

**Università Ca' Foscari Venezia – Corso di Laurea in Informatica  
Sistemi Operativi A**

**Problemi sul file system e sulla gestione della memoria secondaria**

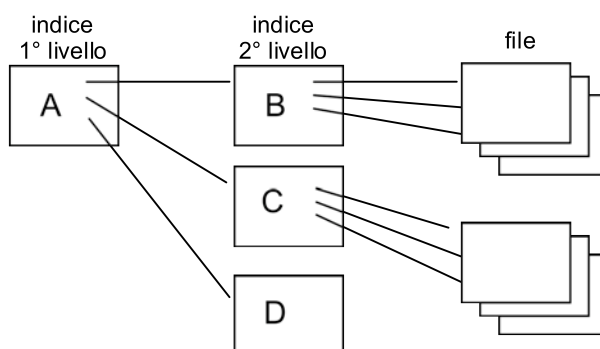
Prof. Augusto Celentano, anno accademico 2007–2008

**Problema 1**

Illustrare il meccanismo di allocazione indicizzata dei file. Si faccia riferimento ad un file system 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? Quanto 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 questo schema:



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 ( $1M = 1024 \times 1024$ )

2048 blocchi richiedono 2048 indici =  $2048 / 128 = 16$  blocchi di 2° livello.

Complessivamente 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  $400 \bmod 128 = 16$ -esimo del blocco.

In generale, l'n-esimo indirizzo occupa i byte da  $(n - 1) \times 4$  a  $(n - 1) \times 4 + 3$ .

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  $236.448 \bmod 512 = 416$ . 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  $128 \times 128 = 16384$  puntatori, e quindi blocchi del file. Il file può avere una dimensione massima di  $16384 \times 512 = 8$  Mbyte (8.388.608 byte), e occupa complessivamente  $16384 + 128 + 1 = 16513$  blocchi

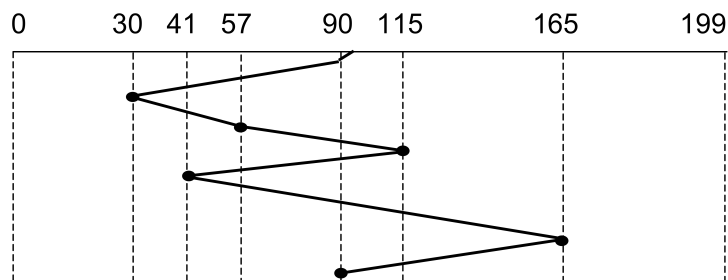
## Problema 2

Nella coda delle richieste di un disco composto da 200 cilindri (numerate da 0 a 199) si trovano, nell'ordine, le richieste dei dati che si trovano sui cilindri 30, 57, 115, 41, 165, 90. La testina ha eseguito l'ultimo movimento portandosi dal cilindro 91 al cilindro 90. Si supponga che lo spostamento da un cilindro ad un altro richieda un tempo pari a  $30 \mu\text{S}$  per cilindro, con un minimo per ogni spostamento di  $500 \mu\text{S}$ . Quanto tempo richiede, complessivamente, l'accesso ai dati per le seguenti politiche di scheduling?

First Come First Served (FCFS)  
 Shortest Seek Time First (SSTF)  
 LOOK (o SCAN)  
 C-LOOK (o C-SCAN)

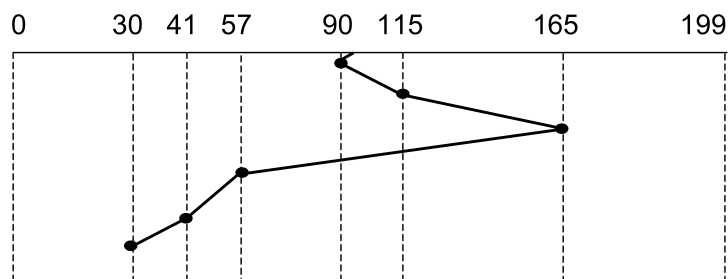
## Soluzione

FCFS



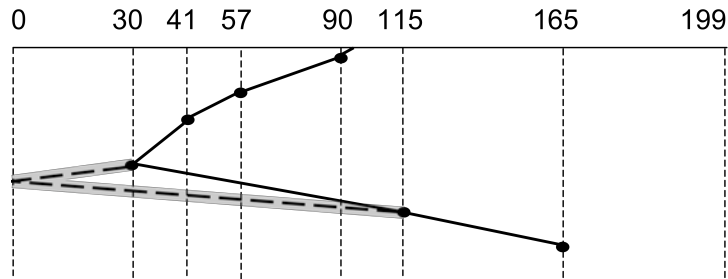
Spostamento	Tempo
$90 - 30 = 60$	$60 * 30 = 1.800 \mu\text{S} = 1,8 \text{ mS}$
$30 - 57 = 27$	$27 * 30 = 810 \mu\text{S} = 0,81 \text{ mS}$
$57 - 115 = 58$	$58 * 30 = 1.740 \mu\text{S} = 1,74 \text{ mS}$
$115 - 41 = 74$	$74 * 30 = 2.220 \mu\text{S} = 2,22 \text{ mS}$
$41 - 165 = 124$	$124 * 30 = 3.720 \mu\text{S} = 3,72 \text{ mS}$
$165 - 90 = 75$	$75 * 30 = 2.250 \mu\text{S} = 2,25 \text{ mS}$
Totale	12,54 mS

SSTF



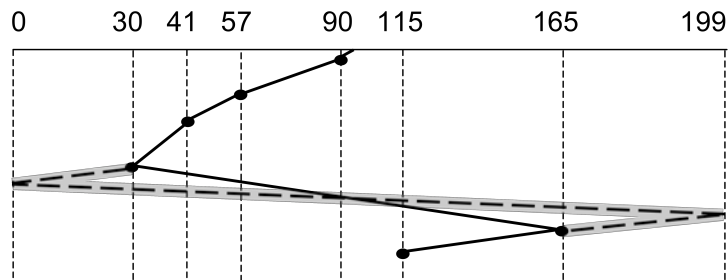
Spostamento	Tempo
$90 - 90 = 0$	$0 * 30 = 0 \mu\text{S} = 0 \text{ mS}$
$90 - 115 = 25$	$25 * 30 = 750 \mu\text{S} = 0,75 \text{ mS}$
$115 - 165 = 50$	$50 * 30 = 1.500 \mu\text{S} = 1,5 \text{ mS}$
$165 - 57 = 108$	$108 * 30 = 3.240 \mu\text{S} = 3,24 \text{ mS}$
$57 - 41 = 16$	$16 * 30 = 480 \mu\text{S} \rightarrow 500 \mu\text{S} = 0,5 \text{ mS}$
$41 - 30 = 11$	$11 * 30 = 330 \mu\text{S} \rightarrow 500 \mu\text{S} = 0,5 \text{ mS}$
Totale	6,49 mS

## LOOK (SCAN)



Spostamento	Tempo
$90 - 90 = 0$	$0 * 30 = 0 \mu S = 0 \text{ mS}$
$90 - 57 = 33$	$33 * 30 = 990 \mu S = 0,99 \text{ mS}$
$57 - 41 = 16$	$16 * 30 = 480 \mu S \rightarrow 500 \mu S = 0,5 \text{ mS}$
$41 - 30 = 11$	$11 * 30 = 330 \mu S \rightarrow 500 \mu S = 0,5 \text{ mS}$
$30 - 0 = 30$	$30 * 30 = 900 \mu S = 0,9 \text{ mS}$
$0 - 115 = 115$	$115 * 30 = 3.450 \mu S = 3,45 \text{ mS}$
$30 - 115 = 85$	$85 * 30 = 2.550 \mu S = 2,55 \text{ mS}$
$115 - 165 = 50$	$50 * 30 = 1.500 \mu S = 1,5 \text{ mS}$
Totale	6,04 mS (7,84 mS)

## C-LOOK (C-SCAN)



Spostamento	Tempo
$90 - 90 = 0$	$0 * 30 = 0 \mu S = 0 \text{ mS}$
$90 - 57 = 33$	$33 * 30 = 990 \mu S = 0,99 \text{ mS}$
$57 - 41 = 16$	$16 * 30 = 480 \mu S \rightarrow 500 \mu S = 0,5 \text{ mS}$
$41 - 30 = 11$	$11 * 30 = 330 \mu S \rightarrow 500 \mu S = 0,5 \text{ mS}$
$30 - 0 = 30$	$30 * 30 = 900 \mu S = 0,9 \text{ mS}$
$0 - 199 = 199$	$199 * 30 = 5.970 \mu S = 5,97 \text{ mS}$
$199 - 165 = 34$	$34 * 30 = 1.020 \mu S = 1,02 \text{ mS}$
$30 - 165 = 135$	$135 * 30 = 4.050 \mu S = 4,05 \text{ mS}$
$165 - 115 = 50$	$50 * 30 = 1.500 \mu S = 1,5 \text{ mS}$
Totale	7,54 mS (11,38 mS)

## Varianti:

1. Si ipotizzi che lo spostamento da un cilindro ad un altro richieda un tempo pari a  $200 \mu S$  per l'avvio del movimento +  $30 \mu S$  a cilindro.
2. Si ipotizzi che lo spostamento da un cilindro ad un altro richieda un tempo pari a  $30 \mu S$  a cilindro, e che l'inversione del senso di movimento della testina richieda un tempo di  $500 \mu S$ .

### Problema 3

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

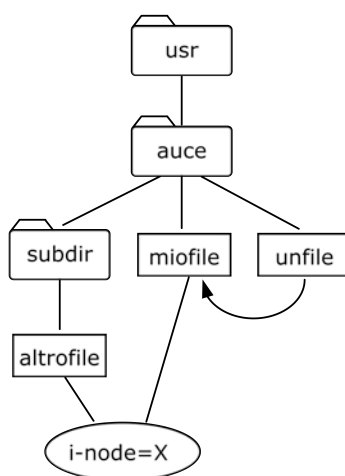
1. un file è identificato da due hard link aventi come pathname rispettivamente `/usr/auce/miofile` e `/usr/auce/subdir/altrofile`;
2. il file di pathname `/usr/auce/unfile` è un link simbolico verso il file `/usr/auce/miofile`.

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

- a. `rm /usr/auce/miofile` , oppure del comando
- b. `rm /usr/auce/unfile`

### Soluzione

La situazione può essere descritta da questa figura



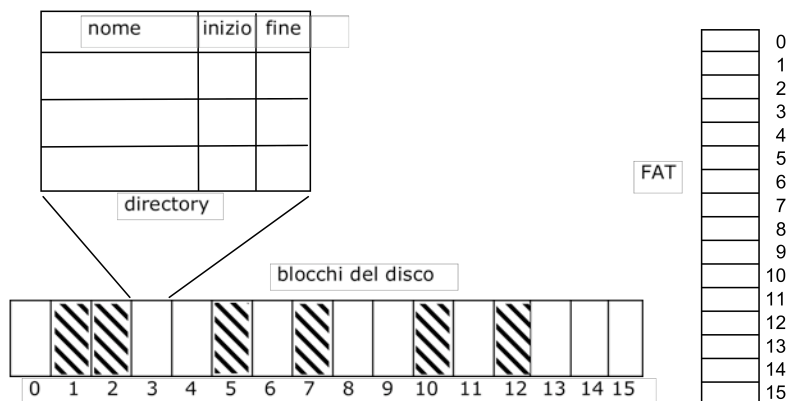
Dopo l'esecuzione del comando a il link hard `/usr/auce/miofile` viene rimosso, e il contatore di link al file viene decrementato di 1. Dal momento che il file è anche identificato dall'altro link hard `/usr/auce/subdir/altrofile`, il contatore rimane  $> 0$  quindi il file non viene cancellato.

Il link simbolico `/usr/auce/unfile` ha come valore 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.

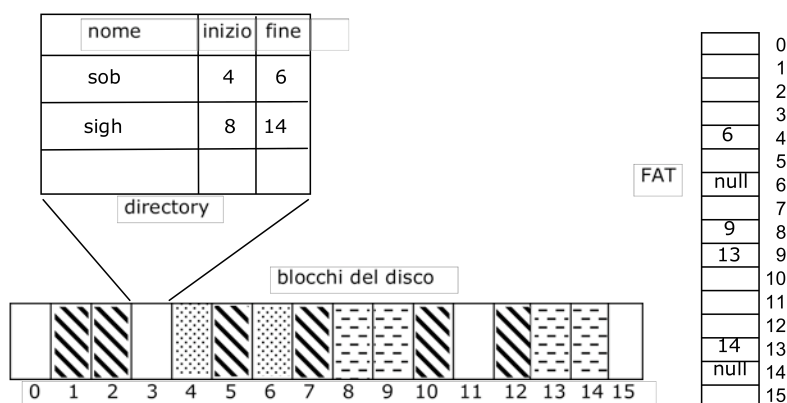
#### Problema 4

Con riferimento al seguente schema che mostra graficamente un'ipotetica struttura di un file system per una memoria di 16 blocchi, illustrare e commentare un esempio di allocazione di due file di nome **sob** e **sigh** rispettivamente di 2 e 4 blocchi secondo una tecnica di allocazione basata su FAT. I blocchi con sfondo tratteggiato sono già occupati. Dov'è allocata la FAT?



#### Soluzione

Questa è una soluzione possibile ma non è l'unica. I blocchi 11 e 15 hanno un contenuto non specificato che dipende dalla storia passata (es. file allocati e poi cancellati)



Nei sistemi Windows la FAT è allocata in una zona della partizione non dedicata ai file, nell'esempio (molto semplice e di dimensioni molto ridotte) potrebbe essere allocata nel blocco 0 (che dovrebbe tuttavia essere marcato come occupato), o in uno degli altri blocchi già occupati.

### Problema 5

Un disco da 5400 giri/minuto presenta un tempo medio di seek tra cilindri adiacenti di 100  $\mu$ S. 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 praticamente 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

$$5400 \text{ giri/min} = 90 \text{ giri/sec} = 11 \text{ mS/giro}$$

$$\text{tempo medio di latenza} = 5,5 \text{ mS, minimo } 0 \text{ mS, max } 11 \text{ mS}$$

$$\begin{aligned} \text{percorso:} \quad & 80 \rightarrow 125 = 45 \text{ tracce, } 4,5 \text{ mS} \\ & 125 \rightarrow 450 = 325 \text{ tracce, } 32,5 \text{ mS} \\ & 450 \rightarrow 500 = 50 \text{ tracce, } 5 \text{ mS} \end{aligned}$$

$$\text{tempo medio di accesso: } 4,5 + 32,5 + 5 + (5,5 \times 3) = 58,5 \text{ mS}$$

tempo esatto di accesso: non è calcolabile, dipende dalla posizione del dato sulla traccia, che non è conosciuta

tempo minimo, max 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 dato solo per valore medio:

$$\begin{aligned} \text{minimo} &= 4,5 + 32,5 + 5 = 42 \text{ mS} \\ \text{max} &= 4,5 + 32,5 + 5 + (11 + 3) = 85 \text{ mS} \end{aligned}$$

## Problema 6

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:

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

## Soluzione

$512 \text{ byte} / 4 \text{ byte} = 128$  puntatori per ogni blocco indice.

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 vi 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 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

## **Problema 7**

Si considerino due processi A e B. Il processo A esegue un numero molto elevato (in un certo periodo di tempo) di accessi random a disco sui cilindri compresi nell'intervallo 10-50, mentre il processo B esegue, nello stesso periodo di tempo, un numero più ridotto di accessi sui cilindri compresi nell'intervallo 90-95, accedendo ad essi in modo sequenziale. Le richieste arrivano al disco da parte dei due processi in modo non prevedibile. Illustrare brevemente almeno due politiche (a scelta) di scheduling degli accessi a disco, commentandone l'efficacia nella situazione sopra descritta.

## **Soluzione**

SSTF: il processo A, eseguendo un numero elevato di accessi ai cilindri 10-50, non consente al processo B di vedere soddisfatte le sue richieste perché i suoi accessi sono più distanti rispetto agli accessi di A. Se il periodo di tempo in cui si verifica questa situazione è lungo il processo B è soggetto ad una forma di starvation, che termina solo quando il processo A rallenta le sue richieste.

LOOK: finché al disco arrivano solo richieste dal processo A, queste vengono servite in ordine di posizione sul disco. Rispetto alla politica SSTF è possibile che l'efficienza sia minore, dal momento che le richieste sono random mentre la scansione del disco è sequenziale. Quando arriva una richiesta dal processo B, le richieste di B vengono servite appena la testina ha finito di servire la richiesta di A in posizione più alta. Se la velocità con cui arrivano le richieste è dello stesso ordine di grandezza della velocità di accesso al disco, o minore, le richieste vengono servite tutte prima di riprendere con le richieste di processo A. Complessivamente i processi si alternano in modo più regolare, ma A viene rallentato.

SCAN: l'esplorazione completa del disco fa sì che le richieste di A e di B si alternino in modo ancor più regolare, ma il processo A subisce un ulteriore rallentamento dal momento che servita la sua richiesta sulla traccia più alta la testina prosegue comunque verso la fine del disco anche se non ci sono richieste da parte di B.



### **Problema 8**

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 rappresenta un cluster di 4 Kbyte.

Nel disco ci sono  $80\text{Gbyte} / 4\text{Kbyte} = 20\text{M}$  cluster, rappresentati da  $20\text{ Mbit} = 2,5\text{ Mbyte}$ .

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.