SCHEDULING

Un sistema gestisce il processore con una politica a code multiple, con 10 classi di priorità di valori da 0 a 9. I processi pronti sono inseriti nella coda della classe corrispondente alla propria priorità; il processore viene assegnato al processo che occupa la prima posizione nella coda non vuota di massima priorità; ai processi pronti della stessa classe di priorità si applica la politica Round Robin con quanto di tempo (QdT) di 10 *msec*. Quando un processo passa in esecuzione, gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.
La politica prevede la revoca del processore, che avviene *immediatamente* quando il processo in esecuzione esaurisce il quanto di tempo o quando viene riattivato un processo con priorità maggiore di quello in esecuzione.
La priorità di ogni processo P è definita dinamicamente nel modo seguente:

- inizialmente si assegna un valore *normale*, pari a 5;
- se P ha utilizzato per intero il suo quanto di tempo, il suo valore di priorità viene decrementato di 1;
- se P era bloccato per qualsiasi motivo e viene riattivato, il suo valore di priorità viene incrementato di 1;
- se P è rimasto ininterrottamente in stato di pronto per 100 *msec* (che equivale a 10 QdT), la priorità di P viene istantaneamente e transitoriamente elevata al valore 9 (questa modifica della priorità avviene alla prima esecuzione dello schedulatore). Questo valore di priorità permane fino a quando il processo P ha ottenuto un turno di esecuzione, utilizzandoli per l'intero quanto di tempo o anche parzialmente, dopo di che la priorità torna al valore normale, pari a 5.

Al tempo T sono presenti 4 processi: A, B, C e D ed è definito il solo semaforo Sem.
Utilizzando le tabelle seguenti, si chiede di analizzare l'evoluzione dello stato dei processi A, B, C, D in base alla politica di scheduling fino al tempo T=40, a partire dalla situazione iniziale espressa in ogni tabella, nell'ipotesi che avvengano i seguenti sequenza di eventi (le eventuali scadenze del quanto di tempo non sono indicate):

- T=25: Il processo in esecuzione esegue wait(Sem)
- T=28: il processo in esecuzione termina
- T=30: Il processo in esecuzione esegue signal(Sem)

a)

Tempo	Evento	Processo A	Processo B	Processo C	Processo D	Sem
T	Scade QdT	Stato: Bloccato Priorità: 6	Stato: In esec. Priorità: 8	Stato: Pronto Priorità: 2 tempo in pronto: 70 ms	Stato: Pronto Priorità: 7 tempo in pronto: 0ms	0, A

b)

Tempo	Evento	Processo A	Processo B	Processo C	Processo D	Sem1
T	Scade QdT	Stato: Pronto Priorità: 6 tempo in pronto: 60 ms	Stato: In esec. Priorità: 8	Stato: Pronto Priorità: 2 tempo in pronto: 84 ms	Stato: Pronto Priorità: 7 tempo in pronto: 0 ms	1,-

TABELLA DELLE PAGINE

Si consideri un sistema dove gli indirizzi logici hanno la lunghezza di 24 bit e le pagine logiche e fisiche hanno ampiezza di 1 kByte. Per la gestione della memoria con paginazione dinamica si utilizzano tabelle delle pagine a 2 livelli. La tabella di primo livello comprende 2^6 elementi.
Gli elementi di ogni tabella di primo o secondo livello hanno una lunghezza di 3 byte. Di questi, 12 bit sono riservati agli indicatori (pagina caricata, riferita, modificata, ecc.) mentre i rimanenti codificano un indice di blocco fisico.
Si chiede:

1. la lunghezza del campo offset, in numero di bit;
2. il numero di elementi contenuti in ogni tabella di secondo livello;
3. lo spazio occupato in memoria da ogni tabella di secondo livello (numero di byte);
4. la massima dimensione della memoria fisica, espressa in numero di blocchi e in numero di byte.

Inoltre, supponendo che la tabella di primo livello sia caricata in memoria e che nessuna delle tabelle di secondo livello interessate sia caricata, si consideri la seguente sequenza di riferimenti a pagine:

a) pagina 8000
b) pagina 10000
c) pagina 600

si chiede quali tabelle di secondo livello devono essere caricate in memoria per portare a termine la traduzione degli indirizzi,supponendo che le informazioni per la traduzione non siano contenute nella cache della MMU

WORKING SET

Un sistema simile a Windows gestisce la memoria con paginazione a domanda mediante un *Working Set Manager*. A un certo tempo, lo stato di occupazione della memoria (senza considerare i blocchi riservati al sistema operativo) è quello descritto nella seguente Core Map, i cui elementi hanno valore nullo se il blocco è libero, o altrimenti identificano il processo e la pagina a cui il blocco è assegnato.

		B,1	C,4	C,8	C,2	A,4	B,2	A,2		B,5	A,1	A,6		B,6	B,7	A,5	C,7	B,9	
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Nel sistema sono presenti i processi A, B e C, le cui tabelle delle pagine sono le seguenti (il campo *TempoRif* registra il ***tempo virtuale del processo*** al quale è avvenuto l'ultimo riferimento alla pagina):

Pagina	Blocco	TempoRif
0		
1	21	4
2	18	7
3		
4	16	9
5	26	1
6	22	5
7		
8		
9		
Processo A		

Pagina	Blocco	TempoRif
0		
1	12	2
2	17	11
3		
4		
5	20	9
6	24	3
7	25	6
8		
9	28	5
Processo B		

Pagina	Blocco	TempoRif
0		
1		
2	15	8
3		
4	13	5
5		
6		
7	27	4
8	14	9
9		
Processo C		

A ogni processo è assegnato un *WorkingSet Ammissibile* di 4 pagine. Per ogni pagina riferita da un processo che avanza, si procede nel modo seguente:

- se la pagina è presente in memoria, si scrive il valore attuale del tempo virtuale del processo nel campo *TempoRif* della tabella delle pagine;
- se si verifica un *PageFault*, la pagina riferita viene caricata in un blocco disponibile (anche se il numero di pagine caricate, o *insieme residente*, supera la dimensione del *Working Set Ammissibile*), registrando il valore attuale del tempo virtuale del processo nel campo *TermpoRif*. I blocchi disponibili vengono assegnati in ordine crescente di indice.
- Il *Working Set Manager* viene attivato quando il numero di blocchi disponibili si riduce a 3, e si comporta nel modo seguente;
  - considera i soli processi la cui dimensione dell'*insieme residente* (numero di pagine caricate in memoria) superi quello del *WorkingSet Ammissibile*, e per questi processi ordina globalmente le pagine per valori decrescenti del parametro *TempoVirtuale- TempoRif,* dove *TempoVirtuale* è il valore attuale del tempo virtuale del processo;
  - scarica dalla memoria le pagine secondo questo ordinamento, fino a quando il numero di blocchi disponibili diventa uguale a 8. In caso di parità tra due o più pagine si considera l'ordine alfabetico dei processi.

A partire dal tempo considerato il sistema evolve nel modo seguente:

1. avanza il processo B, che ai tempi virtuali 12, 13 e 14 riferisce rispettivamente le pagine 1, 0 e 9
2. avanza il processo A, che ai tempi virtuali 11, 12, 13, 14 e 15 riferisce rispettivamente le pagine 9, 5, 6, 4 e 8.
3. quindi avanza il processo B, che ai tempi virtuali 15, 16 e 17 riferisce rispettivamente le pagine 7, 5, 8.

Si chiede la configurazione della *CoreMap* e delle tabelle delle pagine dei 3 processi al termine dei punti 1, 2, 3 e 4. Nei casi nei quali viene eseguito il *Working Set Manager* indicare anche il tempo attuale dei processi al momento dell'esecuzione del *Working Set Manager*, e quante e quali pagine vengono rimosse.

PAGE DAEMON

Un sistema operativo simile a UNIX, che gestisce la memoria con paginazione a domanda, utilizza il processo *PageDaemon,* con parametri *lotsfree= 5 e minfree=3,* e l'algoritmo di sostituzione *Second Chance* globale. 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* sono presenti i processi A, B, C, D e la *Core Map* ha la configurazione mostrata in figura, con il puntatore dell'algoritmo di sostituzione posizionato sul blocco 16.

																↓						
Proc	A	C		C	A		B	C	D		B	D	B	B	C	B	B	A	D	A	D	D
Pag	11	0		9	8		1	3	1		0	6	2	6	7	3	7	9	2	1	7	8
Rif	1	0		1	0		1	1	0		0	0	0	1	0	1	1	1	1	1	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

Core Map al tempo t

Il *PageDaemon* interviene al tempo *t* e successivamente ogni 10 msec. Ad ogni intervento, *PageDaemon* avanza per 1 msec occupando in modo esclusivo il processore e scarica pagine occupate fino a che il numero di pagine libere è pari a *lotsfree*+2 o, in alternativa, se il numero di pagine libere è minore di *minfree,* esegue lo *swap out* di alcuni processi.
Per semplicità, la selezione dei processi candidati allo *swap out* avviene in ordine alfabetico, e i processi vengono scaricati finché in memoria ci sono almeno *lotsfree*+2 blocchi liberi. Il *PageDaemon* esegue anche lo *swap in* dei processi (sempre in ordine alfabetico), dei quali vengono caricate in memoria le sole pagine riferite al momento dello *swap out*. Lo *swap in* avviene quando il numero di blocchi liberi è pari al numero di pagine del processo da caricare più *lotsfree*+2.
In caso di errori di pagina, i blocchi liberi vengono assegnati in ordine crescente di indice.
Considerare la seguente evoluzione del sistema:

1. Dal tempo *t* al tempo *t*+1 avanza il processo *PageDaemon*;
2. Dal tempo *t*+1 al tempo *t*+10 avanzano i processi A, B e C, che riferiscono nell'ordine le pagine A1, A9, A8, A5, B3, B9, C3, C0, C2
3. Dal tempo *t*+10 al tempo *t*+11 avanza il processo *PageDaemon*;
4. Dal tempo *t*+11 al tempo *t*+20 avanzano i processi B, C e D, che riferiscono nell'ordine le pagine B9, B7, B6, B4, B5, B0, D7, D2, D6, C7, C9, C8
5. Dal tempo *t*+20 al tempo *t*+21 avanza il processo *PageDaemon*.

Mostrare la configurazione della *CoreMap* ai tempi 1, 10, 11, 20 e 21

GESTIONE DEL DISCO

Un disco con 2 facce, 800 settori per traccia e 2000 cilindri ha un tempo di seek proporzionale al numero di cilindri attraversati, pari a 0,01 msec per ogni cilindro attraversato. Il periodo di rotazione è di 20 msec: conseguentemente il tempo impiegato per percorrere un settore è di 0,025 msec.
A un certo tempo (convenzionalmente indicato come t=0) termina l'esecuzione dei comandi sul cilindro 1000 e pervengono, nell'ordine, le seguenti richieste di lettura o scrittura:

- al tempo 0: clindro 670, faccia 0, settore 55
- al tempo 10: cilindro 1300, faccia 1, settore 750
- al tempo 23: clindro 32, faccia 0, settore 100
- al tempo 46: clindro 400, faccia 0, settore 88
- al tempo 47: clindro 300, faccia 0, settore 61
- al tempo 48: clindro 400, faccia 0, settore 199
- al tempo 95: clindro 320, faccia 0, settore 50
- al tempo 124: clindro 1200, faccia 0, settore 662

Calcolare il tempo necessario per eseguire tutte queste operazioni supponendo che si adotti la politica di scheduling SCAN, e che al tempo t=0 la direzione della testina sia verso l'alto.
Il tempo di esecuzione di ogni operazione è uguale alla somma dell'eventuale tempo di *seek,* del ritardo rotazionale (tempo necessario per raggiungere il settore indirizzato) e del tempo di percorrenza del settore indirizzato. Si suppone che il tempo necessario per raggiungere un qualsiasi settore dopo un'operazione di seek sia sempre uguale a un periodo di rotazione. Il controllore è dotato di sufficiente capacità di buffering ed è sempre in grado di accettare senza ritardo i dati letti dal disco o quelli da scrivere sul disco. Si assume inoltre che i comandi sullo stesso cilindro vengano eseguiti in ordine FIFO.

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

FAT

Si consideri un file system FAT-32 ospitato da un disco con *NCilindri= 512*, *NFacce= 2* e *NSettori=1024,* con settori di *1 kByte.* Nel file system sul disco, ogni blocco corrisponde ad un settore. I blocchi di indice 0 e 1 sono riservati al codice di *boot* e alla *bitmap* dei blocchi dati, e la copia permanente della FAT occupa il numero necessario di blocchi a partire da quello di indice 2. Gli elementi della FAT sono in corrispondenza uno a uno con i blocchi del disco, ma solo quelli che corrispondono a blocchi dati sono significativi, mentre non sono significativi quelli corrispondenti ai blocchi 0, 1 e a quelli occupati dalla FAT medesima. I blocchi occupati dalla FAT sono caricati a domanda nella memoria principale, le cui pagine fisiche hanno la medesima dimensione dei blocchi del disco.
Su questo File System è definito il file *terra,* che occupa 13 blocchi consecutivi a partire da quello di indice 15600 e 20 ulteriori blocchi consecutivi a partire da quello di indice 9900. Su questo file si esegue la lettura di 3.000 caratteri, quando il puntatore di lettura ha il valore 12.000.
Si chiede:

1. la capacità del disco, in blocchi e in byte;
2. la lunghezza in byte di ciascun elemento della FAT, sufficiente a indirizzare tutti i blocchi del disco;
3. il numero (intero) di blocchi occupati dalla FAT e il numero di elementi della FAT contenuti in ciascun blocco;
4. l'indice del primo blocco del disco riservato al File System come blocco dati;
5. Il numero di blocchi dati nel File System;
6. gli indici dei blocchi dati del File System ai quali si deve accedere per la lettura;
7. gli indici dei blocchi del disco ai quali si deve accedere per leggere gli elementi della FAT che contengono i puntatori ai blocchi dati da leggere, tenendo presente che l'indice del blocco 15600 è contenuto nella directory del file *terra;*
8. il massimo numero di Page Fault dovuti agli accessi alla FAT che sono necessari per l'operazione di lettura dal file *terra.*

SOLUZIONE

1. Capacità del disco: _____ blocchi → _____ byte
2. La lunghezza degli elementi della FAT, necessaria per indirizzare tutti i blocchi del disco, è di _____ bit → _____ byte;
3. La FAT occupa _____ byte → _____ _____ blocchi;

   ciascun blocco contiene _____ elementi della FAT.
4. Il primo blocco del disco disponibile come blocco dati ha indice _____
5. Il numero di blocchi dati nel File System è :_____
6. Il primo *run* è distribuito nei blocchi _____;

   il secondo *run* è distribuito nei blocchi _____..

   Il carattere 12.000 (primo carattere da leggere) occupa la posizione _____ nel blocco logico _____,

   Nel primo run quindi si leggono i blocchi: _____ ;

   Nel primo run quindi si leggono i blocchi: _____ .
7. Gli elementi della FAT ai quali si deve accedere per eseguire la lettura sono quelli di indice:

   _____;

   e i blocchi del disco che bisogna leggere per accedere a questi elementi della FAT sono:

   _____.
8. Il massimo numero di errori di pagina nell'accesso alla FAT è :_____.

FFS

In un file system UNIX i blocchi del disco hanno ampiezza di 1Kbyte e i puntatori ai blocchi sono a 16 bit. Gli i-node contengono, oltre agli altri attributi, 10 puntatori diretti e 3 puntatori indiretti. 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. Si consideri un file aperto (il cui i-node è caricato in memoria principale), la cui lunghezza corrente (in byte) è 498.600 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	900	901	942	940	971	941	972	973	974	961	800	--	--

dove il blocco indiretto 800 ha il seguente contenuto parziale:

Indice di elemento nel blocco	0	1	2	3	4	5	6	7	.....
Valore del puntatore	702	703	704	677	678	705	700	679	.....

Dire la dimensione in blocchi del file e a quali blocchi del disco bisogna accedere per leggere 2000 bytes del file a partire dal byte 13.400.

Soluzione

1. Numero di puntatori che possono essere contenuti in un blocco indiretto: _____;

2. Il primo e l'ultimo blocco indirizzabili con puntatori diretti hanno rispettivamente indici logici: _____;
3. Il primo e l'ultimo blocco indirizzabili con indirizzamento indiretto semplice hanno rispettivamente indici logici:

   _____;

4. L'ultimo carattere del file è contenuto nel blocco _____; quindi il file è composto da _____blocchi;

   Il primo carattere da leggere è il 13.400, ed è contenuto nel blocco di indice logico: _____

   L'ultimo carattere da leggere è il numero _____, che è contenuto nel blocco di indice logico: _____.

   Quindi i blocchi logici da leggere sono: _____.

   I blocchi logici da leggere sono allocati nei blocchi fisici: _____.

   I blocchi fisici da leggere sono: _____.

NTFS

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 del *run* 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= 100*, *NFacce= 2* e *NSettori= 200*. Il tempo necessario per percorrere un settore è di *0,1 msec* e il tempo di seek è proporzionale al numero di cilindri attraversati, dove il tempo di seek tra due cilindri adiacenti è pari a *1 msec*. La corrispondenza tra blocco fisico e indice di cilindro, faccia e settore è data dalle seguenti formule:

cilindro= blocco div *(NFacce*NSettori);*

faccia= (blocco mod *(NFacce*NSettori))* div *NSettori;*

settore= (blocco mod *(NFacce*NSettori))* mod *NSettori*.

In questo file system, si consideri un file che occupa 12 blocchi logici, allocati rispettivamente nei blocchi fisici: 600, 601, 602, 603, 604, 605, 21000, 21001, 21003, 1890, 1891, 1892.
Scrivere le coppie *(inizio, lunghezza)* che corrispondono ad ogni run del file:

numero run	***(inizio, lunghezza)***

Supponendo che la testina di lettura/scrittura del disco sia posizionata sul cilindro numero 88, e che dopo ogni operazione di seek la testina raggiunga il settore da leggere dopo metà del tempo rotazionale, calcolare il tempo necessario per leggere dal disco tutto il 3' run del file:

numero di blocchi da leggere:_____
indirizzo del primo blocco del run: _____
clilindro del primo blocco del run: _____
tempo di seek: _____
tempo rotazionale: _____
tempo di lettura dei blocchi: _____
tempo totale per leggere tutto il 3' run: _____