Sistemi Operativi
AA 2018/19
Esercitazione

10 Dicembre 2018

Esercizio 1

Si consideri un file consistente in 100 blocchi e che il suo *File Control Block* (FCB) sia gia' in memoria. Siano dati due file-systems gestiti rispettivamente tramite allocazione a lista concatenata (*Linked List Allocation*) e allocazione indicizzata (*Indexed Allocation*). Si assuma che nel caso indicizzato il FCB sia in grado di contenere i primi 200 blocchi del file.

Domanda Si calcolino le operazioni di I/O su disco necessarie per eseguire le seguenti azioni in entrambi i file-systems:

1. Aggiunta di un blocco all'inizio del file
2. Aggiunta di un blocco a meta' del file
3. Aggiunta di un blocco alla fine del file

Soluzione Nel caso di file-system con *Indexed Allocation*, ogni file conterra un index block, ovvero un blocco contenente i puntatori a tutti gli altri blocchi componenti il file. Cio' implica che per raggiungere un dato blocco bastera' effettuare una semplice indicizzazione. Nell'implementazione con *Linked List Allocation* invece, ogni blocco contiene il riferimento al precedente ed al successivo. In questo caso, per effettuare operazioni che non siano all'inizio del file bisognera' scorrere la lista fino al blocco desiderato ed in seguito eseguire l'operazione necessaria.

Date queste premesse, i risultati sono:

1. Linked Allocation: 1 `IO-ops`; Indexed Allocation: 1 `IO-ops`
2. Linked Allocation: 52 `IO-ops`; Indexed Allocation: 1 `IO-ops`
3. Linked Allocation: 3 `IO-ops`; Indexed Allocation: 1 `IO-ops`

Esercizio 2

Che relazione ce tra un *File Descriptor* ed una entry nella tabella globale dei file aperti del file system?

Soluzione Un File Descriptor consiste in un file handler che viene restituito ad un processo in seguito ad una chiamata alla syscall `open()`. In seguito a tale chiamata, il sistema scandisce il FS in cerca del file e, una volta trovato, il FCB e' copiato nella tabella globale dei file aperti. Per ogni singolo file aperto, anche se da piu' processi esiste una sola entry nella tabella globale dei file aperti.

Viene, quindi, creata una entry all'interno della tabella dei file aperti detenuta dal processo, la quale puntera' alla relativa entry nella tabella globale, insieme ad altre informazione - e.g. permessi, locazione del cursore all'interno del file, ecc. La syscall `open()` restituisce per l'appunto l'entry all'interno della tabella del processo - il File Descriptor. Piu' `open()` su uno stesso file da parte di uno stesso processo generano descrittori diversi.

Esercizio 3

Cos'e' un *File Control Block* (FCB)? Cosa contiene al suo interno?

Soluzione Il FCB e' una struttura dati che contiene tutte le informazioni relative al file a cui e' associato. Esempi di informazioni possono essere: permessi, dimensione, data di creazione, numero di inode (se esiste), ecc. . Inoltre il FCB contiene informazione su la locazione sul disco dei dati del file - ad esempio in un FS con allocazione concatenata il puntatore al primo blocco del file. In Figura 1 e' riportata una illustrazione di tale struttura.

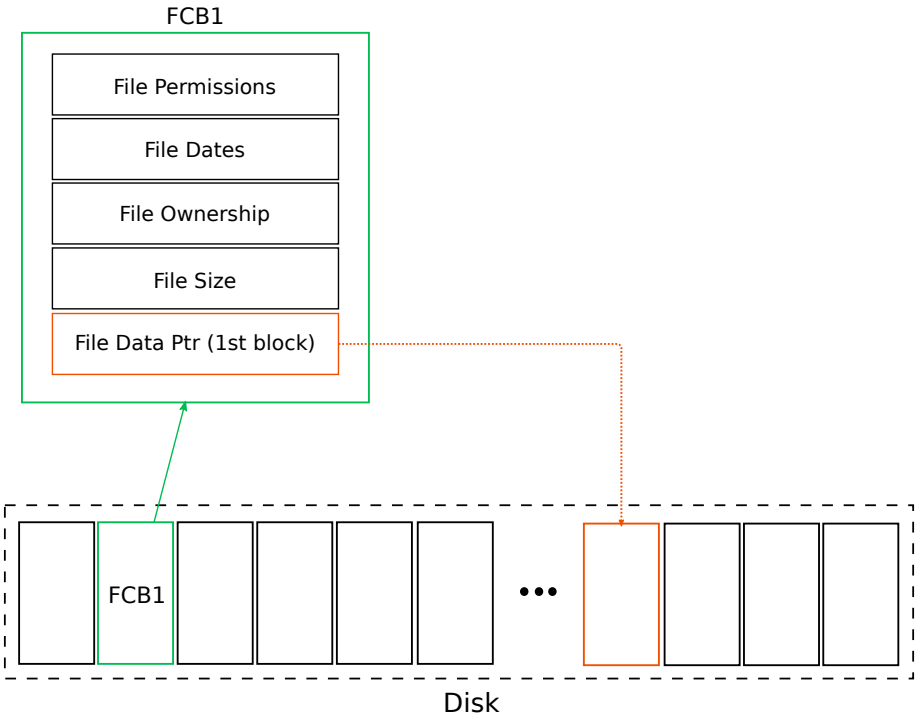

Figure 1: Esempio di FCB.La struttura contiene tutti gli attributi del file compresa la locazione dei dati - qui rappresentato dal puntatore al primo blocco della lista contente i dati, supponendo un FS con linked allocation.

Esercizio 4

Si consideri l'implementazione di un file system con allocazione concatenata (Linked Allocation) ed un file system che invece utilizzi una allocazione ad indice (Indexed Allocation).

Domanda Illustrare brevemente i vantaggi dell'uno e dell'altro nell'eseguire le seguenti operazioni:

1. accesso sequenziale
2. accesso indicizzato
3. operazioni su file di testo

Soluzione

1. **Accesso Sequenziale**: in questo caso, il file system che usa la *lista concatenata* sarà favorito, garantendo una maggiore velocità dell'operazione. Ciò poiché non bisogna effettuare alcuna ricerca per trovare il blocco successivo, poiché esso sarà semplicemente il blocco `next` nella lista.
2. **Accesso Indicizzato**: questa operazione - contrariamente alla precedente - risulta essere molto onerosa per il file system che usa la *linked list*. Infatti, per ogni accesso, bisognerà scorrere tutta la lista finchè non viene trovato il blocco desiderato. La ricerca tramite `inode` risulterà molto più efficiente.
3. **Accesso su file di testo**: per la natura del tipo di file (testo), la *linked list* risulterà più efficiente ancora una volta. Questo poiché i file di testo sono memorizzati in maniera sequenziale sul disco, riportandoci al caso 1.