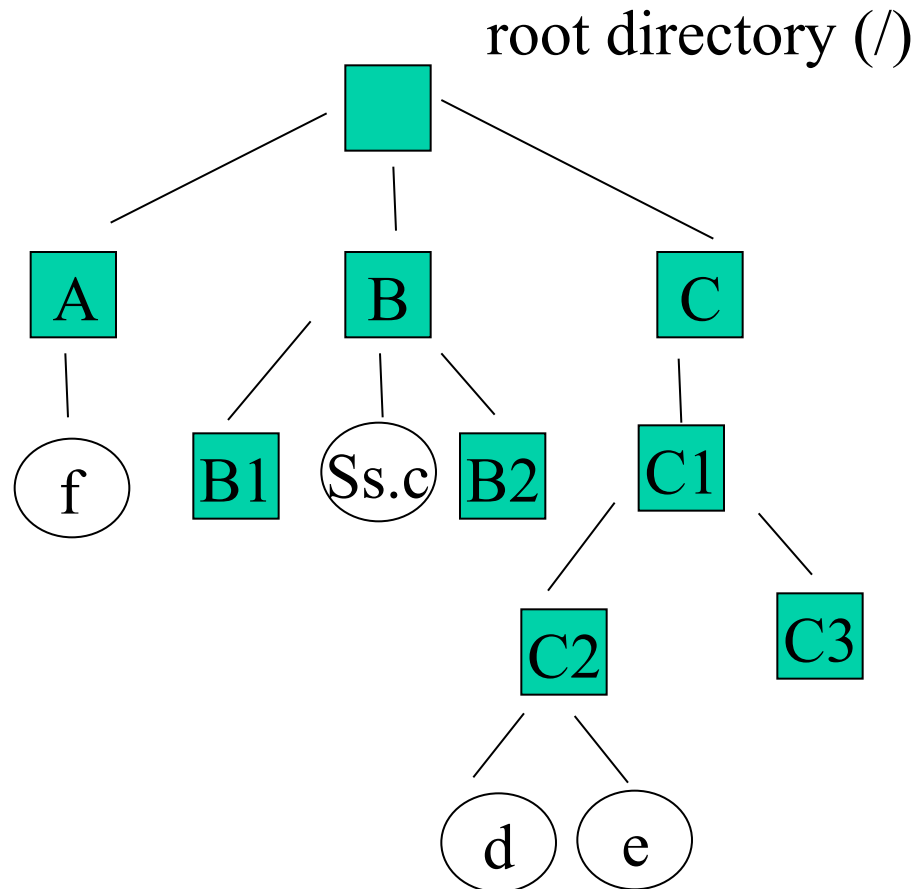
- Esempio di FS senza link file

Path name assoluto



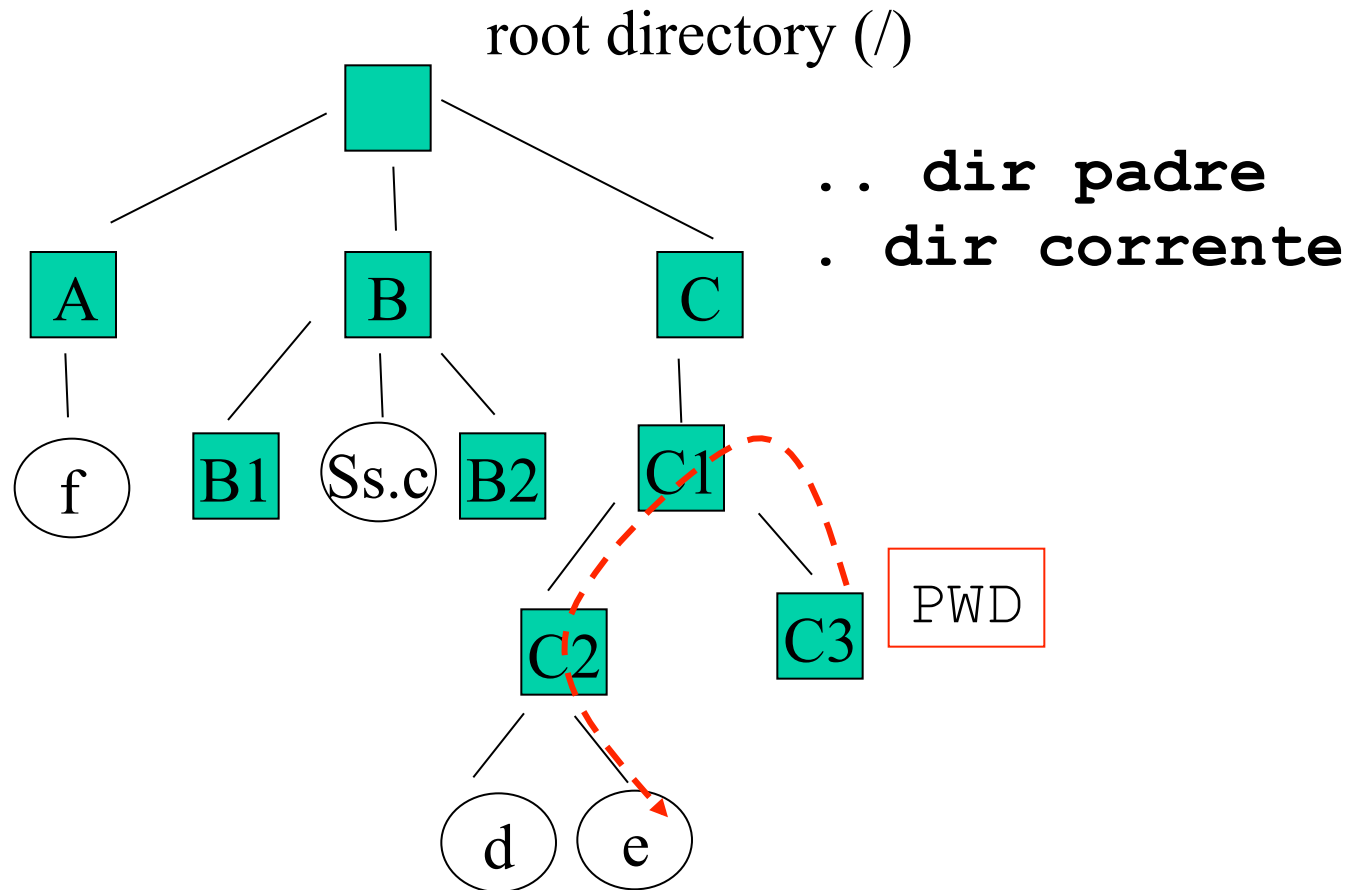
- Ogni file è univocamente determinato dal cammino che lo collega alla radice
 - /C/C1/C2/e

Path name relativo



- Ogni shell ha associata una *working directory*
 - è indicata nella var di ambiente `PWD`
 - si cambia con `cd`

Path name relativo (2)



- Il PNR è il cammino dalla Working Directory
 - . / . . / C2 / e (il '.' iniziale si può omettere)

Implementazione del FS di Unix

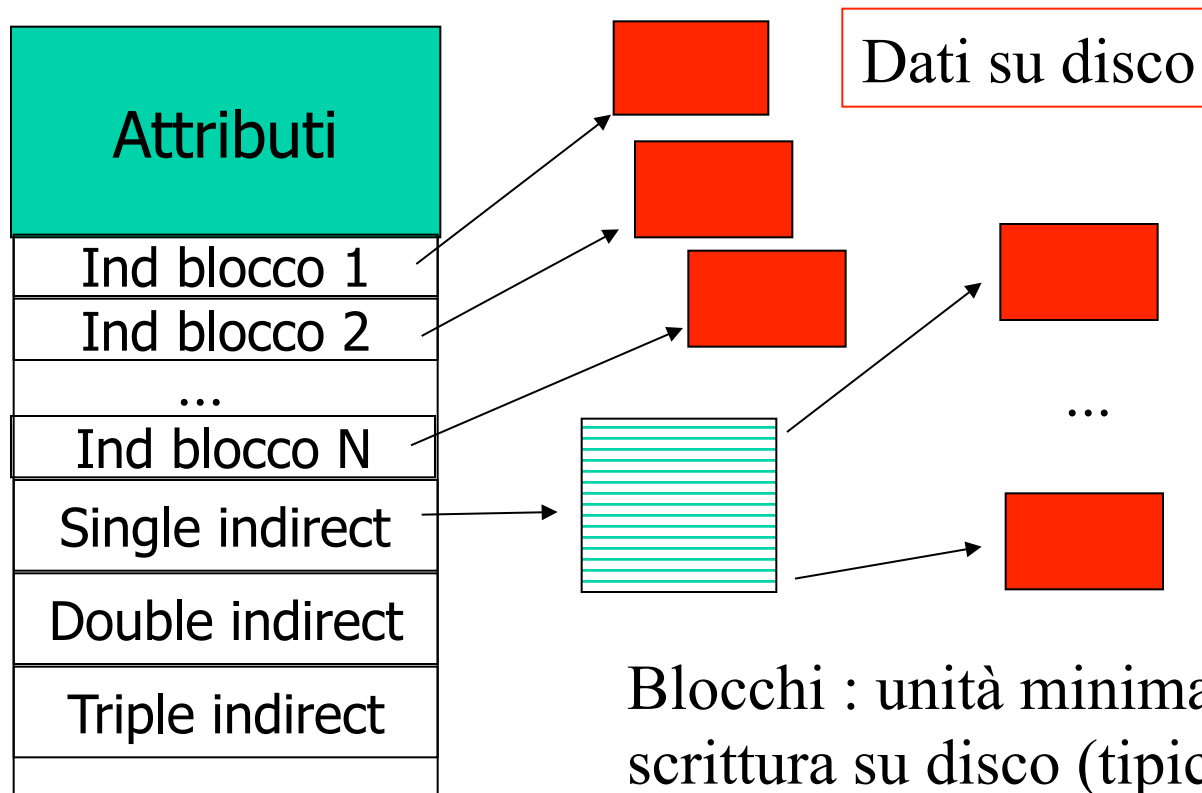
- Ogni file è rappresentato da un i-node.
- Cosa contiene un i-node:
 - tipo di file `-`, `d`, `l` ...
 - modo, bit di protezione (r-w-x)
 - uid, gid : identificativo utente e gruppo
 - size, tempi di creazione, modifica etc
 - campo count per i *link hard*

Implementazione del FS di Unix (2)

- Cosa contiene un i-node :
 - *file regular, directory* :
 - indirizzo dei primi 10 blocchi su disco
 - indirizzo di uno o più blocchi indiretti
 - *device file* : major number, minor number (identificatore del driver e del dispositivo)
 - *link simbolico* : path del file collegato

Implementazione del FS di Unix (2)

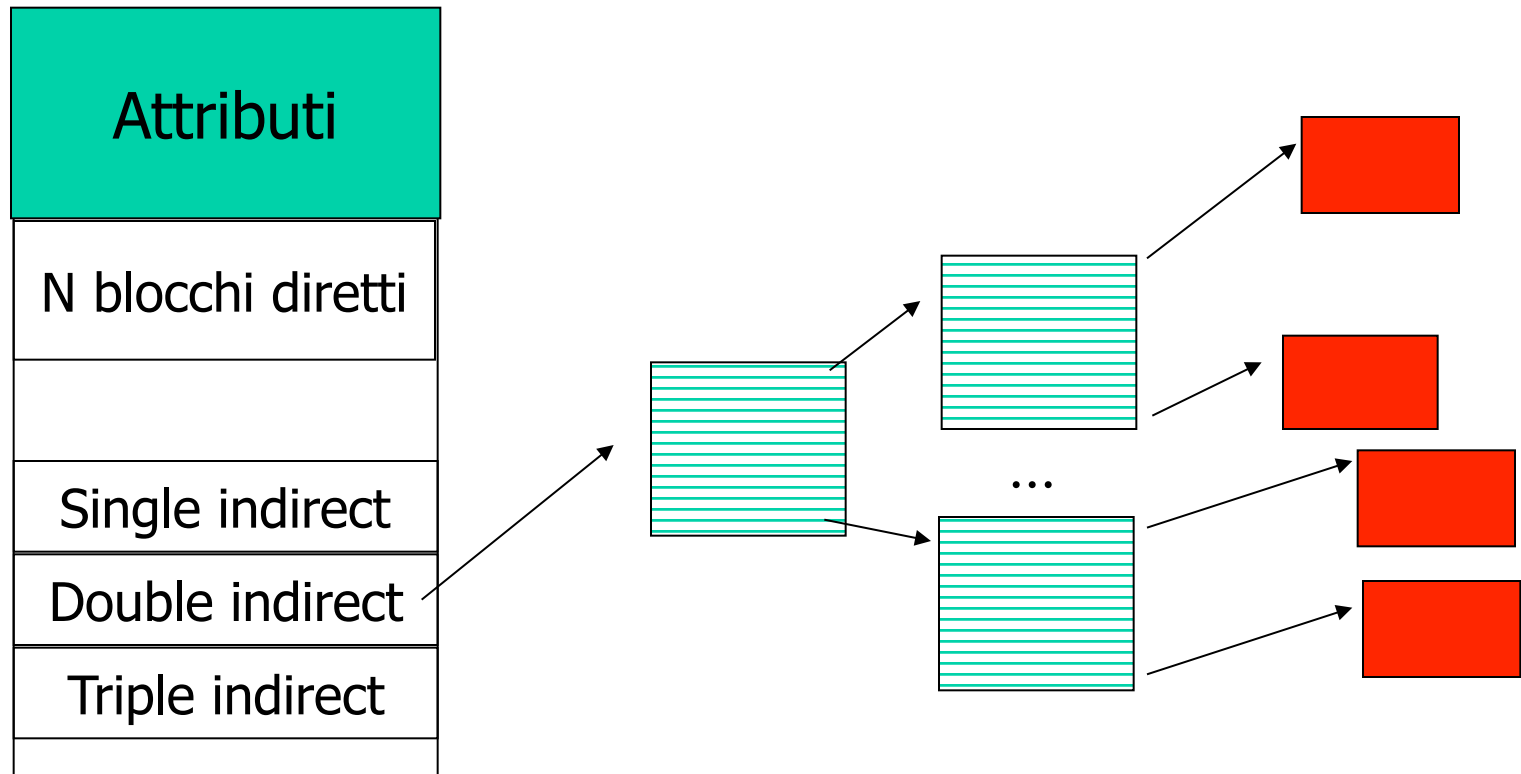
- *i-node* di un file regolare



Blocchi : unità minima di lettura
scrittura su disco (tipico 1-4KB)
Ind. Blocco : tipicamente 4 byte

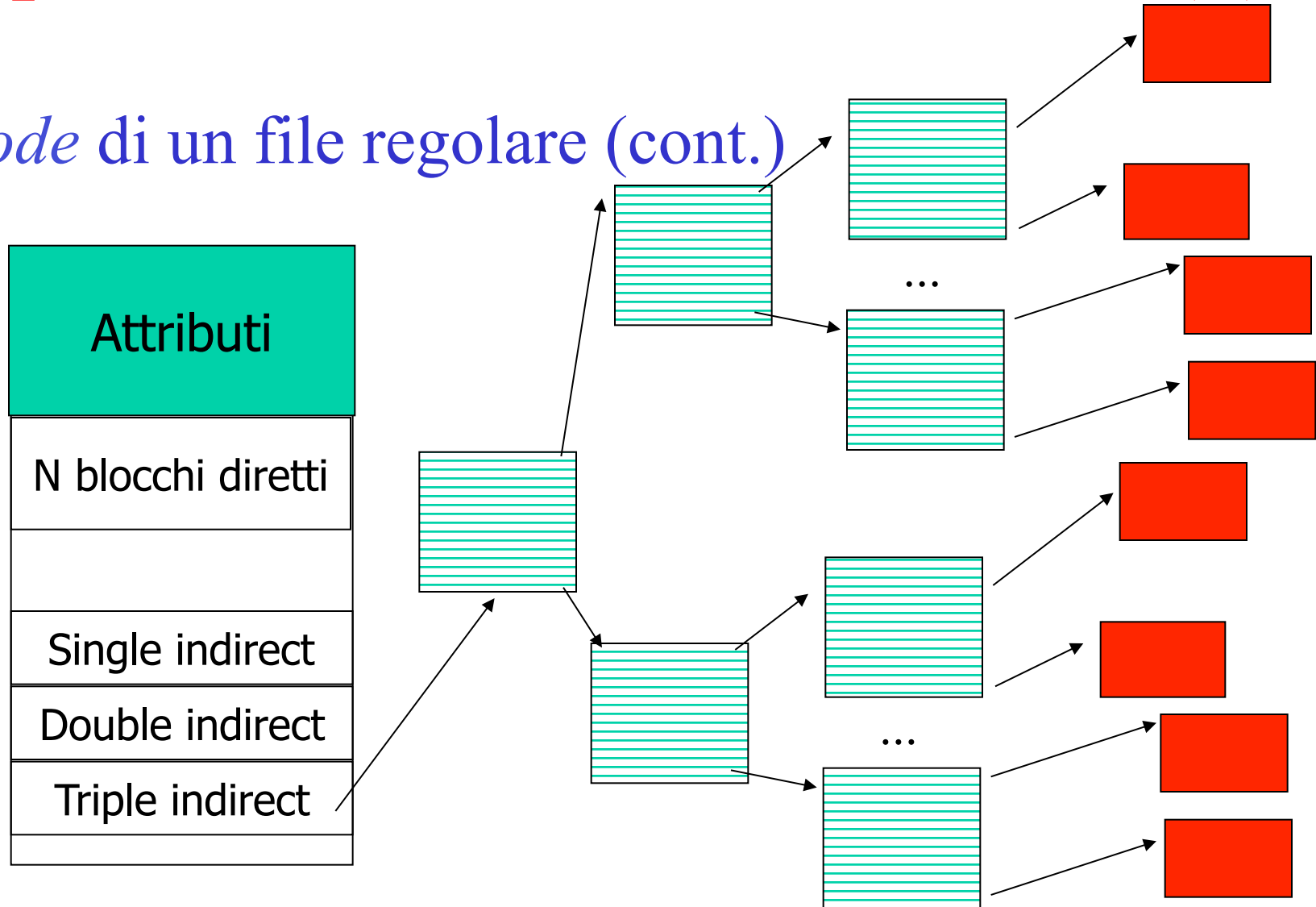
Implementazione del FS di Unix (3)

- *i-node* di un file regolare (cont.)



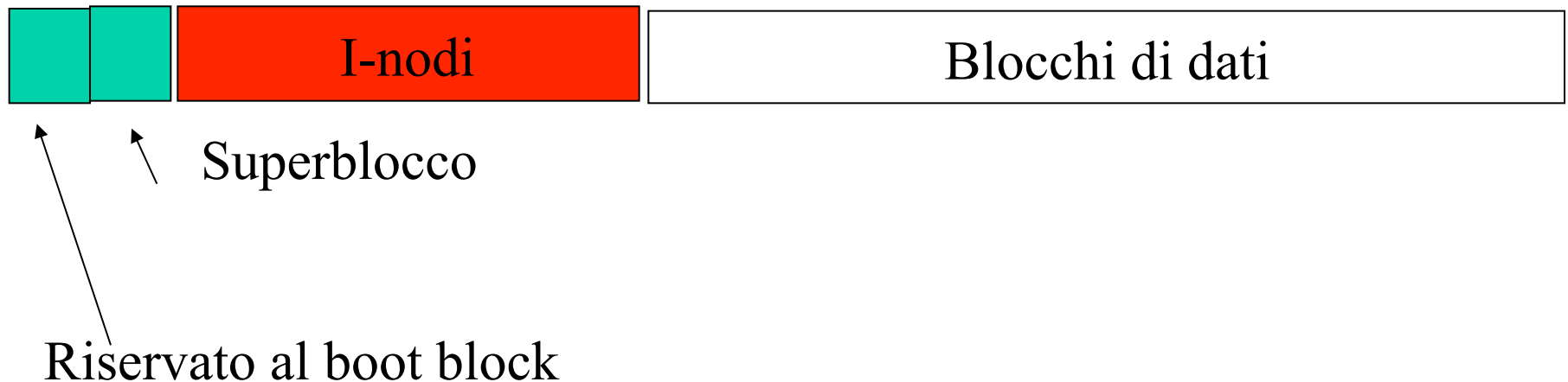
Implementazione del FS di Unix (4)

- i-node* di un file regolare (cont.)



Implementazione del FS di Unix (5)

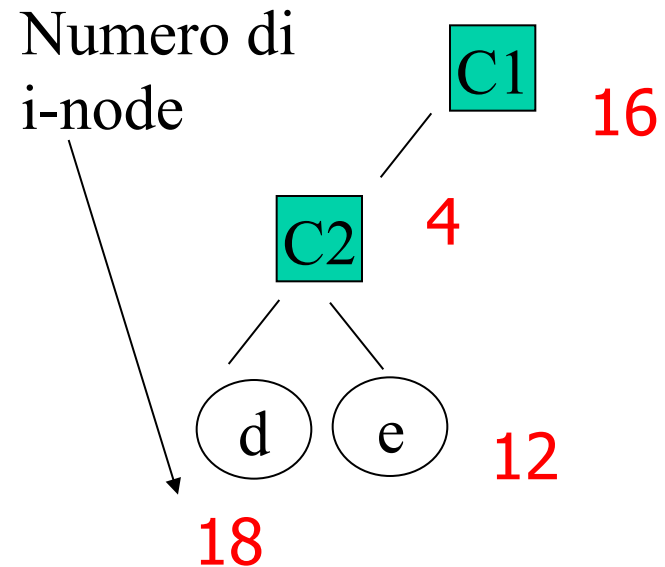
Organizzazione di una partizione in un file system tipico UNIX



Implementazione del FS di Unix (6)

Organizzazione dei blocchi dati di una directory (Unix V7)

4	.(punto)
16	..(punto punto)
12	e
18	d



Blocco dati relativo alla directory C2

Implementazione del FS di Unix (7)

Root directory
(/) (RAM)

1	.
1	..
4	bin
7	dev
6	usr

I-node 6
(/usr)

Attr.
132

132 è
il primo
blocco
dati

Blocco 132
(dati di /usr)

6	.
1	..
19	ast
51	rd
26	sp

I-node 26
(/usr/sp)

Attr.
406

406 è
il primo
blocco
dati

Blocco 406
(dati di /usr/sp)

26	.
6	..
64	mbox
58	tmp
86	bin

I passi necessari per leggere */usr/sp/mbox*

Implementazione del FS di Unix (8)

Root directory
(/) (RAM)

1	.
1	..
4	bin
7	dev
6	usr

I-node 6
(/usr)

Attr.
132

132 è
il primo
blocco
dati

Blocco 132
(dati di /usr)

6	.
1	..
19	ast
51	rd
26	sp

I-node 26
(/usr/sp)

Attr.
406

406 è
il primo
blocco
dati

Blocco 406
(dati di /usr/sp)

26	.
6	..
64	mbox
58	tmp
86	bin

I passi necessari per leggere */usr/sp/mbox*

Tabelle di nucleo relative ai file

- Rappresentazione di un file aperto (subito dopo la `open()`)

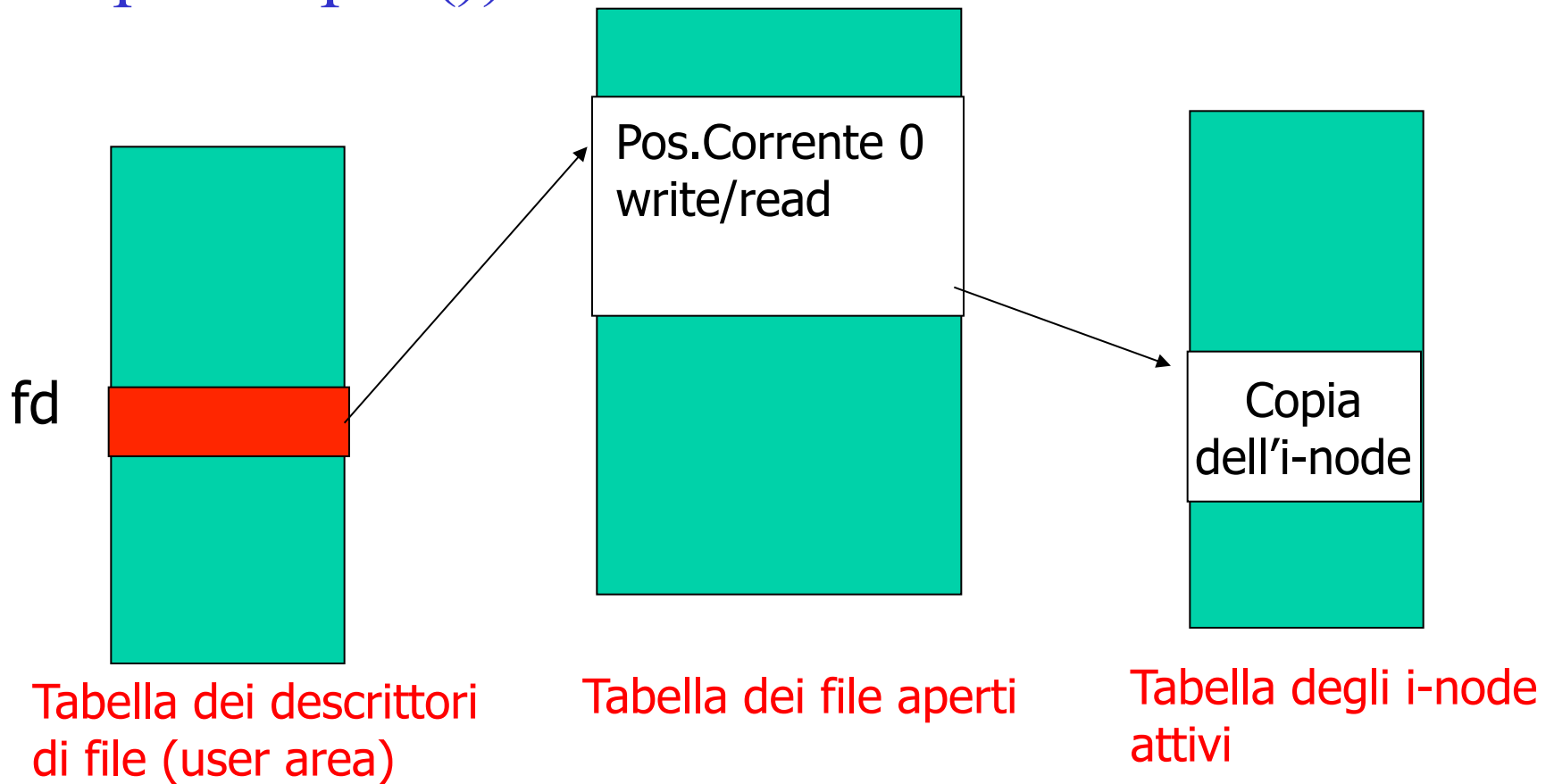


Tabelle di nucleo relative ai file (2)

- Perché 3 diverse ?

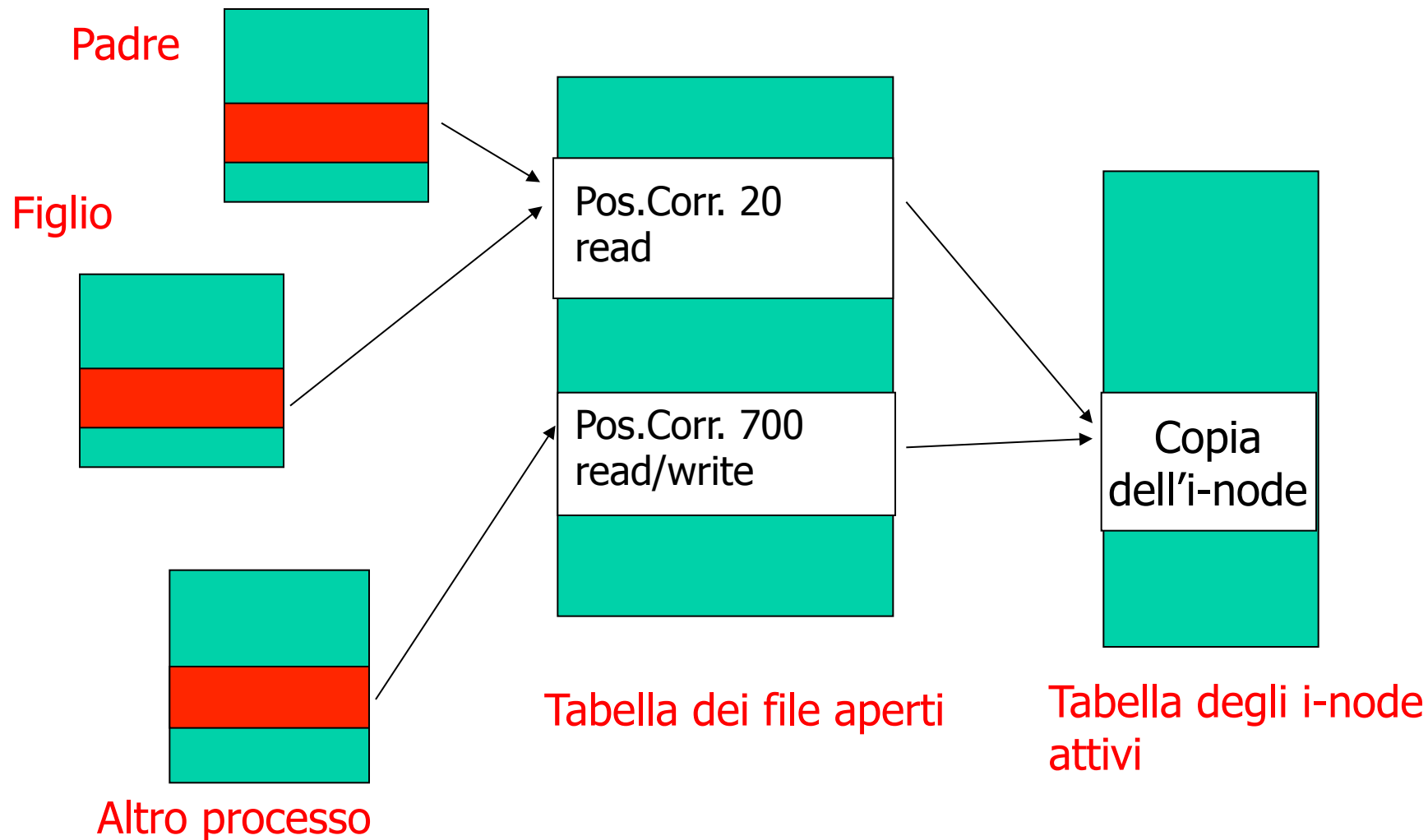
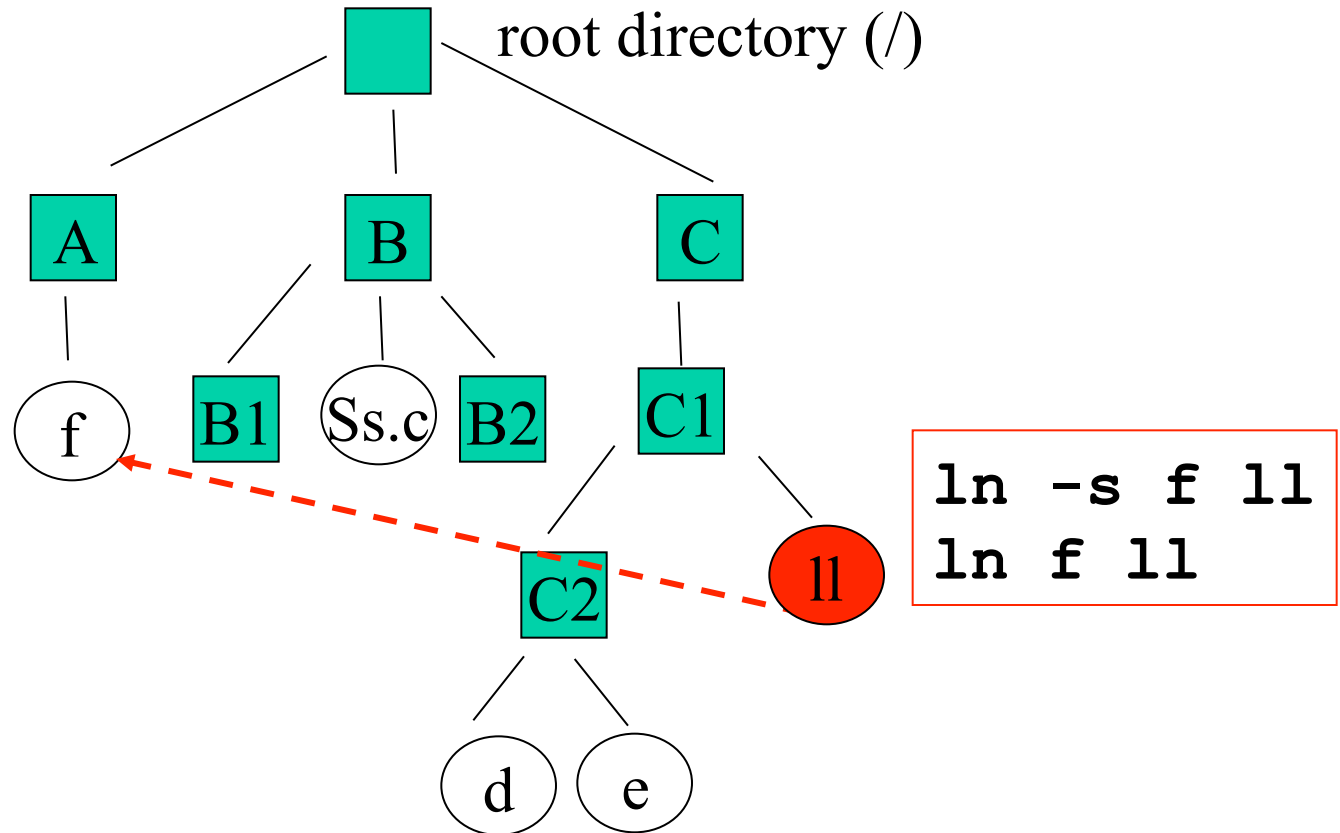


Tabelle di nucleo relative ai file (3)

- Perché 3 diverse ?
 - È necessario avere gli i-nodi attivi in una tabella in RAM per ottimizzare le prestazioni
 - Usando la Tabella dei File Aperti è possibile avere più processi che accedono allo stesso file con '*position counter*' indipendenti
 - Più processi possono condividere la stessa visibilità del file (padri e figli ...)

Condivisione di file : Link

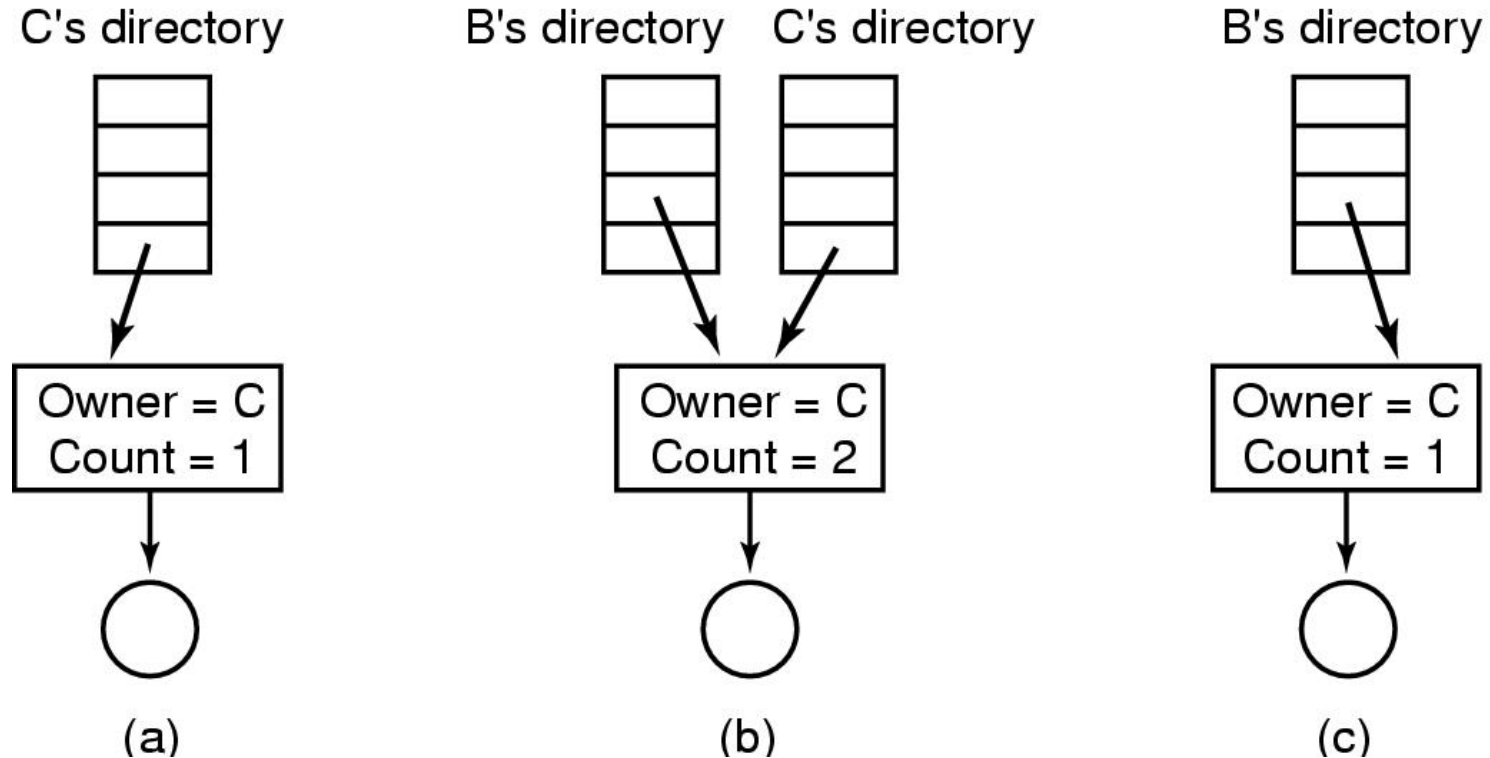


- Forniscono path name alternativi per lo stesso file
 - /C/C1/ll oppure /A/f

Link (2)

- Hard link : `ln f ll`
 - le due directory condividono la struttura dati relativa al file (i-node)
 - paradosso della rimozione da parte dell' owner
- Symbolic Link : `ln -s f ll`
 - la seconda directory contiene un file speciale (LINK) con il path name del file condiviso
 - accesso più lento (il path name deve essere seguito ogni volta che accediamo al file)

Link (3)

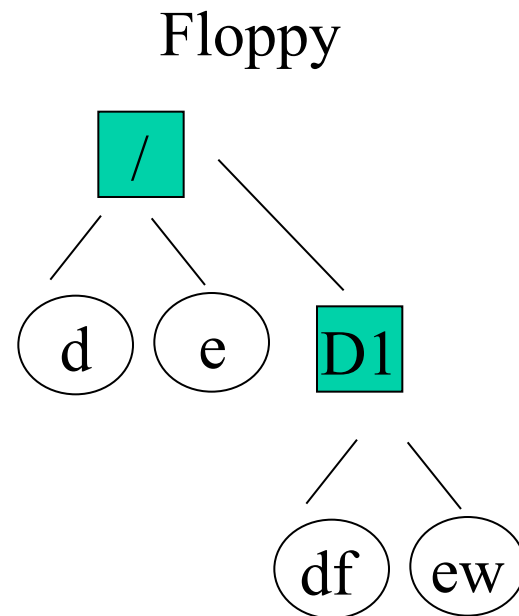
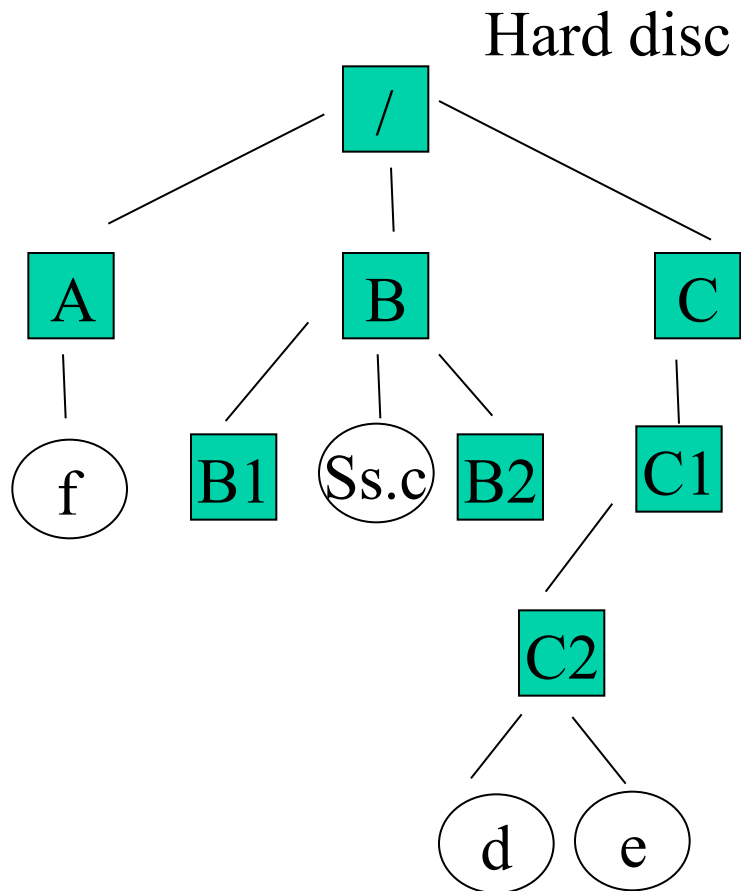


(a) situazione precedente al linking (hard)

(b) dopo la creazione del link

(c) dopo che l' *owner* originale ha rimosso il file

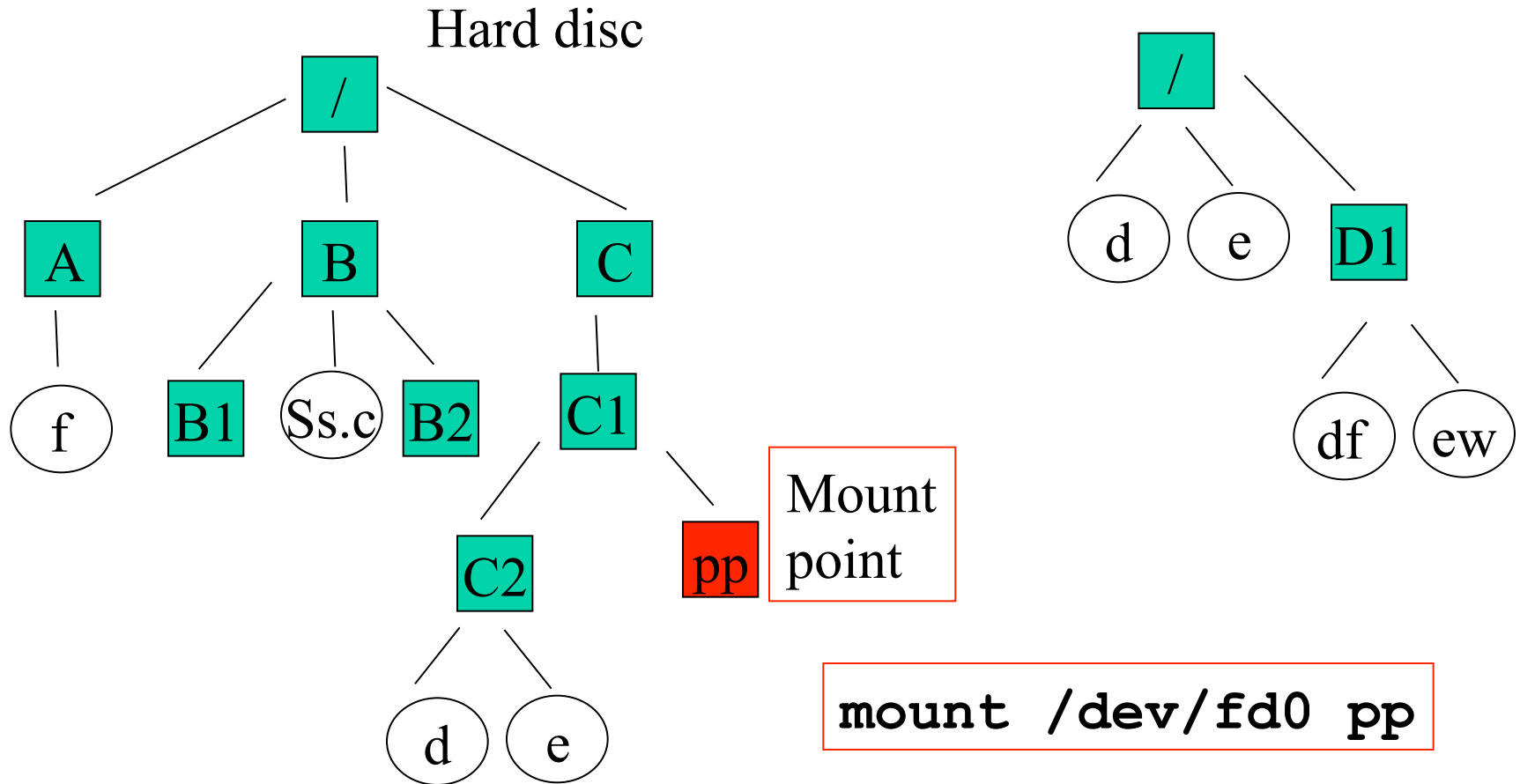
Mounting



`mount -t type dev dir`

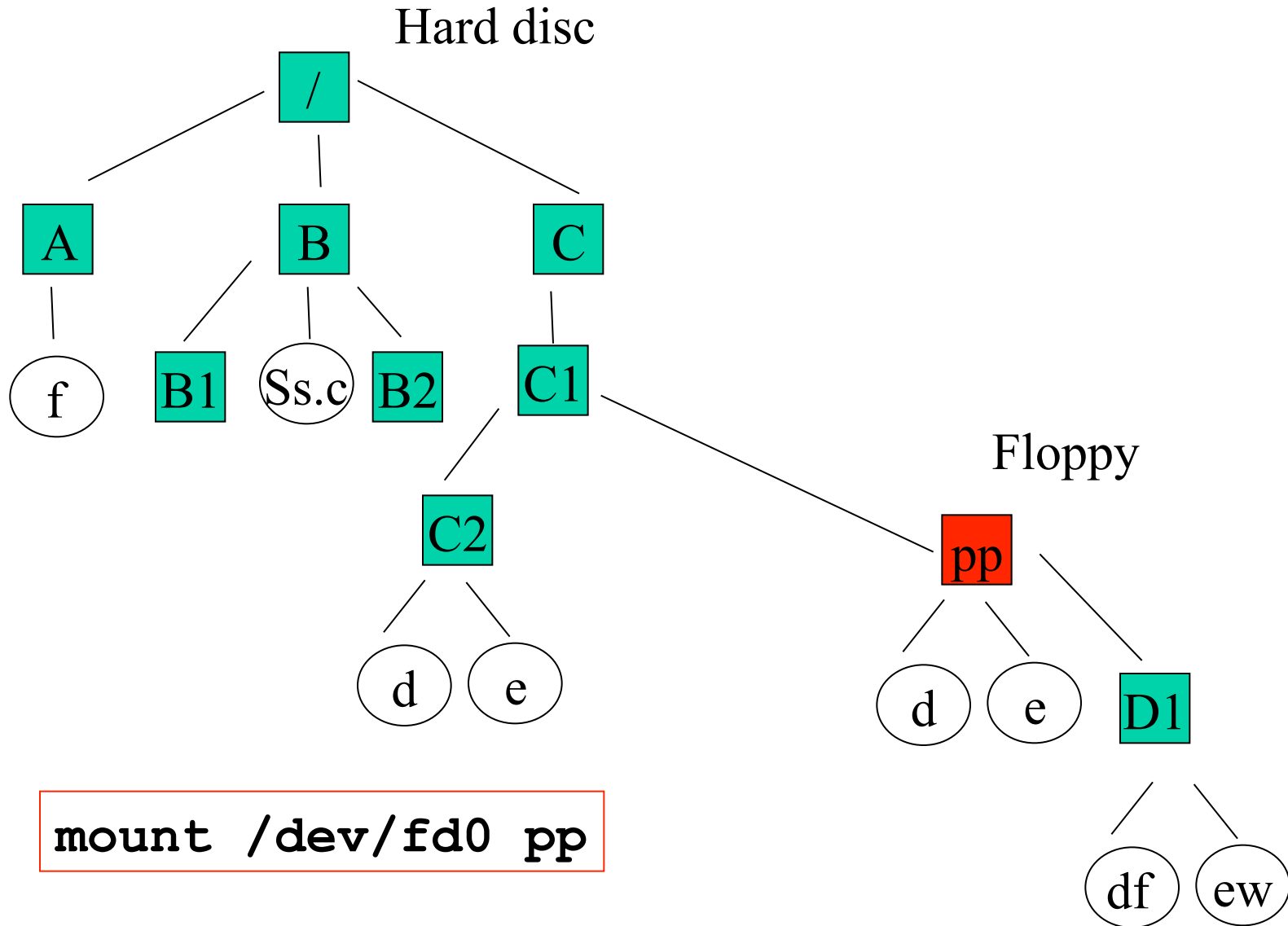
- Permette di unire in un unico albero file system di tipo diverso memorizzati su dispositivi diversi

Mounting (2)

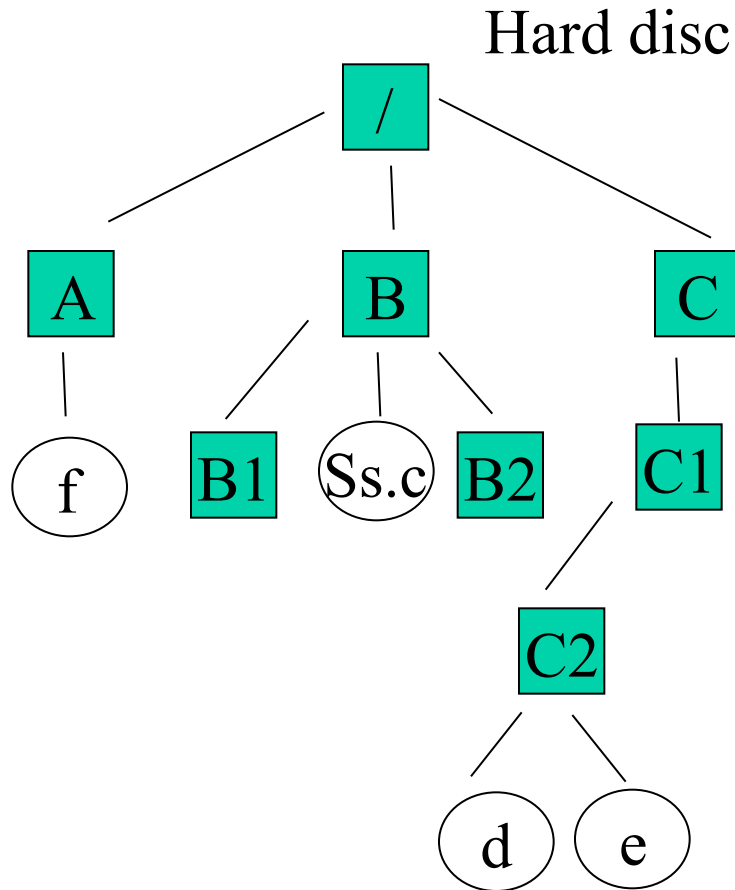


- Permette di unire in un unico albero file system di tipo diverso memorizzati su dispositivi diversi

Mounting (3)

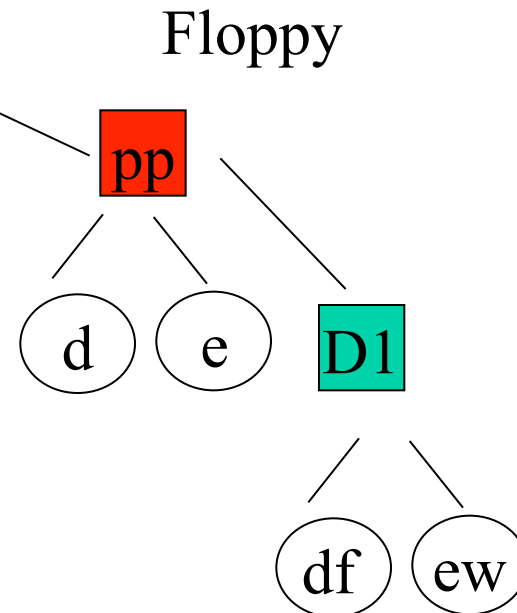


Mounting (4)



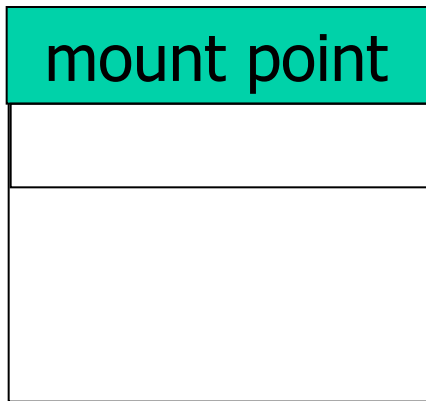
Problemi:

- 1) Cosa contiene l'i-node di pp ?
- 2) Come si segue un path name che contiene pp ?



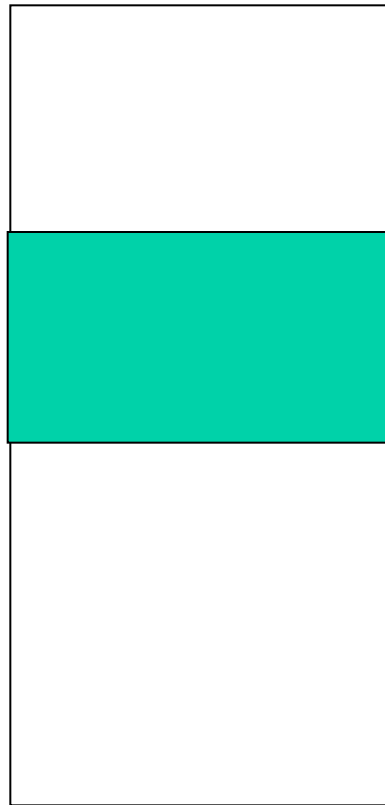
```
mount /dev/fd0 pp
```

Mounting (5)



I-nodo di pp

X



Mount table

In MT(x) :

- device (**/dev/fd0**)
- puntatore all'i-nodo della root del FS montato
- puntatore all'i-nodo del mount point (**pp**)
- puntatore alla copia del supeblock di FS in RAM

Lock dei file

- È possibile definire dei lock su (parti di) un file ed usarli per sincronizzare gli accessi
- Varie opzioni (dipendono dalla versione)
- *lock advisory o mandatory*
 - Il controllo del lock è a carico dell'utente (adv)
 - Il kernel conosce l'esistenza del lock e controlla gli accessi (mandatory)
- *lock shared o exclusive*
 - più processi possono accedere contemporaneamente (shared)
 - può accedere un processo alla volta (exclusive)

SC per la gestione dei file

System call	Description
<code>fd = creat(name, mode)</code>	One way to create a new file
<code>fd = open(file, how, ...)</code>	Open a file for reading, writing or both
<code>s = close(fd)</code>	Close an open file
<code>n = read(fd, buffer, nbytes)</code>	Read data from a file into a buffer
<code>n = write(fd, buffer, nbytes)</code>	Write data from a buffer into a file
<code>position = lseek(fd, offset, whence)</code>	Move the file pointer
<code>s = stat(name, &buf)</code>	Get a file's status information
<code>s = fstat(fd, &buf)</code>	Get a file's status information
<code>s = pipe(&fd[0])</code>	Create a pipe
<code>s = fcntl(fd, cmd, ...)</code>	File locking and other operations

- **s** è un codice di errore
- **fd** è un descrittore di file
- **position** è un offset all' interno del file

La System Call stat

Device the file is on
I-node number (which file on the device)
File mode (includes protection information)
Number of links to the file
Identity of the file's owner
Group the file belongs to
File size (in bytes)
Creation time
Time of last access
Time of last modification

e tipo file

I campi ritornati dalla system call stat

Le SC per la gestione delle directoty

System call	Description
<code>s = mkdir(path, mode)</code>	Create a new directory
<code>s = rmdir(path)</code>	Remove a directory
<code>s = link(oldpath, newpath)</code>	Create a link to an existing file
<code>s = unlink(path)</code>	Unlink a file
<code>s = chdir(path)</code>	Change the working directory
<code>dir = opendir(path)</code>	Open a directory for reading
<code>s = closedir(dir)</code>	Close a directory
<code>dirent = readdir(dir)</code>	Read one directory entry
<code>rewinddir(dir)</code>	Rewind a directory so it can be reread

- **s** è un codice di errore
- **dir** identifica una directory
- **dirent** è un elemento di una directory

System Call per la protezione dei file

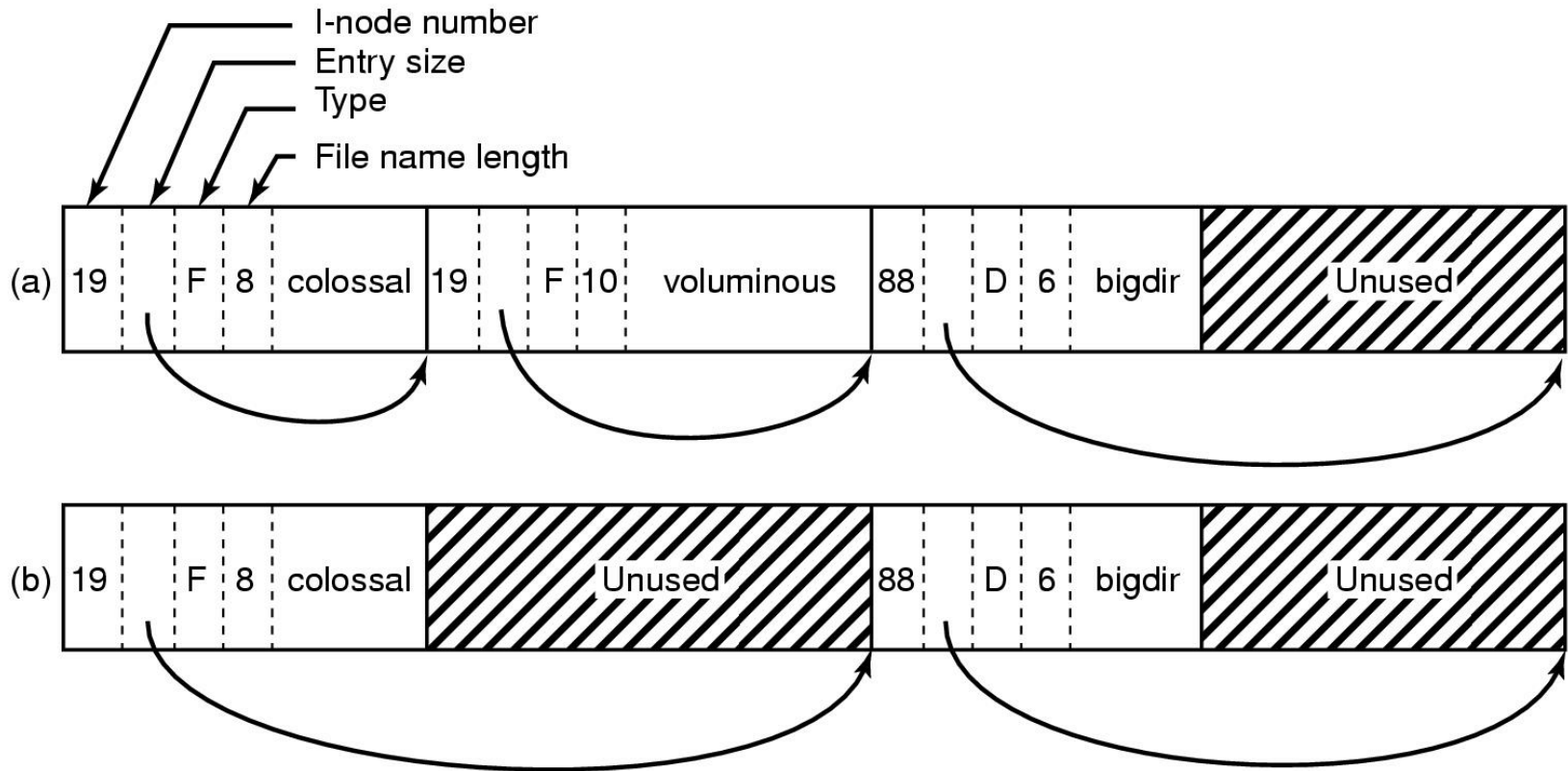
System call	Description
s = chmod(path, mode)	Change a file's protection mode
s = access(path, mode)	Check access using the real UID and GID
uid = getuid()	Get the real UID
uid = geteuid()	Get the effective UID
gid = getgid()	Get the real GID
gid = getegid()	Get the effective GID
s = chown(path, owner, group)	Change owner and group
s = setuid(uid)	Set the UID
s = setgid(gid)	Set the GID

- s è un codice di errore
- uid e gid sono *user e group identifier* (UID e GID)

Il Fast File System di Berkley

- In Unix V 7, i nomi dei file erano limitati a 14 caratteri
 - ogni elemento della directory 14+2 byte (per il numero di i-node)
 - in questo modo la directory è un array regolare
- FFS Permette nomi di file di 256 byte ed usa un formato più complesso
- Usando opendir(), closedir(), readdir() e rewinddir() i programmi sono indipendenti dal formato interno

Il Fast File System di Berkley (2)

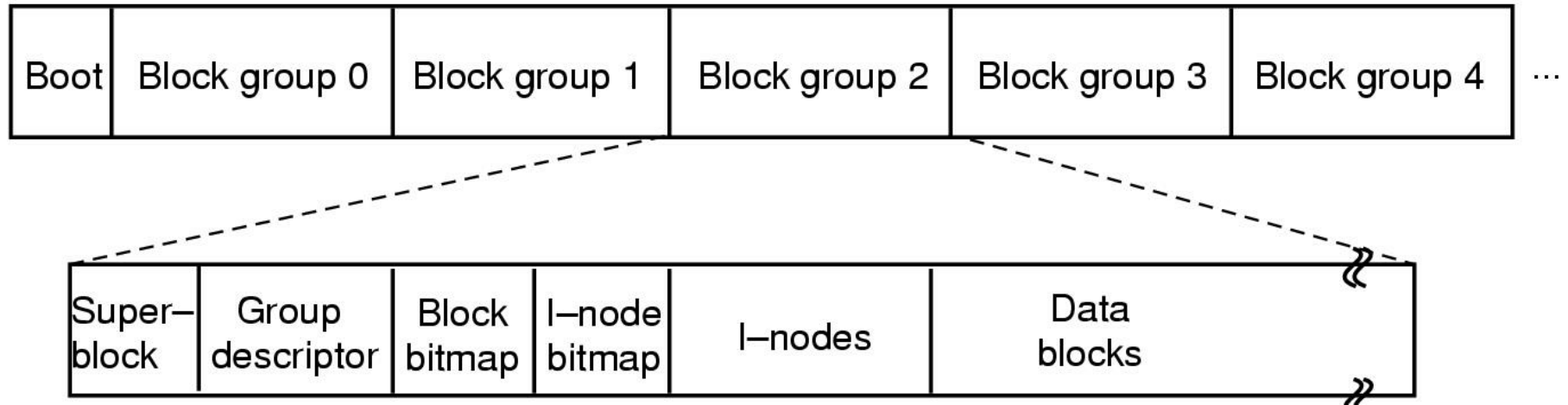


- Una directory BSD con 3 file
- La stessa directory dopo che il file *voluminous* è stato rimosso

Il Fast File System di Berkley (3)

- Caching dei nomi dei file
 - per evitare lunghe ricerche nelle directory
- Divisione del disco in gruppi di cilindri
 - ciascuno con superblock, inode e blocchi
- Due diverse ampiezze di blocco
 - per file grandi e piccoli

Il File System Ext2 di Linux



Organizzazione del file system Ext2 :

- group descriptor : indirizzo delle bitmap del gruppo, numero di directory, i-node e blocchi, indirizzo del primo i-node
- le directory sono distribuite uniformemente fra i gruppi