

# Il file system di Unix

Caratteristiche generali dei FS  
comunemente usati da Unix/Linux

1

# 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

2



# 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 (dati letti come byte)
  - *unbuffered special file* (b) : file che rappresenta una periferica con interfaccia a caratteri (dati letti come 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

File virtuali  
per gestione  
periferiche  
(S.A.) usati  
per connessione.  
I/O = lettura  
e scrittura  
di file.

op. di lettura e  
scrittura riguardano sempre  
block di byte

→ può essere letto dalla  
file

3

# I file di Unix (2)

- Tipi di file Unix (cont.):

- *pipe* (p) : file che rappresenta una pipe (file temporaneo per pipe e fifo)

- *socket* (s) : file che rappresenta un socket

↓  
op. di trasf. tra due macchine.  
IP e mm.

4

## Attributi di un file Unix

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

- es. ls -l pippo.c



### Protezione

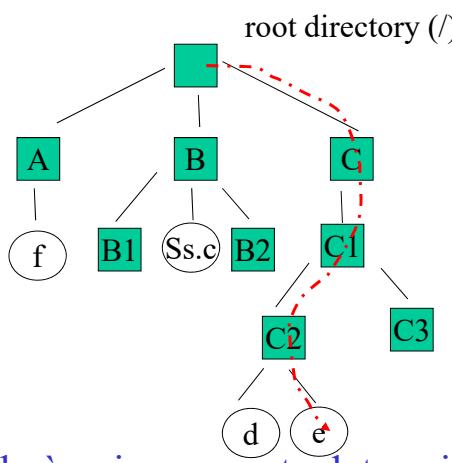
r - permesso di lettura (directory, listing)

w - permesso di scrittura (directory, aggiungere file)

x - permesso di esecuzione (directory, accesso)

5

## Path name assoluto



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

6

Più che altro grafo acchilico perché nello stesso file posso uscire a percorsi diversi.

## Implementazione del FS di Unix

- Ogni file è rappresentato da un i-node.
- Cosa contiene un i-node:
  - tipo dl 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

7

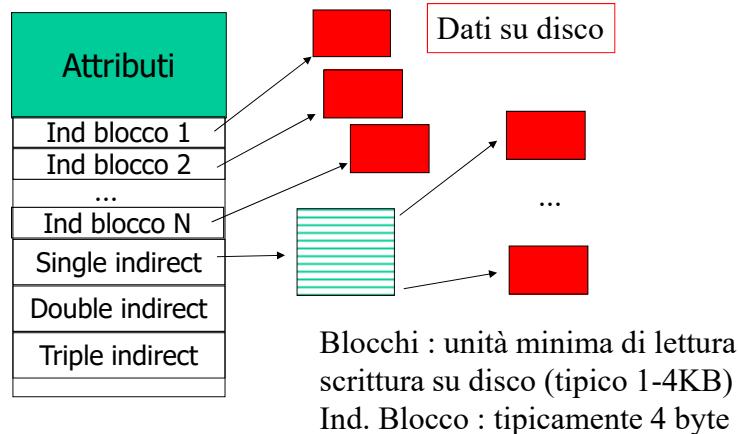
## Implementazione del FS di Unix (2)

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

8

## Implementazione del FS di Unix (2)

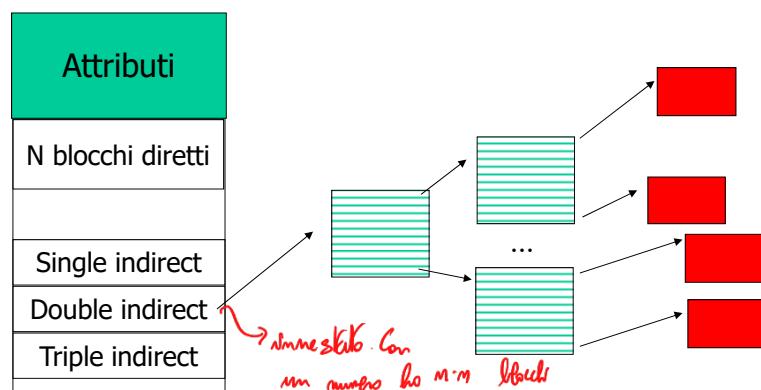
- *i-node* di un file regolare



9

## Implementazione del FS di Unix (3)

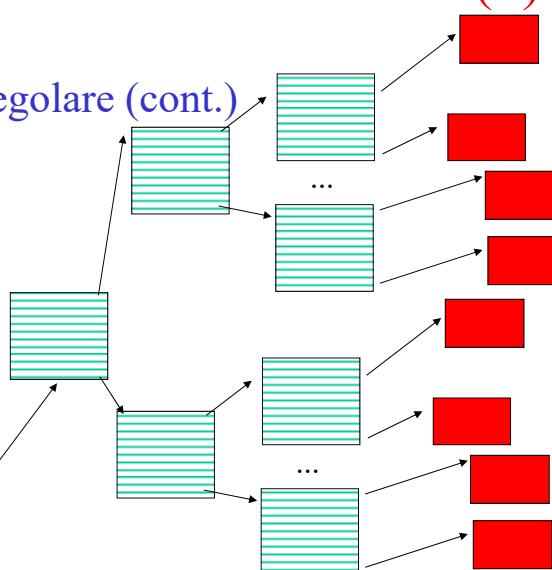
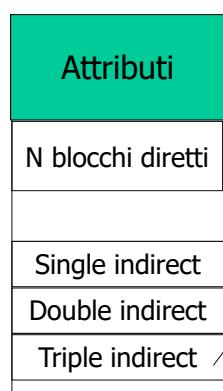
- *i-node* di un file regolare (cont.)



10

## Implementazione del FS di Unix (4)

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



11

## Implementazione del FS di Unix (5)

Organizzazione di una partizione in un file system tipico UNIX

Pezzo iniziale della partizione del disco



Superblocco  
(contiene informazioni critiche per il file system:  
num. di i-node, num di blocchi del disco)

Riservato al boot block

(può contenere il codice di avvio)

consente di fare il bootstrap

Là viene caricato il file mbfs.sys per caricare OS

12

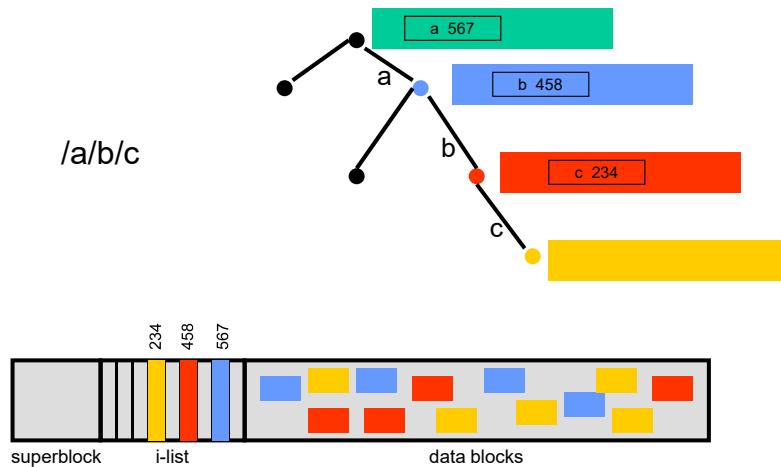
Se so da dove parte l'indirizzo di un i-node, posso recuperare un i-node con numero perché l'ultimo i-node è falso

- Come so se un i-node è libero o occupato? Devo sapere se è stato assegnato a un file o

Se non,

Prima 3 punti liberi sono parte più critica.

## Struttura generale file system

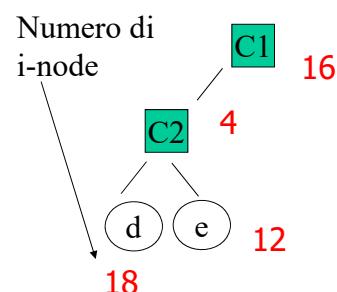


13

## Implementazione del FS di Unix (6)

Organizzazione dei blocchi dati di una directory (Unix V7)

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



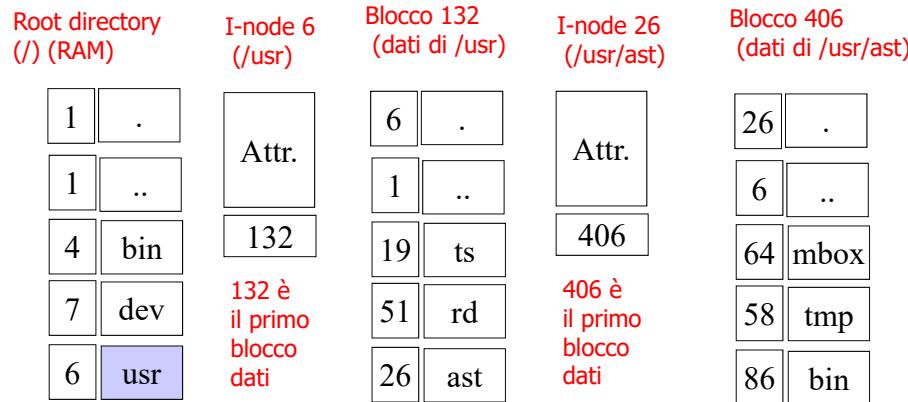
Blocco dati relativo alla directory C2

14

Quanto più ho già di /usr/ast/mbox? 1. Devo vedere se c'è, 2. portare I-node da memoria a disco.  
 Da dove si parte? Cerco "usr" nella directory "root" che tipicamente è in RAM (no accesso al disco).  
 E' inciso l'ultimo byte per uscire. Se lo trovi non in I-node ass. alla directory. Devo recuperare quell'I-node (3° accesso a disco).  
 Tipicamente occupa pochi blocchi. Pendo I-node 6, vedo primo blocco e poi accesso a quell'ID blocco, (2° lettura disco).  
 Da leggo altri sottostrutture "ast" e trovo I-node associato a dire ast. (3° accesso a disco). Vedo primo blocco e accedo a quello che  
 contiene (4° accesso). Poco da memoria (tutto la directory e circa mbox). Se lo trovi, faccio ultimo accesso a disco per I-node trovato.  
 (5° accesso a disco).

## Implementazione del FS di Unix (7)

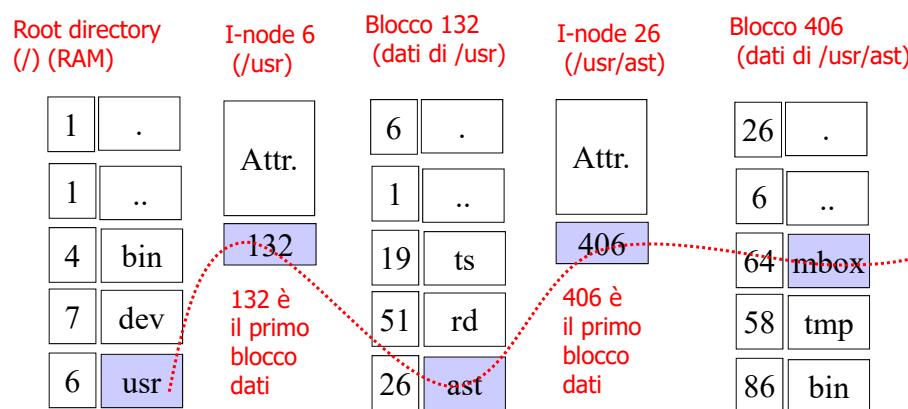
1 accesso per I-node, 1  
 per cercare info su directory.



I passi necessari per leggere /usr/ast/mbox

15

## Implementazione del FS di Unix (8)

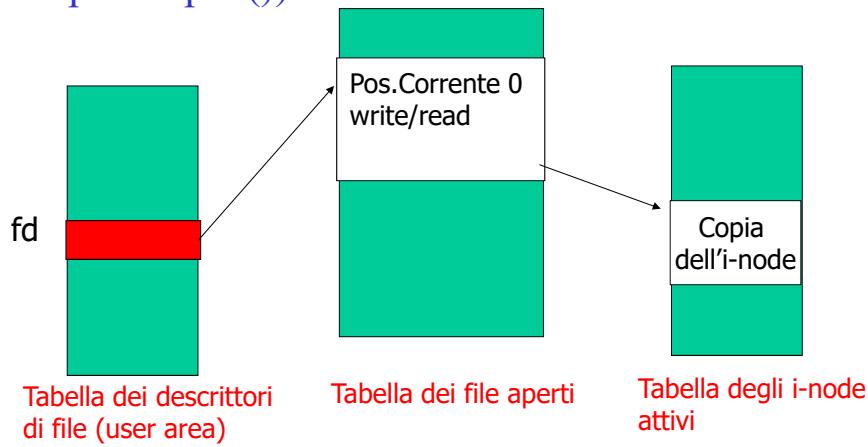


I passi necessari per leggere /usr/ast/mbox

16

## Tabelle relative ai file aperti

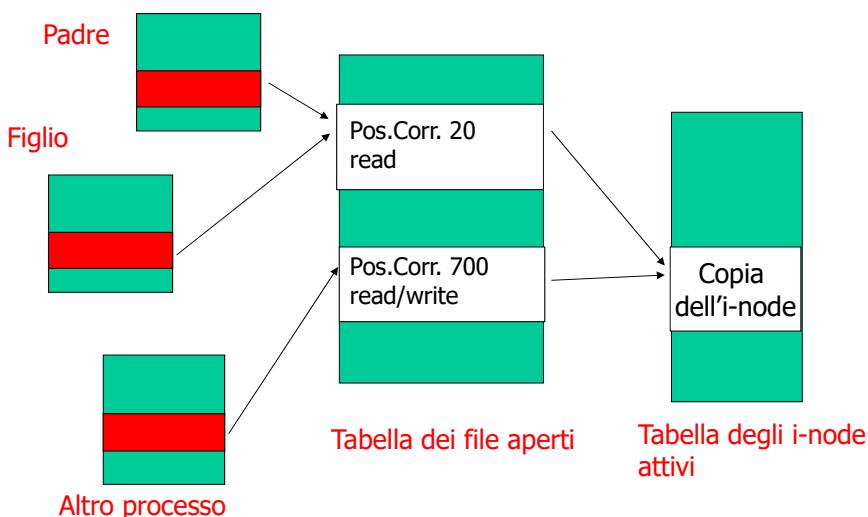
- Rappresentazione di un file aperto (subito dopo la open())



17

## Tabelle relative ai file aperti (2)

- Perché 3 diverse ?



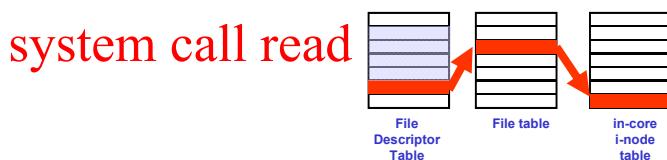
18

## Tabelle relative ai file aperti (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 ...)

19

## La system call read



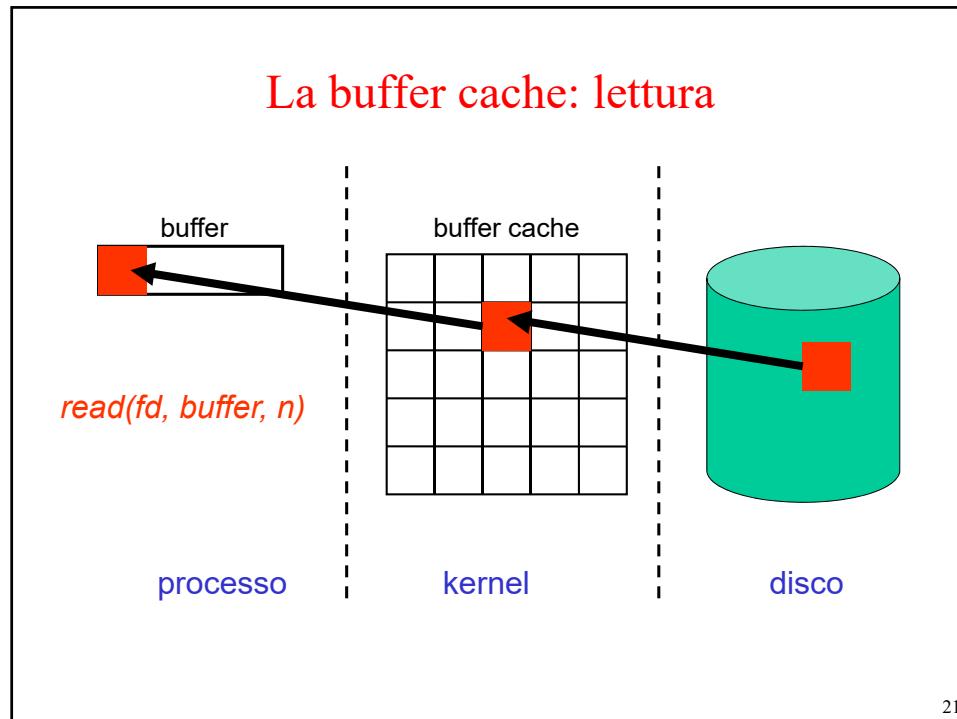
$n = \text{read}(fd, buffer, nbytes)$

- accede alla File Table entry a partire da fd e controlla la modalità di apertura *modo lettura da blocco so dove mi trovo nel file*\*
- accede all'in-core i-node; lock l'in-core i-node
- trasforma l'offset nella File Table entry in un indirizzo fisico (indirizzo blocco + offset all'interno del blocco), attraverso la tabella di indirizzamento nell'in-core i-node
- legge il blocco/i richiesto/i nella buffer cache e copia gli nbytes richiesti in \*buffer
- aggiorna l'i-node; unlock i-node\*
- aggiorna l'offset in File Table entry (*lettura*)
- restituisce al chiamante il numero di byte letti

20

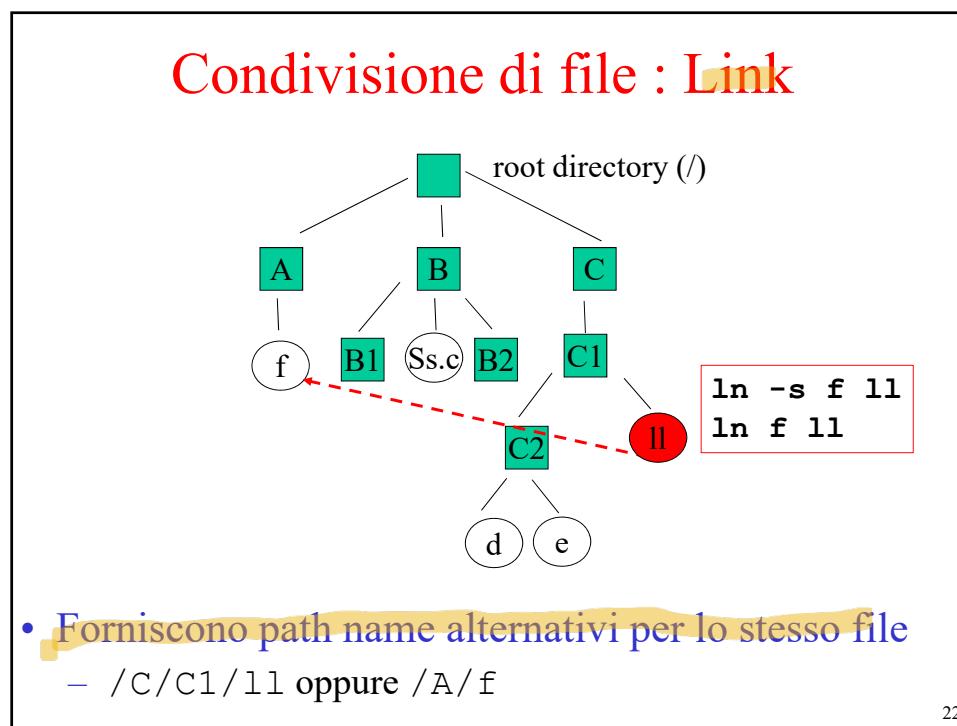
\* Valore di lettura diviso per la dimensione del blocco. nbytes da leggere prende come riferimento la lettura. Dato un nbytes il punto logico in cui mi trovo. Ma se mi suppone in quale blocco sto puntando per quello. Quindi dovrà dirmi dove devo andare. Se fermo nel blocco successivo devo fare 2 letture del blocco successivo. (nbytes sono nell'i-node ma se sono in fondo devo arrivare alle altre porzioni dell'i-node). Firmo nei numeri indiretti da i-node. Devo recuperarlo e partire blocco 10 i-mode in memoria centrale.

Fatto la open non sono richiesti molti accessi al disco.  
Il modo è una struttura di file che può essere aperta da process diversi. Quando MUTUA ESCLUSIONE serve.  
È meccanismo sicurezza.



21

Probabile creare, scrivere.



22

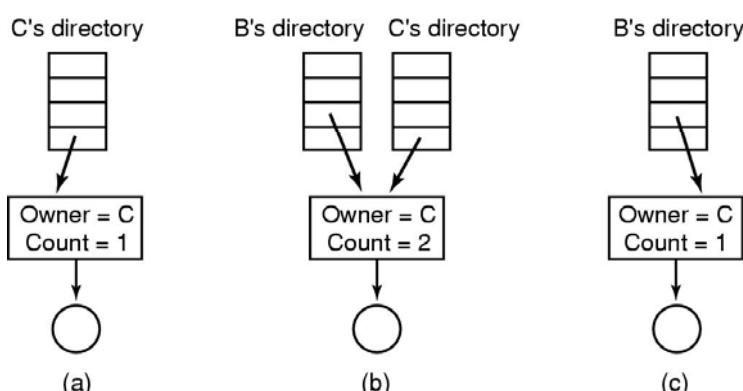
## Link (2)

- Hard link : `ln f 11` *muove entry nella directory che ha file*  
    – le due directory condividono la struttura dati relativa al file (i-node)  
    – paradosso della rimozione da parte dell'owner
- Symbolic Link : `ln -s f 11`  
    – 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)  
*↳ nell'i-node ho solo il percorso del file originale. Poco ma stringa.*

23

Questo link è rimovibile all'insaputa del file.

## Link (3)

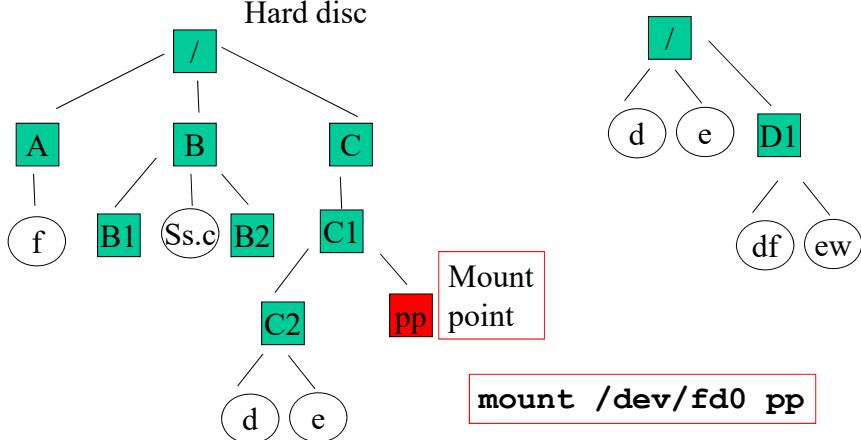


- (a) **situazione precedente al linking (hard)**
- (b) **dopo la creazione del link**
- (c) **dopo che l'owner originale ha rimosso il file**

24

\* In i-node deve segnare che ho 2 ref. allo stesso file. Se cancello uno dei due non serve cancellare file. lo faccio con contatore.

## Mounting

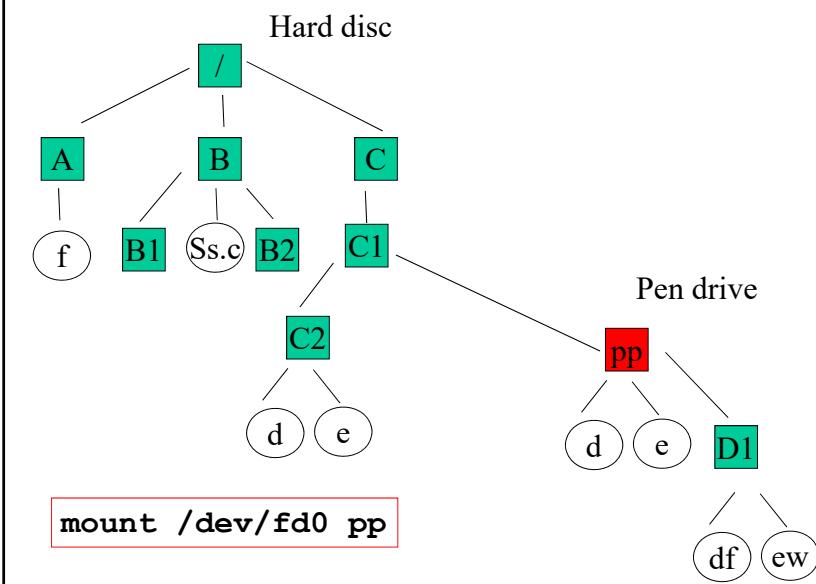


- Permette di unire in un unico albero file system di tipo diverso memorizzati su dispositivi diversi

25

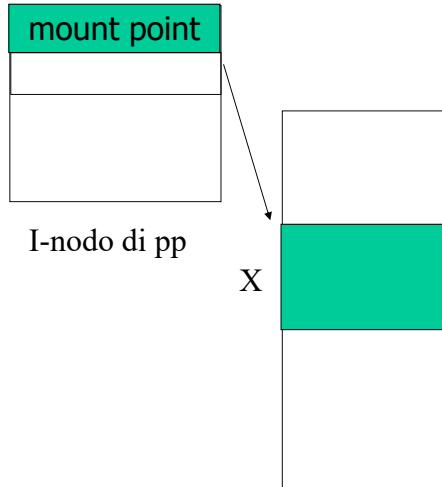
Pendere come procede partizione con albero chs. Soluzione UNIX: prima c'era un punto di mount, ora fatta da automnt. A file system della plancia viene associato un I-NODE che rapp. punti di montaggio che mi dà tutte le info per utilizzarlo (s. info sulla dir root della Riva). Nella pen mi porta la info di accesso.

## Mounting (2)



26

## Mounting (3)



In MT(x) :

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

Mount table

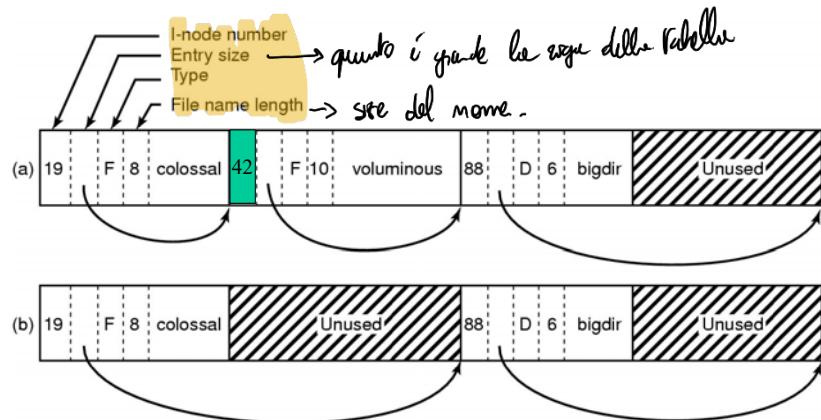
27

## 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

28

## Il Fast File System di Berkley (2)



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

29

La dir contiene sempre stesse info, N. I-NODE, ma le file. Ma sono qualcosa che dice quanto è lungo o lungo per quella directory.

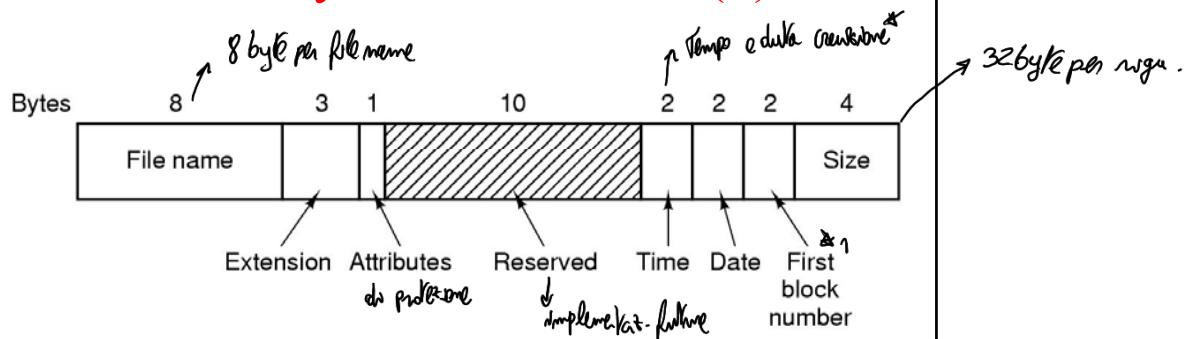
## File system

Casi di studio  
MS-DOS, Windows

30

FAT  $\Rightarrow$  directory più complicate. Altri file nelle directory.

## Il File System MS-DOS (1)



- Rappresentazione di una directory in MS-DOS
- Attributi : read only, file nascosto, file di sistema, etc.
- Usa la FAT

31

\* Ora, minuti e secondi. 24 valori diverse, 5 b.t. Secondi sono 60, ma non arrivano al dell'oglio del secondo. Minuti 60, 6 b.t. 11 b.t.

Per la data: giorni,  $31 = 5$  b.t. Mesi = 12, 4 b.t. Anni = 9 b.t. E ne restano 7 per l'anno. Soluzione: numero da 0 a 127 gli anni e partire da un dell'oglio giorno.

## Il File System MS-DOS (2)

Block size	Presto		
	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Usiamo solo 28 b.t.  
 $2^{28}$  sono 256 blocchi per dir. Si blocca da 512 KB per 1 TB.

Ma fanno per usare di gestire partizioni.

- massima ampiezza delle partizioni per diverse ampiezze dei blocchi
- elementi vuoti = combinazioni non ammesse

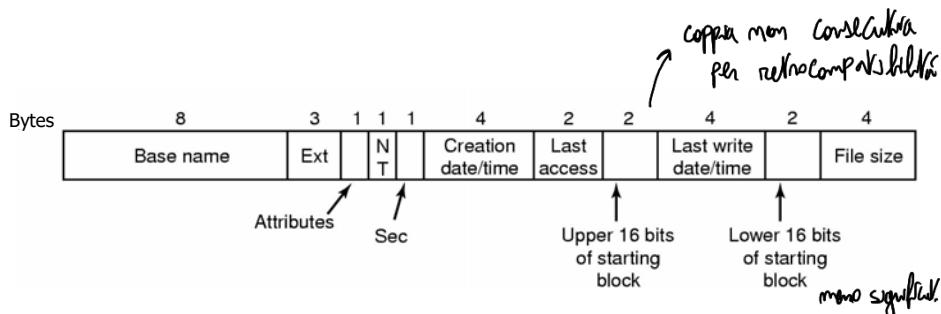
32

Potrei solo aumentare dim. blocco.

Quindi solo 2 byte per 1° blocco posto a destra di 2 GB.

NOTA: fin dall'8 windows ha cercato sempre di mantenere retrocompatibilità (cosa che ha portato costi)

## Il File System di Windows 98



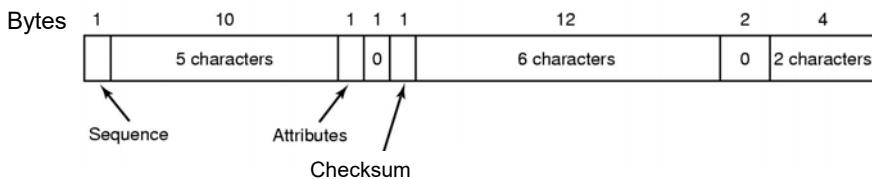
La rappresentazione estesa di una directory MS-DOS utilizzata in Windows 98

- si possono usare ‘nomi lunghi’
- FAT-32 (solo parte della tabella -window- viene tenuta in memoria)

33

Altre migliorie: ma vedevo su dimensione mem. Prob. compatibilità: soluzione? Uso entry aggiuntive per il nome. 32 byte aggiuntive per entry che aggiunge solo il controllo del nome e altre info.

## Il File System di Windows 98 (2)



La rappresentazione di (parte di) un nome di file “lungo” in Windows 98

34

## Il File System di Windows 98 (3)

Copiste qui sotto sono le entry aggiuntive rispetto a quella finale, con 68-64. 68 è il primo byte. Entry fra le entry da 32

68	d	o	g	A	0	C			0	
3	o	v	e	A	0	K	t	h	e	l
2	w	n	f	o	A	0	x	j	u	m
1	T	h	e	q	A	0	i	c	k	b
	T	H	E	Q	U	I	~	1	A	N
Bytes					T	S	Creation time	Last acc	Upp	Last write
										Low
										Size

68 per il nome e  
la S\* che porta info  
più importanti.

Esempio di come viene rappresentato un nome “lungo” in Windows 98

35

Finehha dos non riconosce entry aggiuntive, quindi vede visualizzato il nome continua.

Se cancella file, dos cancella solo entry che riconosce, quindi ultima. Le entry rimangono comunque. Se un altro file occupa quella entry diversa, delle accorgono chi differenza. Lo fa facendo con byte da checksum: la S\* entry non ha niente in che fare con quella precedente.

NOTA: Distinguo entry del nome vs effettuale del campo file entry. Copista che non è entry finale. Guarda n. rappresentato nel primo byte, che dice "per scrivere a quella entry ce ne sono altre X". Gli n. entry successive. Probabilmente per limitare n. canali.

## File system

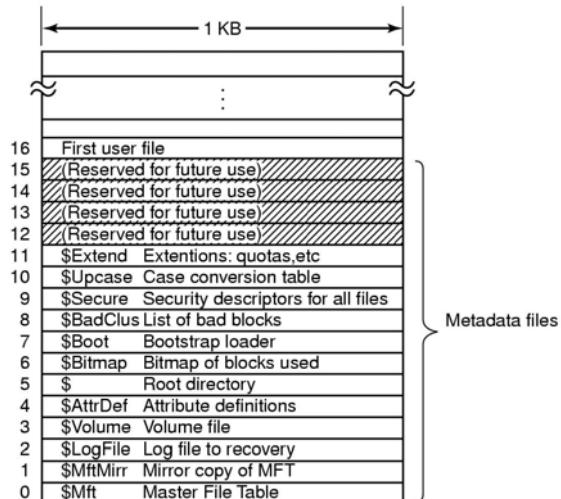
### NTFS

File system completamente nuovo.

Assegna una struttura ad hoc per ogni file.

36

## Struttura del File System NTFS



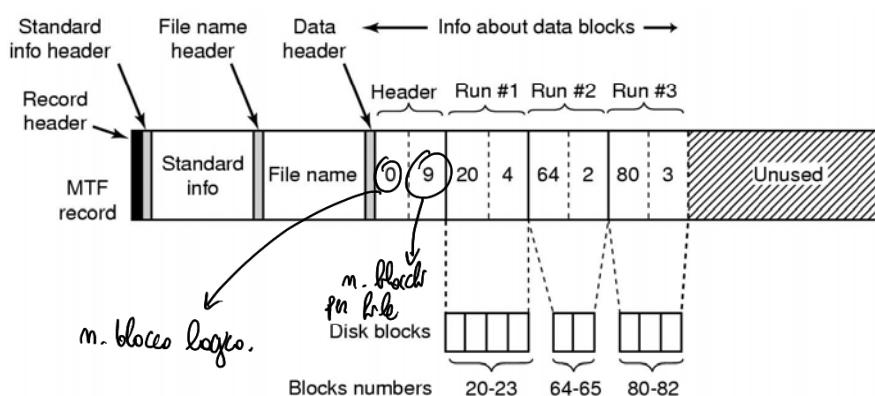
La master file table di NTFS

37

Record salvo da 1 numero associato a file o directory. Ha info per attributi, nome, ecc... per gestire file. Molto simile a index-mode. Dimensione è 1 Kbyte. In più, essi sono memorizzati all'interno della partizione (per gli s-mode). Qui, si memorizza non un file: MASTER FILE TABLE. File contiene record (esclusi i primi 16 che sono speciali) riguardo file e directory.

Poiché nel file? Se succede qualcosa in qualche traccia su cui ha ceso. Quindi non memorizza tutto in blocchi consecutivi. Dove è salvato su disco e lo altre record 0 e 1.

## Record MFT per un file



Un record Master File Table per un file di 3 sequenze (run) e 9 blocchi

38

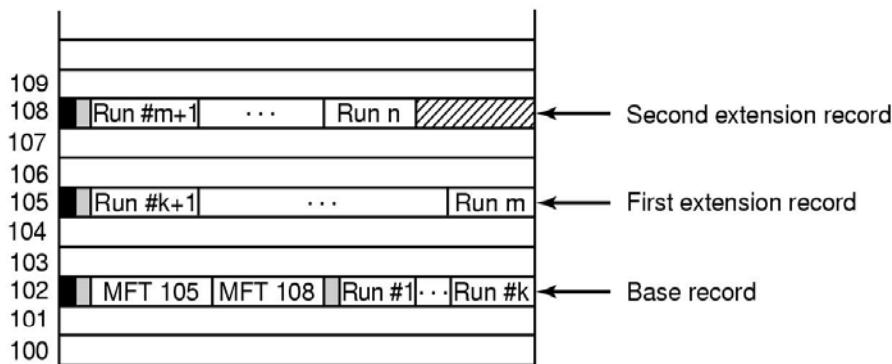
Record per file ha info di s-mode: nome, attributi, sequenza n. blocco. Se file occupa m. conseguenti più meglio, perché sepp. rappresenta affollamento dei num. Il punto X a partire da m. dn num. e così via.

Record 1 e 0 sono il record MFT per la master file table.

Se faccio la opzione, contro un memoria MFT del file.

VANTAGGIO: Semplicemente file  
Nella f. più compatibile dei m. blocco.

## Record MFT per file lunghi



Un file che richiede 3 MFT record per memorizzare i suoi *run*

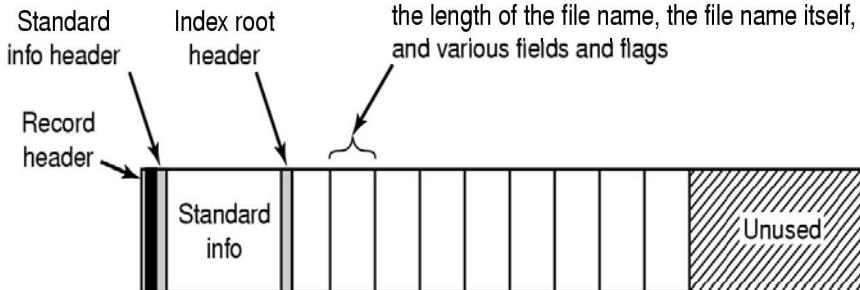
39

M. record associato a file è 102. Nella prima parte del record trovo info su dove prendere gli altri MFT. Nell'header trovo info su quanti blocchi reso a lavorare nella MFT, quindi so subito dall'header se mi devo spostare o no.

## Record MFT per una directory

Un po' più complesse per quelle di unix, ma sempre copia nome - m. record.

A directory entry contains the MFT index for the file, the length of the file name, the file name itself, and various fields and flags



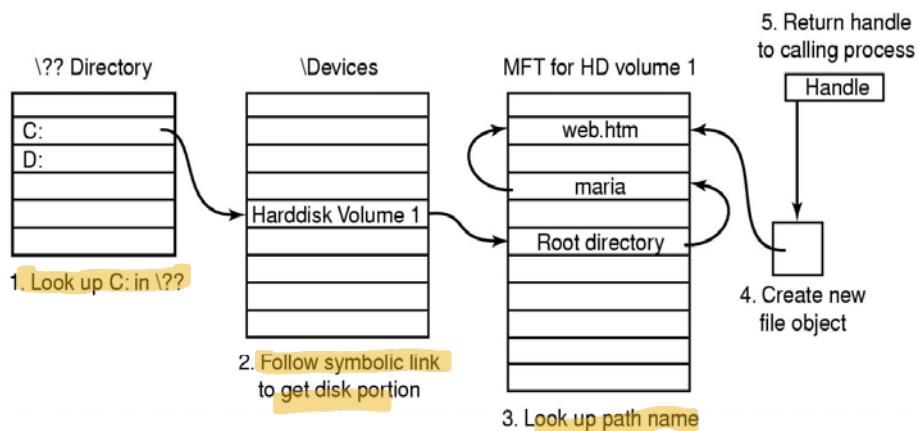
Il record MFT per una piccola directory

40

Nota: MFT record di dove è (se le dir sono piccole) contiene già la directory info.  
Nel record MFT della directory ci sono già le coppie nome - m. del MFT.

VANTAGGIO

## File Name Lookup



I passi necessari a recuperare il file *C:\maria\web.htm*

41

Nella tab. file aperto trova infine un MFT record che fa riferimento al nuovo carrello.