07 - File System

Gestione del file system

Parte di SO che fornisce i meccanismi di accesso e memorizzazione delle informazioni (programmi e dati) allocate in mmorie di massa.

Realizza i concetti astratti

- di **file**: unita' logica di memorizzazione
- di **direttorio**: insieme di file (e direttori)
- di **partizione**: insieme di file associato ad un particolare dispositivo fisico (o porzione di esso)

--> Le caratteristiche di file, direttorio e partizione sono del tutto indipendenti da natura e tipo d dispositivo utilizzato

File

E' un insieme di informazioni:

- programmi
- dati (in rappresentazione binaria)
- dati (in rappresentazione testuale)
- ...

rappresentati come insieme di record logici

Ogni file e' individuato da (almeno) un **nome** simbolico mediante il quale puo' essere riferito (ad esempio, nell'invocazione di comandi o system call). Ogni file e' caratterizzato da un insieme di **atttributi**

Attributi del file

A seconda del SO, i file possono avere attributi diversi.

Solitamente

- **tipo**: stabilisce l'appartenenza a una classe (eseguibili, testo, musica, non modificabili, ...)
- **indirizzo**: puntatore/i a memoria secondaria
- **dimensione** numero di byte contenuti nel file
- **data e ora** (di creazione e/o modifica)

In SO multiutente anche

- **utente proprietario**
- **protezione**: diritti di accesso al file per gli utenti del sistema

Descrittori del file: e' la struttura dati che contiene gli attributi di un file.

Ogni descrittore di file deve essere memorizzato in modo persistente: SO mantiene l'insieme dei descrittori di tutti i file presenti nel file system in apposite strutture in memoria secondaria (ad es. UNIX: i-list)

Operazioni sui file

Compito del SO e' consentire l'accesso on-line ai file: ogni volta che un processo modifica un file, tale cambiamento e' immediatamente visibile per tutti gli altri processi.

Tipiche operazioni

- **Creazione**: allocazione di un file in memoria secondaria e inizializzazione dei suoi attributi
- **Lettura** di record logici dal file
- **Scrittura**: inserimento di nuovi record logici all'interno del file
- **Cancellazione**: eliminazione del file dal file system

Ogni operazione richiederebbe la localizzazione di informazioni su disco, come:

- indirizzi dei record logici a cui accedere
- altri attributi del file
- record logici

Per migliorare l'efficienza:

- SO mantiene in memoria una struttura che registra i file attualmente in uso (file aperti) - tabella dei file aperti per ogni file aperto {puntatore al file, posizione su disco, ...}
- Spesso viene fatto il **memory mapping** dei file aperti:
  - i file aperti (o porzioni di essi) vengono temporaneamente copiati in memoria centrale --> accessi piu' veloci

Operazioni necessarie

- **Apertura**: introduzione di un nuovo elemento nella tabella dei file aperti e eventuale memory mapping del file
- **Chiusura**: salvataggio del file in memoria secondaria ed eliminazione dell'elemento corrispondente dalla tabella dei file aperti

Struttura interna dei file

Ogni dispositivo di memorizzazione secondaria viene partizionato in blocchi (o record fisici)

- **Blocco**: unita' di trasferimento fisico nelle operazioni di I/O da/verso il dispositivo. Sempre di dimensione fissa.

L'utente vede il file come un insieme di record logici

- **Record Logico**: unita' di trasferimento logico nelle operazioni di accesso al file (es. lettura, scrittura di blocchi). Di dimensione variabile.

Blocchi & record logici

Uno dei compiti di SO (parte di gestione del file system) e' stabilire una corrispondenza tra record logici e blocchi

Usualmente:

- Dimensione (blocco) >> Dimensione (record logico)
  impaccamento di record logici all'interno di blocchi

Metodi di accesso

L'accesso a file puo' avvenire secondo varie modalita':

- accesso sequenziale
- accesso diretto
- accesso a indice

Il metodo di accesso e' **indipendente**:

- dal tipo di dispositivo utilizzato
- dalla tecnica di allocazione dei blocchi in memoria secondaria

Accesso sequenziale

Il file e' una sequenza [R_1, R_2, ..., R_N] di record logici:

- per accedere ad un particolare record logico R_i, e' necessario accedere prima agli (i-1) record che lo precedono.
- le operazioni di accesso sono del tipo:
  - **readnext**: lettura del prossimo record logico nella sequenza
  - **writenext**: scrittura del prossimo record logico
- ogni operazione di accesso (lettura/scrittura) posiziona il **puntatore al file** sull'elemento successivo a quello appena letto/scritto

UNIX prevede questo tipo di accesso

Accesso diretto

Il file e' un insieme $\{R_1, R_2, ..., R_N\}$ di record logici numerati con associata la nozione di posizione:

- si puo' accedere direttamente a un particolare record logico specificandone il numero
- operazioni di accesso sono del tipo
  - **read i**: lettura del record logico i
  - **write i**: scrittura del record logico i
- utile quando si vuole accedere a grossi file per estrarre/aggiornare poche informazioni (ad esempio nell'accesso a database)

Accesso a indice

Ad ogni file viene associata una **struttura dati** contenente un indice delle informazioni contenute per accedere a un record logico, si esegue una ricerca nell'indice (utilizzando una chiave)

Directory (o direttorio)

Strumento per organizzare il file all'interno del file system:

- una directory puo' contenere piu' file
- e' realizzata mediante una struttura dati che associa al nome di ogni file come e/o dove e' allocato in memoria di massa

Operazioni sui direttori:

- **Creazione/cancellazione** di directory
- **Aggiunta/cancellazione** di file
- **Listing**: elenco di tutti i file contenuti nella directory
- **Attraversamento** della directory
- **Ricerca** di file in directory

Tipi di directory

La struttura logica delle directory puo' variare a seconda del SO

Schemi piu' comuni:

- **a un livello**
- **a due livelli**
- **ad albero**
- **a grafo aciclico**

Struttura a un livello

Una sola directory per ogni file system

Problemi:

- unicita' nei nomi
- multiutenza: come separare i file dei diversi utenti?

Struttura a due livelli

- primo livello (directory principale): contiene una directory per ogni utente del sistema
- secondo livello: directory utenti (a un livello)

Struttura ad albero

Organizzazione gerarchica a N livelli. Ogni direttorio puo' contenere file e altri direttori

Struttura a grafo aciclico (es. UNIX)

Estende la struttura ad albero con la possibilita' di inserire link differenti allo stesso file

Directory e partizioni

Una singola unita' disco puo' contenere piu' partizioni.
Una singola partizione puo' utilizzare piu' di una unita' disco

File System Mounting

Molti SO richiedono il mounting esplicito all'interno del file system prima di poter usare una (nuova) unita' disco

File system e protezione

Il proprietario/creatore di un file dovrebbe avere la possibilita' di controllare:

- quali azioni sono consentite sul file
- da parte di chi

Possibili tipologie di accesso:

- read
- execute
- delete
- write
- append
- list

Liste di accesso e gruppi (es. UNIX)

Modalita' di accesso: read, write, execute

- 3 classi di utenti
  1. owner access 7 --> RWX
  2. group access 6 --> RW
  3. public access 1 --> X
- Amministratore puo' creare gruppi (con nomi unici) e inserire/eliminare utenti in/da quel gruppo
- Dato un file o una directory, si devono definire le regole di accesso desiderate

Realizzazione del file system

SO si occupa anche della realizzazione del file system sui dispositivi di memorizzazione di massa:

- **realizzazione dei descrittori** e loro organizzazione
- **allocazione dei blocchi fisici**
- **gestione dello spazio libero**

Metodi di allocazione

Ogni blocco contiene un insieme di record logici contigui

Quali sono le tecniche piu' comuni per l'allocazione dei blocchi su disco?

- allocazione **contigua**
- allocazione **a lista**
- allocazione **a indice**

Allocazione contigua

Ogni file e' mappato su un insieme di blocchi fisicamente contigui

Vantaggi

- **costo** della ricerca di un blocco
- possibilita' di **accesso sequenziale e diretto**

Svantaggi

- individuazione dello **spazio libero** per l'allocazione di un nuovo file
- **frammentazione esterna**: man mano che si riempie il disco, rimangono zone contigue sempre piu' piccole, a volte inutilizzabili
  - necessita' di azioni di compattazione

- **aumento dinamico delle dimensioni** di file

Allocazione a lista (concatenata)

I blocchi sui quali viene mappato ogni file sono **organizzati in una lista concatenata**

Vantaggi

- non c'e' **frammentazione esterna**
- minor costo di allocazione

Svantaggi

- possibilita' di errore se link danneggiato
- **maggior occupazione** (spazio occupato dai puntatori)
- difficolta' di realizzazione dell'accesso diretto
- **costo della ricerca** di un bloocco

Allocazione a indice

Allocazione a lista: i puntatori ai blocchi sono distribuiti sul disco

- elevato tempo medio di accesso a un blocco
- complessita' della realizzazione del metodo di accesso diretto

Allocazione a indice: tutti i puntatori ai blocchi utilizzati per l'allocazione di un determinato file sono concentrati in un unico blocco per quel file (blocco indice)

A ogni file e' associato un blocco (indice) in cui sono contenuti tutti gli indirizzi dei blocchi su cui e' allocato il file

Vantaggi

- stessi dell'allocazione a lista, piu'
  - possibilita' di accesso diretto
  - maggiore velocita' di accesso (rispetto a liste)

Svantaggi

- possibile scarso utilizzo dei blocchi indice

Tabella di allocazione dei file (FAT)

Alcuni SO (ad es. DOS e OS/2) realizzano l'allocazione a lista in modo piu' efficiente e robusto:

- per ogni partizione, viene mantenuta una tabella (FAT, File Allocation Table) in cui ogni elemento rappresenta un blocco fisico
- concatenamento di blocchi sui quali e' allocato un file e' rappresentato nella FAT

Metodi di allocazione

Riassumendo, gli aspetti caratterizzanti sono:

- grado di utilizzo della memoria
- tempo di accesso medio al blocco
- realizzazione dei metodi di accesso

Esistono SO che adottano piu' di un metodo di allocazione; spesso:

- file **piccoli** --> allocazione congiunta
- file **grandi** --> allocazione a indice