Sistemi Operativi e Laboratorio, Prova del 25/5/2016

Nome: _____ Cognome: _____ Matricola:_____ fila:___ posto: ___ corso:____

Esercizio 1 (4 punti)

Un sistema con processi A, B, C, D, E e risorse dei tipi R1, R2, R3, R4, ha raggiunto lo stato mostrato nelle tabelle seguenti, che è uno stato sicuro:

Assegnazione attuale				
	R1	R2	R3	R4
A	2	1	0	0
B			1	
C	1	1		1
D			1	1
E	1	1	1	

Esigenza Attuale				
	R1	R2	R3	R4
A	0	2	3	1
B	2	2	2	0
C	1	0	2	0
D	3	2	1	0
E	2	0	1	0

Molteplicità			
R1	R2	R3	R4
5	4	5	2

Disponibilità			
1	1	2	0

Successivamente, i processi A e D eseguono in sequenza le seguenti richieste:

1. A richiede 1 istanza di R3
2. D richiede 1 istanza di R2

Il gestore delle risorse applica l'algoritmo del banchiere per evitare lo stallo. Verificare se il gestore assegna le risorse richieste.

Soluzione

Stato raggiunto dopo l'assegnazione di 1 istanze di R3 al processo A:

Assegnazione attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Esigenza Attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Molteplicità			
R1	R2	R3	R4
5	4	5	2

Disponibilità			

- Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
- Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
- Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
- Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
- Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
- La richiesta lascia il sistema in uno stato sicuro? …… Può essere accettata? ……...
- Se la richiesta non viene accettata in quale stato va il processo A? _____

Stato raggiunto dopo l'assegnazione di 1 istanza di R2 al processo D:

Assegnazione attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Esigenza Attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Molteplicità			
R1	R2	R3	R4
5	4	5	2

Disponibilità			

Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..
Il processo _____ può/non può terminare, il vettore disponibilità diventa: ……..

- La richiesta lascia il sistema in uno stato sicuro? …… Può essere accettata? ……...
- Se la richiesta non viene accettata in quale stato va il processo D? _____

Soluzione

Sistemi Operativi e Laboratorio, Prova del 25/5/2016

Stato raggiunto dopo l'assegnazione di 1 istanze di R3 al processo A:

Assegnazione attuale				
	R1	R2	R3	R4
A	2	1	1	0
B			1	
C	1	1		1
D			1	1
E	1	1	1	

Esigenza Attuale				
	R1	R2	R3	R4
A	0	2	2	1
B	2	2	2	0
C	1	0	2	0
D	3	2	1	0
E	2	0	1	0

Molteplicità			
R1	R2	R3	R4
5	4	5	2

Disponibilità			
1	1	1	0

Nessun processo può terminare
La richiesta lascia il sistema in uno stato NON sicuro e non è accettata.
Il processo A va in stato di ATTESA

Stato raggiunto dopo l'assegnazione di 1 istanza di R2 al processo D:

Assegnazione attuale				
	R1	R2	R3	R4
A	2	1	0	0
B			1	
C	1	1		1
D		1	1	1
E	1	1	1	

Esigenza Attuale				
	R1	R2	R3	R4
A	0	2	3	1
B	2	2	2	0
C	1	0	2	0
D	3	1	1	0
E	2	0	1	0

Molteplicità			
R1	R2	R3	R4
5	4	5	2

Disponibilità			
1	0	2	0

Il processo C può può terminare, il vettore disponibilità diventa: 2,1,2,1
Il processo E può può terminare, il vettore disponibilità diventa: 3,2,3,1
I processi A,B,D possono terminare.
La richiesta lascia il sistema in uno stato sicuro e può essere accettata.
Il processo D resta in stato di ESECUZIONE

Esercizio 2 (4 punti)

Un sistema gestisce il processore combinando le politiche a priorità e RoundRobin con la tecnica delle code multiple (Multi-Level Feedback Queue - MFQ).
Il quanto di tempo è di *10 msec*.
La politica prevede il prerilascio, che avviene immediatamente dopo l'evento che lo provoca, senza attendere l'esaurimento del quanto di tempo corrente. Quando un processo va in esecuzione gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.
Al tempo T, nel sistema sono presenti i seguenti processi (valore maggiore di priorità comporta maggiore priorità nello scheduling):

- Processo A, con priorità 3, che al tempo T è in stato di attesa sul semaforo Sem;
- Processo B, con priorità 2, che al tempo T va in stato esecuzione (inizia il quanto di tempo);
- Processo C, con priorità 2, che al tempo T è in stato di attesa sul semaforo Sem.
- Processo D, con priorità 1, che al tempo T è in stato di pronto;

Al tempo T la coda del semaforo Sem contiene invece C->A (C è in testa).

Si chiede quale è il processo in esecuzione e la composizione delle code multiple al tempo T+30 se si verifica la seguente sequenza di eventi:

1. al tempo T+8 il processo in esecuzione esegue una *V* sul semaforo Sem;
2. al tempo T+11 il processo in esecuzione esegue una *P* sul semaforo Sem;
3. al tempo T+15 il processo in esecuzione esegue una *V* sul semaforo Sem;
4. al tempo T+20 il processo in esecuzione esegue una *P* sul semaforo Sem;
5. al tempo T+25 termina il processo in esecuzione.

Soluzione

Tempo	Evento	In esecuzione	Coda priorità 1	Coda priorità 2	Coda priorità 3	Sem: <valore,coda>
T	-	B	D	-	-	<0,C->A>

T+8	B esegue V(Sem)	B	D	C		<0,A>
T+10	Scade quanto di tempo	C	D	B	-	<0,A>
T+11	C esegue P(Sem)	B	D	-	-	<0,A,C>
T+15	B esegue V(Sem)	A	D	B	-	<0,C>
T+20	A esegue P(Sem)	B	D	-	-	<0,C,A>
T+25	Termina B	D	-	-	-	<0,C,A>
T+30	-	D	-	-	-	<0,C,A>

Soluzione

Tempo	Evento	In esecuzione	Coda priorità 1	Coda priorità 2	Coda priorità 3	Sem: <valore,coda>
T	-	B	D	-	-	<0,C->A>

Esercizio 3 (6 punti)

Un sistema operativo simile a UNIX, che gestisce la memoria con paginazione a domanda, utilizza il processo *PageDaemon,* con parametri *lotsfree= 8 e minfree=3,* e l'algoritmo di sostituzione *Second Chance.* Gli elementi della *CoreMap* hanno i campi *Proc* (processo a cui è assegnato il blocco; il campo è vuoto se il blocco è libero); *Pag* (pagina del processo caricata nel blocco), *Rif* (bit di pagina riferita utilizzato da *Second Chance*). Al tempo *t*, quando viene eseguito il *PageDaemon*, sono presenti i processi A, B, C, D. Al termine dell'esecuzione del *PageDaemon* (al tempo *t*+1) la *Core Map* ha la configurazione mostrata in figura, con il puntatore dell'algoritmo di sostituzione posizionato sul blocco 11.

											↓													
Proc	D	C	C		D		B		B	A		B	C		D	A	D		A	B		C		C
Pag	0	0	8		1		0		2	2		3	3		2	5	6		7	6		7		2
Rif	0	0	1		0		0		0	0		1	1		1	1	0		1	0		1		0
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t+1* (dopo l'esecuzione del PageDaemon)

Il *PageDaemon* interviene successivamente ogni 10 msec. Ad ogni intervento, *PageDaemon* avanza per 1 msec occupando in modo esclusivo il processore e scarica pagine fino ad arrivare a *lotsfree* pagine libere (se sono già presenti almeno *lotsfree* pagine libere non scarica niente, ma può eventualmente effettuare lo swapin di uno o più processi) o, se il numero di pagine libere è minore di *minfree*, esegue in alternativa lo *swapout* di più processi, sino ad ottenere *lotsfree* pagine libere.
La selezione dei processi candidati allo *swapout* avviene in ordine alfabetico. In caso di errori di pagina, i blocchi liberi vengono assegnati in ordine crescente di indice. Se il numero di pagine libere è sufficiente, *PageDaemon* può invece effettuare lo *swapin* di uno o più processi (scelti in ordine alfabetico) a condizione di avere sempre almeno *lotsfree* pagine libere. Lo *swapin* carica le pagine del processo che erano presenti in memoria al momento dello *swapout*.
Considerare la seguente evoluzione del sistema:

1. Dal tempo *t*+1 al tempo *t*+10 i processi riferiscono nell'ordine le pagine: B2, B4, C3, D2, D3, D5, B7, B8, C9, C2
2. Dal tempo *t*+10 al tempo *t*+11 avanza il processo *PageDaemon*;
3. Dal tempo *t*+11 al tempo *t*+20 i processi riferiscono nell'ordine le pagine C3, D0, D4, D3, C7, C2, C3, D5, C8
4. Dal tempo *t*+20 al tempo *t*+21 avanza il processo *PageDaemon.*
5. Dal tempo *t*+21 al tempo *t*+30 i processi riferiscono nell'ordine le pagine A1, A0, A3, D2, C2, C9, D7, D4, C3
6. Dal tempo *t*+30 al tempo *t*+31 avanza il processo *PageDaemon.*

Mostrare la configurazione della *CoreMap* ai tempi 10, 11, 20, 21, 30 e 31 indicando anche le pagine eventualmente scaricate o l'eventuale swapin/swapout di processi.

Soluzione

(Modificare incrementalmente la configurazione iniziale della *CoreMap*, aggiornando anche la posizione del puntatore)

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+10

Sistemi Operativi e Laboratorio, Prova del 25/5/2016

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+11 (dopo l'esecuzione del PageDaemon)

Pagine scaricate: _____swapin/out dei processi: _____

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+20

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+21 (dopo l'esecuzione del PageDaemon)

Pagine scaricate: _____swapin/out dei processi: _____

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+30

Proc																								
Pag																								
Rif																								
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+31 (dopo l'esecuzione del PageDaemon)

Pagine scaricate: _____swapin/out dei processi: _____

Soluzione

(Modificare incrementalmente la configurazione iniziale della *CoreMap*, aggiornando anche la posizione del puntatore)

↓

Proc	D	C	C	B	D	D	B	D	B	A	B	B	C	B	D	A	D	C	A	B		C		C
Pag	0	0	8	4	1	3	0	5	2	2	7	3	3	8	2	5	6	9	7	6		7		2
Rif	0	0	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	1	1	0		1		1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+10

↓

Proc	D	C	C		D	D		D					C		D		D	C				C		C
Pag	0	0	8		1	3		5					3		2		6	9				7		2
Rif	0	0	1		0	1		1					1		1		0	1				1		1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+11 (dopo l'esecuzione del PageDaemon) – rimuove A e B

↓

Proc	D	C	C	D	D	D		D					C		D		D	C				C		C
Pag	0	0	8	4	1	3		5					3		2		6	9				7		2
Rif	1	0	1	1	0	1		1					1		1		0	1				1		1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+20

↓

Proc	D	C	C	D	D	D	A	D	A	A			C		D		D	C				C		C
Pag	0	0	8	4	1	3	2	5	5	7			3		2		6	9				7		2
Rif	1	0	1	1	0	1	0	1	1	1			1		1		0	1				1		1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+21 (dopo l'esecuzione del PageDaemon) – carica A

↓

Proc	D	C	C	D	D	D	A	D	A	A	A	A	C	A	D	D	D	C				C		C
Pag	0	0	8	4	1	3	2	5	5	7	1	0	3	3	2	7	6	9				7		2
Rif	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1				1		1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+30

Sistemi Operativi e Laboratorio, Prova del 25/5/2016

								↓																
Proc	D		C	D		D		D	A	A	A	A	C	A	D	D		C				C		C
Pag	0		8	4		3		5	5	7	1	0	3	3	2	7		9				7		2
Rif	0		0	0		0		1	1	1	1	0	0	0	0	0		0				0		0
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*+31 (dopo l'esecuzione del PageDaemon)- Rimosse: D6, C0, D1, A2

Esercizio 4 (5 punti)

Si consideri un sistema di grandi dimensioni che gestisce la memoria con paginazione dinamica con le seguenti caratteristiche:

- indirizzi logici di 48 bit e ampiezza dello spazio logico di ogni processo pari a 2^{48} byte;
- pagine logiche e blocchi fisici di 8 KByte;
- tabelle delle pagine a tre livelli; la tabella di primo livello comprende 2^{11} elementi, mentre le tabelle di secondo e terzo livello hanno tutte e due la stessa dimensione;
- tutti gli elementi della tabella di primo, di secondo e di terzo livello hanno lunghezza pari a 64 bit, di cui 16 sono indicatori e i restanti bit codificano l'indice di un blocco fisico;

Si chiede:

1. La lunghezza, in numero di bit, delle componenti dell'indirizzo logico che indirizzano, rispettivamente, la tabella di primo, di secondo, e di terzo livello, e l'offset;
2. Lo spazio minimo occupato in memoria dalle sole tabelle di primo, secondo e terzo livello necessarie per tradurre un indirizzo virtuale riferito alla pagina logica 193 di un processo P presente in memoria (in byte e in blocchi fisici);
3. La massima dimensione della memoria fisica (numero di blocchi, espresso come potenza di 2);
4. Lo spazio occupato in memoria dalla tabella delle pagine nell'ipotesi in cui, nello stesso sistema, se si utilizzasse una tabella ad un solo livello (byte e numero di blocchi);
5. Quali elementi delle tabelle di primo, secondo e terzo livello sono usati per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P.

48 bit
13 offset
35 num pagina

Soluzione

1. Lunghezza del campo che indirizza la tabella di 1° livello: _____ bit;
   Lunghezza del campo che indirizza la tabella di 2° e di 3° livello: _____ bit;
   Lunghezza del campo offset: _____ bit;
2. Spazio minimo occupato in memoria dalle tabella di 1°, 2° e 3° livello per recuperare l'indirizzo fisico della pagina logica 193: _____Byte e _____blocchi fisici;
3. Massima dimensione della memoria fisica: _____ blocchi = _____Byte
4. Spazio occupato in memoria dalla tabella delle pagine (se si utilizzasse una tabella delle pagine ad un solo livello): _____ Byte = _____ blocchi;
5. L'elemento della tabella di primo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento _____;
   l'elemento della tabella di secondo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento _____;
   l'elemento della tabella di terzo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento _____

Soluzione

1. Lunghezza del campo che indirizza la tabella di 1° livello: 11 bit
   Lunghezza del campo che indirizza la tabella di 2° e di 3° livello: 12 bit
   Lunghezza del campo offset: 13 bit
2. Spazio minimo occupato in memoria dalle tabella di 1°, 2° e 3° livello per recuperare l'indirizzo fisico della pagina logica 193: $2^{11} \cdot 8$ (per la tabella di primo livello) + $2^{12} \cdot 8$ (per le tabelle di secondo e di terzo livello) = 81920 Byte e quindi 10 blocchi fisici
3. Massima dimensione della memoria fisica: 2^{48} blocchi = 2^{48+13} Bytes
4. Spazio occupato in memoria dalla tabella delle pagine (se si utilizzasse una tabella ad un solo livello): 2^{35} elementi di 6 byte = $2^{35} \cdot 6 = 192$ GB; ogni blocco fisico contiene 8KB e quindi i blocchi fisici occupati sarebbero $2^{22} \cdot 6 = 24$M blocchi
5. La pagina logica $2^{25} - 1$ del processo P è l'ultima pagina logica indirizzata dal figlio più a destra del secondo figlio della tabella di primo livello (rappresentando le tabelle come albero):
   L'elemento della tabella di primo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento 1;

l'elemento della tabella di secondo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento 4095;
l'elemento della tabella di terzo livello usato per trovare l'indirizzo fisico della pagina logica 2^{25}-1 del processo P è l'elemento 4095

Esercizio 5 (5 punti)

In un file system UNIX i blocchi del disco hanno ampiezza di 512 byte e i puntatori ai blocchi sono a 24 bit. Gli i-node contengono, oltre agli altri attributi, 5 puntatori diretti e 2 puntatori indiretti. Il primo blocco del disco ha indice logico 0.
Si consideri il file (aperto) individuato dal file descriptor *fd*, la cui lunghezza corrente è di 451.233 byte e il cui *i-node* contiene i seguenti puntatori a blocchi:

Puntatore	0	1	2	3	4	5	6
Valore del puntatore	100	101	102	120	121	122	300

dove i blocchi indiretti 122, 300 e 450 hanno i seguenti contenuti parziali:

Blocco 122:

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

Blocco 300:

Indice di elemento nel blocco	0	1	2	3	4	5	…….
Valore del puntatore	400	401	402	450	451	452	…….

Blocco 450:

Indice di elemento nel blocco	0	1	2	3	4	5	…….
Valore del puntatore	10.033	10.034	12.020	12.021	12.022	12.023	…….

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 indirizzamento indiretto semplice;
3. L'indice logico del primo blocco e dell'ultimo blocco indirizzabili con indirizzamento indiretto doppio;
4. Il numero di blocchi che compongono il file;
5. Quali sono i blocchi indiretti che vengono letti per eseguire l'operazione *read(fd,&buf,1*) quando lo I/O pointer ha valore 5.088
6. Quali sono i blocchi indiretti che vengono letti per eseguire l'operazione *read(fd,&buf,1)* quando lo I/O pointer ha valore 351.233

Soluzione

1. Il numero di puntatori che possono essere contenuti in un blocco indiretto è la parte intera inferiore di 512/3 = 170;
2. Il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto semplice hanno rispettivamente indici logici 5 e 174;
3. Il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto doppio hanno rispettivamente indici logici 175 e (175+170^2-1)- 1= 29.074;
4. L'ultimo carattere del file è contenuto nel blocco (451.233 – 1) *div* 2^9= 881; quindi il file è composto da 882 blocchi
5. Il carattere 5.088, sul quale è posizionato il puntatore di lettura, è contenuto nel blocco di indice logico *BL*=5.088 *div* 2^9= **9**.
   Poiché *BL*≥5 si utilizza l'indirizzamento **indiretto singolo**.
   Il blocco indice di primo livello utilizzato ha indice *ind*= (9- 5) div 170= 0;
   Il puntatore occupa in questo blocco la posizione *off*= (9-5) mod 170= 4;
   Poiché *ind*=0, *il* blocco indice è raggiunto tramite il puntatore indiretto singolo (numero 5) dello i-node, e occupa il blocco fisico 122.
   Il blocco dati da leggere è raggiunto tramite il puntatore *off* (quello di indice 4) del blocco indice e occupa nel disco il blocco di indice **308**.
   Quindi per eseguire l'operazione *read(fd,&buf,1*) deve essere letto il blocco indiretto di indice **122**.
6. Il carattere 351.233, sul quale è posizionato il puntatore di lettura, è contenuto nel blocco di indice logico *BL*=351.233 *div* 2^9= **686**.
   Poiché *BL*≥5 si utilizza l'indirizzamento indiretto mediante un blocco indice.
   Dato che *BL*≥175 si utilizza l'indirizzamento **indiretto doppio**, pertanto bisogna leggere il blocco indice di primo livello allocato nel blocco fisico 300.
   Il blocco indice di secondo livello utilizzato ha indice *ind*= (686- 175) div 170= 3;
   Il puntatore occupa in questo blocco la posizione *off*= (686- 175) mod 170= 1;
   Poiché *ind*=3, *il* blocco indice è raggiunto tramite il puntatore indiretto doppio (numero 3) del blocco indice, e occupa il blocco fisico 450.

Il blocco dati da leggere è raggiunto tramite il puntatore *off* (quello di indice 1) del blocco indice e occupa nel disco il blocco di indice **10.034.**

Quindi per eseguire l'operazione *read(fd,&buf,1*) devono essere letti i blocchi indiretti di indice **300 e 450**.

Soluzione

1. Numero di puntatori che possono essere contenuti in un blocco indiretto: _____
2. Il primo blocco indirizzabile con indirizzamento indiretto semplice ha indice logico: _____;
   l'ultimo blocco indirizzabile con indirizzamento indiretto semplice ha indice logico: _____
3. Il primo blocco indirizzabile con indirizzamento indiretto doppio ha indice logico: _____;
   l'ultimo blocco indirizzabile con indirizzamento indiretto doppio ha indice logico: _____
4. Il file è composto di: _____ blocchi
5. Il carattere 5.088 è contenuto nel blocco logico: _____
   Per individuare il blocco fisico che contiene il blocco logico si usa indirizzamento indiretto [singolo/doppio]: _____
   Il/i blocco/i indice da leggere sono: _____
6. Il carattere 351.233 è contenuto nel blocco logico: _____
   Per individuare il blocco fisico che contiene il blocco logico si usa indirizzamento indiretto [singolo/doppio]: _____
   Il/i blocco/i indice da leggere sono: _____

.

Esercizio 6 (3 punti)

In un disco con blocchi di 2 Kbyte, è definito un file system FAT.
Ogni elemento della FAT ha lunghezza di 4 byte e indirizza un blocco del disco.
Il primo blocco di ogni file è identificato dalla coppia (*nomefile, indiceblocco)* contenuto nella rispettiva directory.

1. Qual è la massima capacità (teorica) del disco, espressa in blocchi e in Mbyte?
2. Quanti byte occupa la FAT nel caso in cui il disco raggiunga la massima dimensione teorica?
3. Supponendo che il file *sidecar* occupi (ordinatamente) i blocchi fisici 801, 802, 798, 797 e 804, quali sono gli elementi della FAT che descrivono il file e quale è il loro contenuto?

Soluzione

1. Capacità del disco: 2^{32} blocchi → 2^{43} byte = 2^{23} Mbyte
2. Lunghezza della FAT: $2^{32} * 4$ byte → 16 GByte
3. Elementi della FAT che indirizzano il file e loro contenuto:

ELEMENTO FAT	CONTENUTO
796	…
797	804
798	797
799	…
800	…
801	802
802	798
803	…
804	Ø
805	…

Soluzione

4. Capacità del disco: _____blocchi; _____Mbyte
5. Lunghezza della FAT: _____ byte = _____ MByte
6. Elementi della FAT che indirizzano il file e loro contenuto:

ELEMENTO FAT	CONTENUTO

Sistemi Operativi e Laboratorio, Prova del 25/5/2016

Esercizio 7 (3 punti)

Un disco con 2 facce, 1000 settori per traccia e 10.000 cilindri ha un tempo di seek pari a 0,01*x msec, dove x è il valore assoluto della differenza tra l'indice del cilindro di partenza e quello del cilindro di arrivo. Il periodo di rotazione è di 20 msec: conseguentemente il tempo impiegato per percorrere un settore è di 0,02 msec.
A un certo tempo (convenzionalmente indicato come t=0) termina l'esecuzione dei comandi sul cilindro 4750 e sono pervenute, nell'ordine, le seguenti richieste di lettura o scrittura:

- clindro 9800, faccia 0, settore 555
- clindro 2100, faccia 1, settore 56
- clindro 5670, faccia 0, settore 100
- clindro 6009, faccia 1, settore 920

Inoltre:
Al tempo 90 arriva un comando di lettura sul cilindro 5999, faccia 1, settore 656
Al tempo 91 arrivano un comando di lettura sul cilindro 7400, faccia 1, settore 111

Calcolare il tempo necessario per eseguire tutte queste operazioni (tenendo conto del tempo di seek, ritardo rotazionale e tempo di percorrenza) supponendo che si adotti la politica SSTF (Shortest Seek Time First).
Il controllore è dotato di sufficiente capacità di buffering ed è sempre in grado di accettare senza ritardo i dati da leggere o da scrivere sul disco.
Per il ritardo rotazionale dopo un'operazione di *seek* si assume sempre il valore medio, pari a metà periodo di rotazione (10 msec).

Soluzione

op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:

Soluzione

op. su cilindro:	5670	settore:	100						
inizio:	0	seek:	9,2	rotazione:	10	percorrenza:	0,02	fine:	19,22
op. su cilindro:	6009	settore:	920						
inizio:	19,22	seek:	3,39	rotazione:	10	percorrenza:	0,02	fine:	32,63
op. su cilindro:	9800	settore:	555						
inizio:	32,63	seek:	37,91	rotazione:	10	percorrenza:	0,02	fine:	80,56
op. su cilindro:	2100	settore:	56						
inizio:	80,56	seek:	77	rotazione:	10	percorrenza:	0,02	fine:	167,58
op. su cilindro:	5999	settore:	656						
inizio:	167,58	seek:	38,99	rotazione:	10	percorrenza:	0,02	fine:	216,59
op. su cilindro:	7400	settore:	111						
inizio:	216,59	seek:	14,01	rotazione:	10	percorrenza:	0,02	fine:	240,62