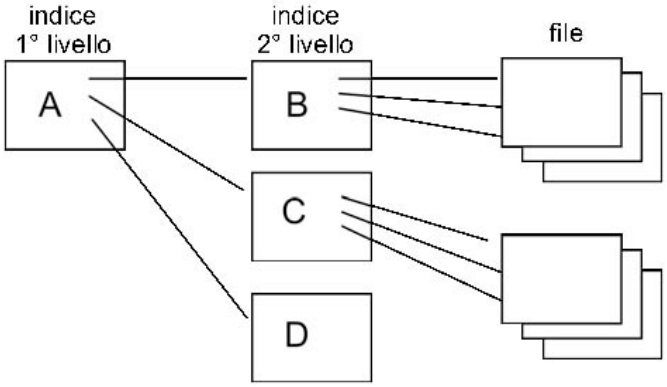
Università Ca' Foscari Venezia - Corso di Laurea in Informatica
Sistemi Operativi – prof. Augusto Celentano

Il file system e la memoria secondaria – Esercizi risolti

1) Si faccia riferimento ad un file system indicizzato con indici a due livelli, dimensione del blocco logico pari a 512 byte e indirizzi di 4 byte. Come si alloca un file di 1 Mb? Quanti blocchi servono? Come si accede al suo 400° blocco? Come si accede al byte 236.448? Qual è la dimensione massima di un file? Quanti blocchi occupa complessivamente?

Soluzione

L'allocazione indicizzata a due livelli segue lo schema illustrato dalla figura seguente:



Se il blocco logico è di 512 byte e gli indirizzi di 4 byte, si hanno 512 / 4 = 128 indirizzi per blocco. Un file di 1 Mbyte occupa 1M / 512 = 2K = 2048 blocchi (1M = 1024*1024), che richiedono 2048 indici corrispondenti a 2048/128 = 16 blocchi di secondo livello, a loro volta indicizzati da un blocco di primo livello.
Complessivamente, quindi il file occupa 2048 + 16 + 1 = 2065 blocchi.

Il blocco n. 400 è indicizzato dal blocco indice di 2° livello n. 400 / 128 = 3, l'indirizzo è il 400 mod 128 = 16-esimo del blocco. In generale, l'n-esimo indirizzo occupa i byte da (n -1) x 4 a (n -1) x 4 + 3. Il blocco indice di secondo livello a sua volta è indicizzato dal 4° indirizzo (indirizzo n. 3) dell'indice di 1° livello, che si trova nei byte 12-15 del blocco.

L'accesso al byte 236.448 si risolve così: il byte occupa il blocco 236.448 / 512 = 461 (contando da 0, quindi è il 462-esimo), e si trova all'offset 236.448 mod 512 = 416. Trovato il blocco si procede come sopra.

La dimensione massima di un file è data dal numero massimo di indirizzi utilizzabili per indicizzarlo.Nell'indice di 1° livello ci sono al massimo 128 puntatori, quindi ci sono al massimo 128 blocchi indice di 2° livello, ciascuno dei quali contiene 128 puntatori ai blocchi del file, per un totale di 128 x 128 = 16384 puntatori, e quindi blocchi del file. Il file può avere una dimensione massima di 16384 x 512 = 8 Mbyte, e tale dimensione occupa complessivamente 16384 + 128 + 1 = 16513 blocchi.

2) Con riferimento al file system di Unix (indicizzato multilivello) con dimensione del blocco di allocazione di 512 byte e puntatori di 4 byte, si consideri un file costituito da 200 blocchi. Indicare (motivando la risposta) il numero di accessi a disco necessari per:

- leggere con accesso diretto il blocco n. 10 (contando da 1);
- leggere con accesso diretto il blocco n. 190 (contando da 1);
- leggere in modo sequenziale i primi 50 blocchi del file.

Soluzione

In ogni blocco indice co sono 512 byte / 4 byte = 128 puntatori. Il file richiede un numero di blocchi indice che risulta da queste considerazioni:

- 12 indici per i primi 12 blocchi del file sono nell'i-node;
- 128 indici per i blocchi da 13 a 140 (contando da 1) sono nel blocco indice puntato dal 13° indice dell'i-node;
- i blocchi 141-200 sono indicizzati da un blocco indice di secondo livello, indicizzato dal primo elemento di un blocco indice di primo livello puntato dal 14° indice dell'i-node;
- il 15° indice nell'i-node non è utilizzato.

Quindi, assumendo che l'i-node sia in memoria e che all'inizio non via sia nulla in cache:

- il blocco 10 del file è letto con un solo accesso;
- il blocco 190 richiede l'accesso al blocco indice di primo livello puntato dal 14° puntatore dell'i-node, quindi l'accesso al primo blocco indice di secondo livello, quindi l'accesso al file (3 accessi);
- la lettura sequenziale dei primi 50 blocchi richiede 12 accessi ai primi 12 blocchi, un accesso al blocco indice di primo livello puntato dall'13° indice dell'i-node per trovare i successivi 38 puntatori, e 38 accessi per accedere al file, per un totale di 51 accessi.

3) Si considerino tre pathname in un file system di tipo Unix che descrivono la seguente situazione:

1. un file è identificato da due hard link aventi come pathname rispettivamente /usr/auce/miofile e /usr/auce/subdir/altrofile;
2. il file di pathname /usr/auce/unfile è un link simbolico verso il file /usr/auce/miofile.

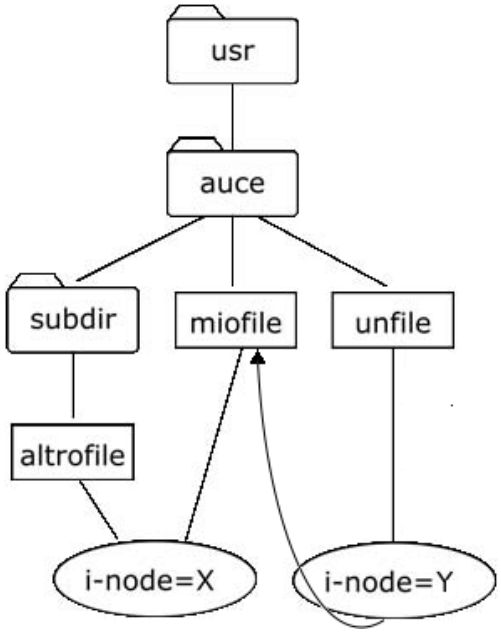
Illustrare, motivando la risposta, che cosa succede ai file identificati da questi pathname dopo l'esecuzione del comando

a) rm /usr/auce/miofile oppure, in alternativa, del comando

b) rm /usr/auce/unfile

Soluzione

La situazione può essere descritta da questa figura, nella quale la freccia dall'i-node Y all'elemento di directory "miofile" rappresenta il link simbolico, che è di fatto un file che ha come contenuto il pathname di un altro file.



Dopo l'esecuzione del comando *a* il link hard /usr/auce/miofile viene rimosso, e il contatore di link al file viene decrementato di 1. Dal momento che il file è anche identificato dall'altro link hard /usr/auce/subdir/altrofile, il contatore rimane > 0 quindi il file non viene cancellato.

Il link simbolico /usr/auce/unfile è un file che ha come contenuto il pathname /usr/auce/miofile, che è stato rimosso, quindi non punta più ad alcun file, e un tentativo di utilizzarlo come nome di file produce errore.

Se invece si esegue il comando *b*, si rimuove solo il file /usr/auce/unfile, e non si ha alcun effetto sul file identificato dagli altri due pathname.

4) Si considerino, nel file system di Unix, due file di nome file_A e file_B che contengono rispettivamente il testo “Io sono il file A” e “Io sono il file B”. Data questa sequenza di comandi:

```
 1  $ ln file_A file_C
 2  $ ln -s file_B file_D
 3  $ rm file_A
 4  $ cat file_A
 5  $ cat file_B
 6  $ cat file_C
 7  $ cat file_D
 8  $ rm file_B
 9  $ mv file_C file_B
10  $ ln -s file_B file_A
11  $ cat file_A
12  $ cat file_B
13  $ cat file_C
14  $ cat file_D
```

dire che cosa viene visualizzato dai comandi alle righe 4–7 e 11–14, motivando la risposta. Si ricorda che:

`$ ln [-s] X Y`	crea un link di nome Y al file X (link simbolico se è specificata l’opzione -s)
`$ rm X`	cancella il file X
`$ mv X Y`	rinomina il file X dandogli pathname Y
`$ cat X`	visualizza il contenuto del file X

Soluzione

1	`$ ln file_A file_C`	crea un hard link a file_A con pathname file_C
2	`$ ln -s file_B file_D`	crea un link simbolico a file_B con pathname file_D
3	`$ rm file_A`	cancella il pathname file_A, il file esiste ed è accessibile con pathname file_C
4	`$ cat file_A`	$ no such file --> file_A è stato rimosso
5	`$ cat file_B`	$ Io sono B
6	`$ cat file_C`	$ Io sono A --> file_C è hard link del precedente file_A
7	`$ cat file_D`	$ Io sono B --> file_D è symbolic link di file_B
8	`$ rm file_B`	cancella il pathname file_B, il file non è più accessibile
9	`$ mv file_C file_B`	rinomina file_C in file_B (che contiene “Io sono A”)
10	`$ ln -s file_B file_A`	crea un link simbolico a file_B con pathname file_A
11	`$ cat file_A`	$ Io sono A (hard link di file_B)
12	`$ cat file_B`	$ Io sono A (vedi sopra riga 9)
13	`$ cat file_C`	$ no such file --> file_C è stato ridenominato file_B
14	`$ cat file_D`	$ Io sono A (file_D è link simbolico a file_B che ora contiene “Io sono A”)

5) Si consideri la struttura di indicizzazione del file system di Unix, con dimensione del cluster di allocazione su disco di 4 Kbyte (4096 byte) e puntatori di 4 byte. Dato un file di 512 Mbyte (1 Mbyte = 2^{20} byte), rispondere alle seguenti domande, motivando la risposta:

a) quanti blocchi di indice, oltre all'i-node, sono necessari per indicizzare il file;
b) quanti accessi a disco sono necessari per leggere con accesso diretto il primo e l'ultimo byte del file, supponendo che il file sia già aperto e quindi il suo i-node sia già in memoria centrale.

Soluzione

Ci sono 4096 / 4 = 1024 puntatori in ogni blocco indice.

Un file da 512 Mbyte richiede 512M/ 4K = 128 K blocchi di dati, che a loro volta richiedono 128 K (128 * 1024) indici così organizzati, oltre all'i-node che indicizza direttamente i primi 12 blocchi:

- un blocco indice di primo livello per gli indici dei blocchi da 13 a 1024+12 = 1036
- un blocco indice di primo livello per indirizzare gli indici di secondo livello
- 127 blocchi indice di secondo livello per i blocchi dati da 1037 a 128 K; l'ultimo blocco indice è occupato da 1024 - 12 indici.

Complessivamente quindi servono 129 blocchi di indice.

Per leggere il primo byte serve solo un accesso a disco per leggere il primo blocco dati dal momento che il suo indice è contenuto nell'i-node; per leggere l'ultimo byte sono necessari 3 accessi a disco: l'indice di 1° livello, l'indice di 2° livello e il blocco dei dati.

6) Un disco da 5400 giri/minuto presenta un tempo medio di seek tra cilindri adiacenti di 100 μS. Si supponga di dover servire quattro richieste che si trovano, nell'ordine, sui cilindri 80, 125, 450 e 500. Si calcoli il tempo complessivo medio di accesso all'insieme dei dati richiesti (a partire dal momento in cui la testina del disco è posizionata sulla traccia 80) nell'ipotesi che ogni blocco di dati sia contenuto in un solo settore, e che quindi il tempo di accesso sia definito solo dal tempo di seek e dal tempo di rotazione. E' possibile calcolare il tempo esatto di trasferimento? e il tempo minimo o massimo?

Soluzione

5400 giri/min corrispondono a 90 giri/sec; quindi si ha un tempo di rotazione di 11 mS/giro e un tempo medio di latenza = 5,5 mS, con un minimo di 0 mS e un massimo di 11 mS.

La testina impiega questi tempi medi per accedere alle tracce su cui si trovano i dati:

80 -->125 = 45 tracce, 4,5 mS
125 --> 450 = 325 tracce, 32,5 mS
450 --> 500 = 50 tracce, 5 mS

cui corrisponde un tempo medio di accesso: 4,5 + 32,5 +5 + (5,5 x 3) = 58,5 mS

Il tempo esatto di accesso non è calcolabile, perché dipende dalla posizione del dato sulla traccia, che non è conosciuta, e perché i tempi di seek sono tempi medi che possono presentare variazioni.

I tempo minimo e massimo di accesso sono calcolabili solo rispetto alla posizione del dato sulla traccia, quindi considerano le variazioni del tempo di latenza, perché il tempo di seek è un valore medio:

minimo = 4,5 + 32,5 + 5 = 42 mS (latenza zero in tutti e tre i casi)
max = 4,5 + 32,5 + 5 + (11 x 3) = 85 mS (latenza massima in tutti e tre i casi)

7) In un disco di 80Gbyte diviso in cluster di 4 Kbyte quanti byte (o Kbyte o Mbyte) occupa la bitmap che rappresenta la posizione delle aree libere?Quanto occupa complessivamente un indice FAT32?

Soluzione

Ogni bit della bitmap rappresenta un cluster di 4 Kbyte. Nel disco ci sono 80Gbyte / 4Kbyte = 20M cluster, rappresentati da 20 Mbit = 2,5 Mbyte.

Ogni cluster è indirizzato da un puntatore di 4 byte. 20 M cluster richiedono 80 Mbyte per i puntatori, quindi la FAT occupa 80 Mbyte. Si noto che normalmente la FAT è duplicata per motivi di sicurezza, quindi lo spazio occupato è il doppio, 160 Mbyte.