

Modelli e problematiche di *file system* Parte 2 - Indice

1. Realizzazione del *file system*
2. Realizzazione dei *file*
3. Realizzazione delle *directory*
4. Esempi di *file system*
5. Integrità e prestazioni del *file system*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 101

Modelli e problematiche di *file system* Realizzazione del *file system* - 1

- I *file system* (FS) sono memorizzati su disco
 - I dischi possono essere partizionati
 - Ogni partizione può contenere un proprio FS
- Il settore 0 del disco contiene le informazioni di inizializzazione del sistema (*Master Boot Record*)
 - L'inizializzazione è eseguita dal BIOS
 - L'MBR contiene una descrizione delle partizioni, che identifica quella attiva
 - Il primo blocco di informazione di ogni partizione contiene le sue specifiche informazioni di inizializzazione (*boot block*)

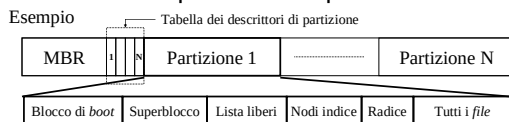
Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 102

Modelli e problematiche di *file system* Realizzazione del *file system* - 2

- I dischi vengono letti e scritti a **blocchi** di ampiezza fissa (*cluster* per Microsoft!)
 - Rischio di frammentazione interna
 - Unità informativa su disco è il **settore**
 - 1 blocco = N settori ($N \geq 1$)
- Struttura di partizione è specifica del FS



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 103

Modelli e problematiche di *file system* Realizzazione dei *file* - 1

- A livello fisico, un *file* è un insieme di blocchi di disco
 - Occorre decidere quali blocchi assegnare a quale *file* e come tenerne traccia
- 3 strategie di allocazione di blocchi a *file*
 - Allocazione contigua
 - Allocazione a lista concatenata (*linked list*)
 - Allocazione a lista indicizzata

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 104

Modelli e problematiche di *file system* Realizzazione dei *file* - 2

- **Allocazione contigua**
 - Memorizzazione dei *file* su blocchi **consecutivi**
 - Ogni *file* è descritto dall'indirizzo del suo primo blocco e dal numero di blocchi utilizzati
 - Consente sia accesso sequenziale che diretto
 - Può essere letto e scritto con un solo accesso al disco
 - Ideale per CD-ROM e DVD
 - Ogni modifica di *file* induce frammentazione esterna
 - Ricompattazione periodica molto costosa
 - L'alternativa richiede l'utilizzo dei gruppi di blocchi liberi
 - Mantenere la lista dei blocchi liberi e la loro dimensione → oneroso
 - Conoscere in anticipo la dimensione massima dei nuovi *file* → rischioso

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 105

Modelli e problematiche di *file system* Realizzazione dei *file* - 3

- **Allocazione a lista concatenata**
 - Un *file* come lista concatenata di blocchi
 - Ogni *file* è descritto dal puntatore al primo blocco
 - Per alcuni S/O, anche dal puntatore all'ultimo blocco del *file*
 - Ciascun blocco di *file* deve contenere il puntatore al blocco successivo (o fine lista)
 - Questo sottrae spazio ai dati
 - L'accesso sequenziale resta semplice, ma può richiedere molte operazioni su disco
 - Accesso diretto molto più complesso ed oneroso
 - Un solo blocco guasto corrompe l'intero *file*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 106

Modelli e problematiche di *file system* Realizzazione dei file - 4

• Allocations a lista indicizzata

- Si pongono i puntatori ai blocchi in strutture apposite
 - Ciascun blocco contiene solo dati
- File descritto dall'insieme dei suoi puntatori
- 2 strategie di organizzazione
 - Forma tabulare (**FAT**, *File Allocation Table*)
 - Forma indicizzata (nodo indice, *i-node*)
- Non causa frammentazione esterna
- Consente accesso sequenziale e diretto
- Non richiede di conoscere preventivamente la dimensione massima di ogni nuovo file

Il file system

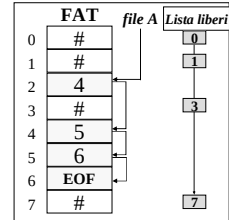
Architettura degli elaboratori 2 - T. Vardanega

Pagina 107

Modelli e problematiche di *file system* Allocazione a lista indicizzata - 1

• File Allocation Table (MS-DOS → Windows)

- Tabella ordinata di puntatori, uno per *ogni* blocco del disco
 - Cresce con il crescere della capienza del disco
- La parte di FAT relativa ai file in uso *deve* essere in RAM
- Consente accesso diretto



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 108

Modelli e problematiche di *file system* Allocazione a lista indicizzata - 2

• Nodi indice (UNIX → GNU/Linux) 1/2

- Una struttura indice (*i-node*) per ogni file, con gli attributi del file ed i puntatori ai suoi blocchi
 - L'*i-node* è contenuto in un blocco dedicato
- In RAM una tabella di *i-node* per i soli file in uso
 - La dimensione massima di tabella dipende dal massimo numeri di file apribili simultaneamente, non più dalla capacità del disco!
- Un *i-node* contiene un numero limitato di puntatori a blocchi
 - Quale soluzione per file composti da un numero maggiore di blocchi?

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 109

Modelli e problematiche di *file system* Allocazione a lista indicizzata - 3

• Nodi indice (UNIX → GNU/Linux) 2/2

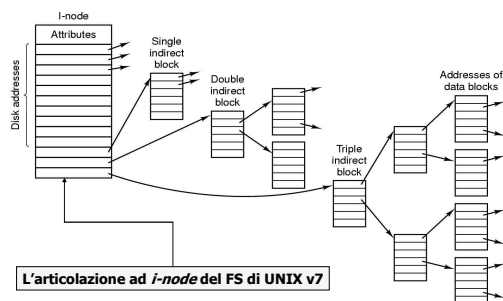
- File di piccola dimensione
 - Gli indirizzi dei blocchi dei dati sono ampiamente contenuti in un singolo *i-node* (frammentazione interna)
- File di media dimensione
 - Un campo dell'*i-node* punta ad un nuovo blocco *i-node*
- File di grandi dimensioni
 - Un campo dell'*i-node* principale punta ad un livello di blocchi *i-node* intermedi, che a loro volta puntano ai blocchi dei dati
- Per file di dimensioni ancora maggior basta aggiungere un ulteriore livello di indizione

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 110

Modelli e problematiche di *file system* Allocazione a lista indicizzata - 4



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 111

Modelli e problematiche di *file system* realizzazione dei file - 5

• Gestione dei file condivisi

- Come preservarne la consistenza senza costi eccessivi
 - Non porre i blocchi dei dati nella *directory* del file
 - Per ogni file condiviso porre nella *directory* remota un *symbolic link* verso il file originale
 - Esistono così 1 solo descrittore ed il file originale
 - L'accesso condiviso avviene tramite cammino sul FS
 - Altrimenti si può porre nella *directory* remota il puntatore diretto (*hard link*) al descrittore (*i-node*) del file originale
 - Più possessori di descrittori dello stesso file condiviso
 - Un solo proprietario effettivo del file condiviso
 - Il file condiviso non può più essere distrutto fino quando esistono suoi descrittori remoti, anche se il suo proprietario avesse inteso cancellarlo

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 112

Modelli e problematiche di *file system* *Realizzazione dei file - 6*

• Gestione dei blocchi liberi

- Vettore di *bit* (*bitmap*), dove ogni *bit* indica lo stato del corrispondente blocco
 - 0 = libero
 - 1 = occupato
- Lista concatenata di blocchi, sfruttando i campi puntatore al successivo (stile FAT)

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 113

Modelli e problematiche di *file system* *Realizzazione delle directory - 1*

- La *directory* fornisce informazioni su nome, collocazione ed attributi dei *file* in catalogo
 - *File* e *directory* risiedono in aree logiche distinte
- Come minimizzare la complessità della struttura interna di *directory*?
 - [Nome + attributi] o [Nome + puntatore a nodo indice con attributi] in struttura di lunghezza fissa
 - Frammentazione interna trascurabile per nomi di *file* fino ad 8 caratteri + 3 di estensione
 - Problema più serio con nomi lunghi

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 114

Modelli e problematiche di *file system* *Realizzazione delle directory - 2*

- La ricerca di un *file* ne correla il nome (stringa ASCII) alle informazioni necessarie per l'accesso
 - Nome e *directory* di appartenenza del *file* sono determinati dal percorso indicato dalla richiesta
- La ricerca lineare in *directory* è di facile realizzazione, ma di esecuzione onerosa
- La ricerca mediante tabelle *hash* è più complessa ma più veloce
 - $F(\text{nome}) = \text{posizione in tabella} \rightarrow \text{puntatore al file}$
- Si può anche creare in RAM una *cache* di supporto alla ricerca

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 115

Modelli e problematiche di *file system* *Esempi storici di file system - 1*

- CP/M (1973-1981)
- MS-DOS & Windows 95 (1981 → 1997)
- Windows 98 (1998-1999)
- UNIX v7 (1979)

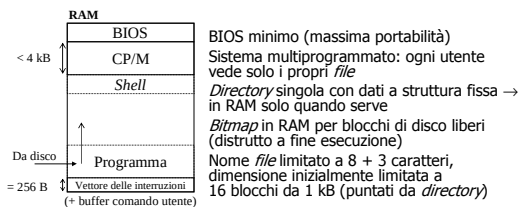
Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 116

Modelli e problematiche di *file system* *Esempi storici di file system - 2*

• CP/M (Control Program for Microcomputers)



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 117

Modelli e problematiche di *file system* *Esempi storici di file system - 3*

• MS-DOS

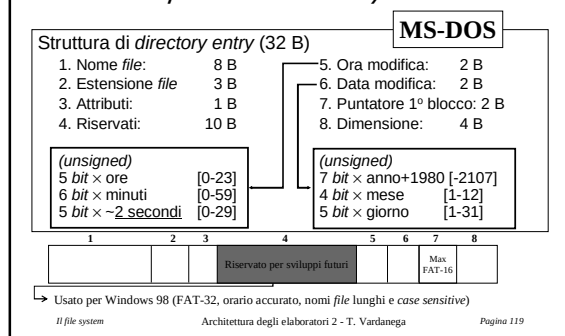
- Non multiprogrammato: ogni utente vede tutto il FS
- FS gerarchico senza limite di profondità, senza condivisione
 - Fino a 4 partizioni per disco (C: D: E: F:)
- Directory a lunghezza variabile con *entry* di 32 B
 - Nomi di *file* normalizzati a 8+3 caratteri (maiuscoli)
- Allocazione *file* a lista (FAT)
 - **FAT-X** per X = numero di *bit* per indirizzo di blocco ($12 \leq X < 32$)
 - Blocchi di dimensione multipla di 512 B (1 settore / *cluster*)
 - **FAT-16** : *File* e partizione limitati a 2 GB
 - $2^{16} = 64k$ (puntatori a) blocchi di 32 kB ciascuno = 2 GB
 - **FAT-32** : blocchi da 4 - 32 kB ed indirizzi da 28 *bit* (!)
 - 2 TB è il limite intrinseco di capacità per partizione
 - 2^{32} settori (*cluster*) da 512 B → 2048 GB

Il file system

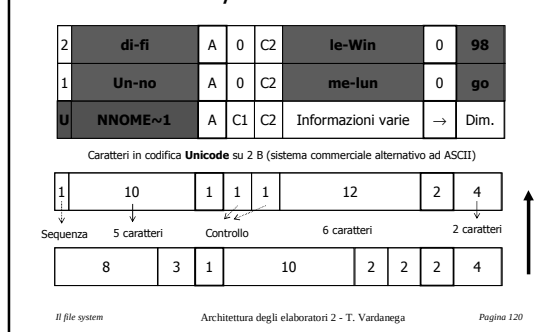
Architettura degli elaboratori 2 - T. Vardanega

Pagina 118

Modelli e problematiche di *file system* Esempi storici di *file system* - 4



Modelli e problematiche di *file system* Esempio: Windows 98

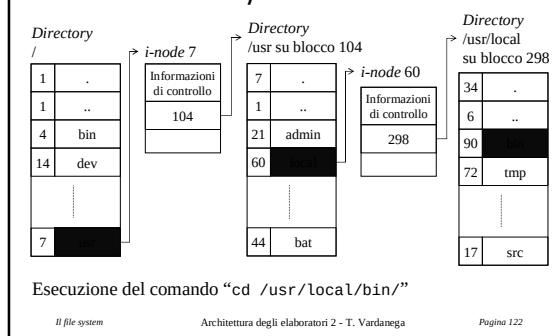


Modelli e problematiche di *file system* Esempi storici di *file system* - 5

- **UNIX v7** (Ken Thompson & Dennis Ritchie)
 - Struttura ad albero con radice e condivisione di *file* (grafo aciclico)
 - Nomi di *file* fino a 14 caratteri ASCII (escluso /)
 - *Directory* contiene nome *file* e puntatore al suo *i-node*, su 2 B
 - Max 64 k *file* per FS (2^{16} *i-node* distinti)
 - L'*i-node* contiene gli attributi del *file*
 - Incluso il contatore di *directory* che puntano al *file* (via *link*)
 - Se contatore = 0, il nodo ed i blocchi del *file* diventano liberi

Il file system Architettura degli elaboratori 2 - T. Vardanega Pagina 121

Modelli e problematiche di *file system* Esempio: UNIX v7



Modelli e problematiche di *file system* File system su CD-ROM

- **ISO 9660**
 - Supporta fino a $2^{16}-1$ dischi partizionabili
 - Dimensione di blocco 2-8 kB
 - *Directory* a struttura variabile internamente ordinate alfabeticamente
 - FS limitato ad 8 livelli di annidamento
- **Rock Ridge**
 - Estensione definita dal mondo UNIX per rappresentare il proprio FS
- **Joliet**
 - Estensione definita da Microsoft per lo stesso motivo

Il file system Architettura degli elaboratori 2 - T. Vardanega Pagina 123

Modelli e problematiche di *file system* Integrità - 1

- **Gestione dei blocchi danneggiati**
 - Via *hardware*, creando in un settore del disco un elenco di blocchi danneggiati ed i loro sostituti
 - Via *software*, ricorrendo ad un falso *file* che utilizza tutti i blocchi danneggiati
- **Salvataggio del FS**
 - Su nastro, tempi lunghi, anche per incrementi
 - Su disco, con partizione di *back-up* o architettura RAID (*Redundant Array of Inexpensive Disks*)

Il file system Architettura degli elaboratori 2 - T. Vardanega Pagina 124

Modelli e problematiche di *file system*

Integrità - 2

• **Consistenza del FS**

- Un *file* viene aperto, modificato e poi salvato
Se il sistema cade tra la modifica ed il salvataggio, il *file* risulta essere inconsistente
- *Consistenza dei blocchi* (come per i *file*)
2 liste di blocchi con un contatore per ogni blocco:
lista dei blocchi in uso dei *file* e lista dei blocchi liberi
 - *Consistenza*: ciascun blocco appartiene ad una ed una sola lista
 - *Perdita*: il blocco non appartiene ad alcuna lista
 - *Duplicazione*: il contatore del blocco è >1 in una delle due liste

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 125

Modelli e problematiche di *file system*

Prestazioni

- Per ridurre la frequenza di accesso ai dischi, una porzione di memoria principale viene usata come *cache* di (alcune migliaia di) blocchi
 - Accesso ai blocchi mediante ricerca *hash*
 - Gestione richiede specifica politica di rimpiazzo blocchi
- Come assicurare la consistenza dei dati su disco
 - MS-DOS : blocchi modificati copiati immediatamente su disco (*write through*)
 - Alto costo ma consistenza sicura (specie con dischi rimovibili)
 - UNIX → GNU/Linux : un processo periodico opera l'aggiornamento (*sync*) dei blocchi su disco
 - Basso costo e basso rischio con dischi fissi affidabili

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 126