Esercizio FileSystem 1

In un disco con blocchi di 1 Kbyte (= 2^{10} byte), è definito un file system FAT.
Gli elementi della FAT sono in corrispondenza biunivoca con i blocchi fisici del disco.
Ogni elemento ha lunghezza di 3 byte e indirizza un blocco del disco.
Ogni file è descritto da una lista concatenata di indirizzi di blocchi, realizzata sulla FAT.
Il primo blocco di ogni file è identificato dalla coppia (*nomefile, indiceblocco)* contenuto nella rispettiva directory.

1. Qual è la massima capacità del disco, espressa in blocchi e in byte?
2. Quanti byte occupa la FAT?
3. Supponendo che il file *pippo* occupi (ordinatamente) i blocchi fisici 15, 30, 16, 64 e 40, quali sono gli elementi della FAT che descrivono il file e quale è il loro contenuto?

Soluzione:

1. Capacità del disco: 2^{24} blocchi → 2^{34} byte
2. Lunghezza della FAT: $2^{24} * 3$ byte → 48 MByte
3. Elementi della FAT che indirizzano il file e loro contenuto:

ELEMENTO FAT	CONTENUTO
15	30
30	16
16	64
64	40
40	Ø

Esercizio FileSystem 2

Un sistema operativo gestisce la memoria con paginazione dinamica con pagine di 2 Kbyte e utilizza un file system di tipo FAT-32 (con indirizzi a 32 bit) con blocchi di 2 Kbyte per gestire un disco di 10 Gbyte. La FAT è allocata sul disco a partire dal blocco 3.
Nel file system il file `foto` contenuto nella directory `personale` occupa 9 blocchi logici allocati come segue:

Blocco logico:	0	1	2	3	4	5	6	7	8
Blocco fisico:	398.203	1.716.882	106.987	2.448.123	12.186	65.637	4.876.999	3.161.002	908.543

Ad un dato istante di tempo, quando nessun blocco della FAT è caricato in memoria principale e la directory `personale` è caricata in memoria, il sistema operativo riceve una richiesta di lettura dei caratteri del file `foto` compresi tra 8.000 e 14.000 (estremi inclusi).
Si chiede:

1. Quali blocchi logici del file `foto` vengono letti dal sistema operativo
2. Quali blocchi fisici del file `foto` vengono letti dal sistema operativo
3. Quali blocchi fisici contenenti la FAT vengono letti dal sistema operativo
4. Quanti page fault causa l'operazione di lettura nell'ipotesi che i buffer destinati ad accogliere i dati letti dal file siano già allocati e presenti in memoria.

SOLUZIONE

1. Blocco logico del file `foto` che contiene il byte 8.000: 8.000 **div** 2048 = 3;
   Blocco logico del file `foto` che contiene il byte 14.000: 14.000 **div** 2048 = 6;
   Blocchi logici del file `foto` letti dal sistema operativo: 3, 4, 5, 6
2. Blocchi fisici del file `foto` letti dal sistema operativo: 2.448.123; 12.186; 65.637; 4.876.999
3. Blocchi della FAT letti:
   Ogni blocco del disco contiene 2048/4 = 512 puntatori della FAT, di conseguenza:
   - Il puntatore 2.448.123 è contenuto nel blocco 2.448.123 **div** 512= 4781 della FAT, che corrisponde al blocco 3+4781=4784 del disco
   - Il puntatore 12.186 è contenuto nel blocco 12.186 **div** 512= 23 della FAT, che corrisponde al blocco 3+23=26 del disco
   - Il puntatore 65.637 è contenuto nel blocco 65.637 **div** 512= 131 della FAT, che corrisponde al blocco 3+128=131 del disco
   - Il puntatore 4.876.999 è contenuto nel blocco 4.876.999 **div** 512= 9525 della FAT, che corrisponde al blocco 3+9525= 9528 del disco
   Quindi i blocchi fisici da leggere contenenti la FAT sono: 4784, 26, 131 e 9528
4. Page fault causati dall'operazione di lettura: 4 per la lettura della FAT

Esercizio FileSystem 3

In un file system UNIX i blocchi del disco hanno ampiezza di 1Kbyte e gli i-node contengono 10 indirizzi diretti e 3 indirizzi indiretti. Tutti gli indirizzi hanno una lunghezza di 4 byte.
Si chiede:

1. la massima capacità del disco che ospita il file system, in blocchi e in byte
2. la massima dimensione dei file indirizzabili dall'i-node, in blocchi e in byte.

SOLUZIONE

1. Massima capacità del disco che ospita il file system: 2^{32} blocchi; $2^{32}*2^{10}= 2^{42}$ byte (4 Tbyte)
2. considerato che:

   - lo i-node indirizza direttamente 10 blocchi
   - il blocco indiretto di primo livello puntato dall'indirizzo indiretto semplice indirizza 2^{10} **div** $4 = 2^8$ blocchi dati
   - il blocco indiretto di secondo livello puntato dall'indirizzo indiretto doppio indirizza 2^{10} **div** $4 =2^8$ blocchi indiretti di primo livello, ciascuno dei quali indirizza 2^8 blocchi dati,
   - il blocco indiretto di terzo livello puntato dall'indirizzo indiretto triplo indirizza 2^{10} **div** 4 $=2^8$ blocchi indiretti di secondo livello, ciascuno dei quali indirizza 2^{10} **div** $4 =2^8$ blocchi indiretti di primo livello, ciascuno dei quali indirizza 2^8 blocchi dati,

   la massima dimensione dei file indirizzabili dall'i-node è pari a:

   - $10 + 2^8 + 2^{16} + 2^{24} \sim 2^{24}$ blocchi (precisamente 10+ 256+ 65536+ 16777216 =16.843.018 blocchi),
   - ovvero ~ 16 Gbyte (precisamente 16.843.018 Kbyte).

Esercizio FileSystem 4

In un file system UNIX i blocchi del disco hanno ampiezza di 1Kbyte e i puntatori ai blocchi sono a 32 bit. Gli i-node contengono, oltre agli altri attributi, 10 puntatori diretti e 3 puntatori indiretti.
Tenendo presente che il primo blocco del disco ha indice logico 0, si chiede:

1. il numero di puntatori che possono essere contenuti in un blocco indiretto;
2. l'indice logico del primo blocco e dell'ultimo blocco indirizzabili con puntatori diretti;
3. l'indice logico del primo blocco e dell'ultimo blocco indirizzabili con indirizzamento indiretto semplice;
4. l'indice logico del primo blocco e dell'ultimo blocco indirizzabili con indirizzamento indiretto doppio;
5. l'indice logico del primo blocco e dell'ultimo blocco indirizzabili con indirizzamento indiretto triplo;

Considerato il file (aperto) individuato dal file descriptor *fd,* la cui lunghezza corrente (in byte) è 278.538 e il cui *i-node* contiene i seguenti puntatori a blocchi:

Puntatore	0	1	2	3	4	5	6	7	8	9	10	11	12
Valore del puntatore	100	101	102	120	121	122	300	301	302	303	500	700	--

dove i blocchi indiretti 500, 700, e 800 hanno i seguenti contenuti parziali:

Blocco 500:

Indice di elemento nel blocco	0	1	2	3	4	5	……
Valore del puntatore	304	305	306	307	308	309	……

Blocco 700:

Indice di elemento nel blocco	0	1	2	3	4	5	……
Valore del puntatore	800	801	802	850	851	852	……

Blocco 800:

Indice di elemento nel blocco	0	1	2	3	4	5	……
Valore del puntatore	1200	1201	1202	1203	1204	1205	……

si chiede inoltre:

6. il numero di blocchi che compongono il file;
7. quali sono i blocchi indiretti che vengono letti per eseguire l'operazione *read(fd,&buf,1*) quando lo I/O pointer ha valore 12.298
8. quali sono i blocchi indiretti che vengono letti per eseguire l'operazione *read(fd,&buf,1)* quando lo I/O pointer ha valore 273.428.

SOLUZIONE

1. il numero di puntatori che possono essere contenuti in un blocco indiretto è 2^8= 256;
2. il primo e l'ultimo blocco indirizzabili con puntatori diretti hanno rispettivamente indici logici 0 e 9;
3. il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto semplice hanno rispettivamente indici logici 10 e (10+ 2^8)- 1= 265;
4. il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto doppio hanno rispettivamente indici logici 10+ 2^8= 266 e (10+ 2^8+ 2^16)- 1= 65.801;
5. il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto triplo hanno rispettivamente indici logici 10+ 2^8+ 2^16= 65.802 e (10+ 2^8+ 2^16+ 2^24)- 1= 16.843.017;
6. l'ultimo carattere del file è contenuto nel blocco 278.538 ***div*** 2^10= 272; quindi il file è composto da 273 blocchi
7. il carattere *12.298,* sul quale è posizionato il puntatore di lettura, è contenuto nel blocco di indice logico *12.298* ***div*** 2^10= 12, indirizzato dal puntatore di indice 2 e valore 306 del blocco indiretto di primo livello 500, individuato dal puntatore 10 dello *i-node*. Quindi per eseguire l'operazione *read(fd,&buf,1*) deve essere letto il blocco indiretto 500.
8. il carattere *273.428,* sul quale è posizionato il puntatore di lettura, è contenuto nel blocco di indice logico *273.428* ***div*** 2^10= 267, individuato dal puntatore di indice 1 e valore 1201 nel blocco indiretto di primo livello 800, a sua volta individuato dal puntatore di indice 0 del blocco indiretto di secondo livello 700 corrispondente al puntatore 11 dello i-node. Quindi per eseguire l'operazione *read(fd,&buf,1*) devono essere letti il blocco indiretto di secondo livello 700 e il blocco indiretto di primo livello 800.

Esercizio FileSystem 5

Si consideri la chiamata *read(4, &buf, 2000)* di un sistema simile a UNIX, dove il file descriptor 4 corrisponde all'i-node 15.
Lo i-node contiene 5 indirizzi di blocchi diretti, che hanno rispettivamente valore 512, 567, 45, 34, 28, oltre agli indirizzi di 2 blocchi indiretti. La lunghezza degli indirizzi è di 2 byte.
Gli indirizzi 0, 1, 2, 3, 4 del blocco indiretto di primo livello raggiungibile con indirizzamento indiretto semplice hanno ordinatamente i valori 56, 47, 67, 89, 23.
I blocchi del disco hanno ampiezza di 1024 byte e la lunghezza corrente del file è di 10.000 byte.
Il puntatore alla posizione corrente di lettura ha il valore 8500.
Domande:

1. Quali blocchi fisici vengono letti per eseguire l'operazione ?
2. Quanti caratteri vengono copiati in *buf* da ogni blocco interessato alla lettura?
3. Qual è il valore intero restituito dalla chiamata?

Soluzione

Considerato che il file contiene 10.000 byte, l'operazione restituirà 1.500 caratteri e terminerà al raggiungimento del carattere di EOF, successivo all'ultimo carattere di informazione.
Il puntatore di lettura è posizionato sul carattere 8500 **mod** 1024 = 308 del blocco logico 8500 **div** 1024 = 8.
Il carattere EOF occupa la posizione 10.000 **mod** 1024 = 784 del blocco logico 10.000 **div** 1024 = 9.
Ogni blocco indice contiene 1024/2= 512 indirizzi.
Il blocco fisico corrispondente al blocco logico di indice 8 è individuato dall'indirizzo (8-5) **mod** 512= 3 del blocco indiretto di primo livello (8-5) **div** 512= 0, a sua volta individuato con indirizzamento indiretto semplice.
Il blocco fisico corrispondente al blocco logico di indice 9 è individuato dall'indirizzo (9-5) **mod** 512= 4 del blocco indiretto di primo livello (8-5) **div** 512= 0, a sua volta individuato con indirizzamento indiretto semplice.
Quindi:
1. Considerato che il file è necessariamente già aperto e che pertanto lo i-node è caricato in memoria, per eseguire l'operazione si leggono i seguenti blocchi:

   - blocco indiretto di primo livello di indice 0, raggiungibile con indirizzamento indiretto semplice
   - Blocco fisico 89, corrispondente al blocco logico 8 e individuato dall'indirizzo 3 di questo blocco indiretto. Dal blocco fisico 89 si leggono 1024-308= 716 caratteri e pertanto rimangono da leggere 2000-716= 1284 caratteri
   - Blocco fisico 23, corrispondente al blocco logico 9 e individuato dall'indirizzo 4 del medesimo blocco indiretto. Dal blocco fisico 23 si leggono caratteri fino a raggiungere il carattere EOF: quindi 784 caratteri

2. Numero di caratteri copiati da ciascun blocco interessato alla lettura:
   - 716 caratteri dal blocco 89
   - 784 caratteri dal blocco 23

3. Valore restituito dalla chiamata: 716+ 784= 1500

Esercizio FileSystem 6

In un file system UNIX dove gli i-node occupano 1 blocco, si consideri il file individuato dal pathname *compiti/terzo_appello,* relativo alla directory corrente. La directory corrente è caricata in memoria, mentre la directory *compiti* (che occupa 1 blocco) e il rispettivo i-node risiede su disco. Tutti gli i-node occupano 1 blocco.
Gli i-node della directory *compiti* e del file *terzo_appello* sono caricati, rispettivamente, nei blocchi fisici 35 e 71.
Gli indirizzi diretti dello i-node della directory *compiti* hanno i valori:

Indirizzo diretto	0	1	2	3	4	5	6	7	8	9
Valore	25									

Gli indirizzi diretti dello i-node del file *terzo_appello* hanno i valori:

Indirizzo diretto	0	1	2	3	4	5	6	7	8	9
Valore	29	30	42	64	65	66	78	90	91	93

Quali sono i blocchi fisici da leggere per portare in memoria i primi 2 blocchi del file *terzo_appello,* supponendo che questo file sia già stato aperto?

Soluzione

Lo i-node del file *terzo_appello* risiede in memoria, dove è stato caricato al momento dell'apertura.

I blocchi, che devono essere letti sul disco sono i seguenti:

1. blocco 35 , che contiene lo i-node di *compiti*
2. blocco 25 , che contiene la directory *compiti*
3. blocco 29 , che contiene il primo blocco di *terzo_appello*
4. blocco 30 , che contiene il secondo blocco di *terzo_appello*

Esercizio FileSystem 7

In un file system UNIX si consideri il file */usr/tizio/appunti/esercitazione,* creato dall'utente *tizio*.
I diritti associati alle directory *usr, tizio, appunti* e al file *esercitazione* sono i seguenti

	owner (r w x)	*group (r w x)*	*others (r w x)*
usr	*1 0 1*	*1 0 1*	*1 0 1*
tizio	*1 1 1*	*0 0 1*	*0 0 1*
appunti	*1 1 1*	*1 0 1*	*1 0 0*
esercitazione	*1 1 0*	*1 1 0*	*1 0 0*

Quali tra le operazioni di lettura, scrittura e cancellazione possono eseguite sul file *esercitazione* dall'utente *caio* se:

1. *caio* e *tizio* appartengono allo stesso gruppo;
2. *caio* e *tizio* appartengono gruppi diversi.

Soluzione

IPOTESI	LETTURA	SCRITTURA	CANCELLAZIONE
caio e *tizio* appartengono allo stesso gruppo	SI	SI	NO
caio e *tizio* appartengono a gruppi diversi.	NO	NO	NO

Esercizio FileSystem 8

Ricordando che per la creazione di un *pipe* il sistema UNIX utilizza la chiamata di sistema:

```
int pipe (int fd[2])
```

si chiede quali chiamate di sistema sono utilizzate per:

1. scrivere k byte nel pipe
2. leggere k byte dal pipe
3. chiudere il pipe per la scrittura.

Soluzione

1. per scrivere k byte: `write(fd[1], *buff, k)`
2. per leggere k byte : `read(fd[0], *buff, k)`
3. per chiudere il pipe in scrittura: `close(fd[1])`

Esercizio FileSystem 9

Si consideri un File System simile a NTFS che alloca i file in sequenze contigue *(run)* individuate mediante coppie del tipo *(inizio, lunghezza),* dove *inizio* è l'indice del primo blocco fisico della sequenza e *lunghezza* esprime il numero di blocchi che la compongono. Ogni file è descritto da un *Master Record* che contiene, oltre ad altri attributi, una o più coppie *(inizio, lunghezza)*. Il File System è ospitato da un disco con *NCilindri= 50*, *NFacce= 4* e *NSettori= 20*. Il tempo necessario per percorrere un settore è di *0,1 msec* e il tempo medio di esecuzione di un'operazione di *seek* (comprensivo del ritardo rotazionale per raggiungere il primo settore indirizzato) è di *5 msec.*
A un certo tempo viene eseguita l'operazione *read(filename, &buffer, Ncaratteri),* per effetto della quale si leggono 10 blocchi a partire dal nono blocco del file, cioè da quello di indice logico 8. Nel *Master Record* del file *filename* sono definite, nell'ordine, i seguenti *run* contigui:

1. *(1525, 15)*
2. *(3170, 12)*

dove l'indice di blocco *1525* corrisponde alla terna *(cilindro= 19, faccia= 0, settore= 5)* e l'indice di blocco *3170* corrisponde alla terna *(cilindro= 39, faccia= 2, settore= 10).*
Si calcoli il tempo necessario per eseguire la lettura, supponendo che le teste di lettura scrittura siano inizialmente posizionate sul cilindro *12* e che tempo di esecuzione delle eventuali operazioni di *seek* sia sempre uguale a quello medio.

SOLUZIONE

- Il primo blocco da leggere ha indice logico 8 e indice fisico 1533; pertanto l'indice del primo blocco fisico estratto dal primo *run* 1 è il blocco 1533
- Numero di blocchi estratti dal primo *run*: 15- 8= 7 blocchi
- Numero di cilindri su cui è distribuita il primo *run*: 1
- Tempo necessario per estrarre i blocchi del primo *run*: 7 * 0,1= 0,7 msec
- Indice del primo blocco estratto dal secondo *run*: blocco 3170
- Numero di blocchi estratti dal secondo *run*: 3 blocchi
- Numero di cilindri su cui è distribuito il secondo *run*: 1
- Tempo necessario per estrarre i blocchi della sequenza 2: 3* 0,1= 0,3 msec
- Numero di operazioni di *seek* eseguite: 2
- Tempo totale impiegato: 2*5 + 1,7+ 0,3= 11 msec.

Esercizio FileSystem 10

Data la seguente matrice di protezione:

	File 1	File 2	File 3	File 4	Scanner	Stampante
Utente Mario	R,W	X	R,X		R	
Utente Franco		R,X	W			
Utente Rosa	R	W				W
Utente Nina	R		R,X	R,W	R	W

convertirla in:

1. liste di capability
2. liste di controllo degli accessi

SOLUZIONE

1. liste di capability:

Utente Mario: <File1:R,W> → <File2:X> → <File3:R,X> → <Scanner:R>
Utente Franco: <File2:R,X> → <File3:W>
Utente Rosa: <File1:R> → <File2:W> → <Stampante:W>
Utente Nina: <File1:R> → <File3:R,X> → <File4:R,W> → <Scanner:R> → <Stampante:W>

2. liste di controllo degli accessi:

File1: <Mario:R,W> → <Rosa:R> → < Nina:R>
File2: <Mario:X> → <Franco:R,X> → <Rosa:W>
File3: <Mario:R,X> → <Franco:W> → <Nina:R,X>
File4: <Nina:R,W>
Scanner: <Mario:R> → <Nina:R>
Stampante: <Rosa:W> → <Nina:W>