ESERCIZIO M-1 –Rilocazione statica con caricamento in partizioni fisse

In un sistema operativo che utilizza la rilocazione statica e gestisce la memoria con partizioni fisse, la memoria fisica ha un'ampiezza di 10 Mbyte ed è configurata con le seguenti partizioni:

- Partizione 1, ampiezza 1 Mbyte, riservata al Sistema Operativo;
- Partizione 2, ampiezza 0,5 Mbyte, disponibile per processi di ampiezza a con $a \leq 0{,}5$ Mbyte;
- Partizione 3, ampiezza 1 Mbyte, disponibile per processi di ampiezza a con $0{,}5 < a \leq 1$ Mbyte;
- Partizione 4, ampiezza 2,5 Mbyte, disponibile per processi di ampiezza a con $1 < a \leq 2{,}5$ Mbyte;
- Partizione 5, ampiezza 5 Mbyte, disponibile per processi di ampiezza a con $2{,}5 < a \leq 5$ Mbyte.

A partire dal tempo t=0 vengono generati i seguenti processi:

- t= 0 : Processo P1, che occupa 0,3 Mbyte con tempo di completamento di 3 sec;
- t= 0+ ε : Processo P2, che occupa 0,9 Mbyte con tempo di completamento di 2 sec;
- t= 0+ 2ε : Processo P3, che occupa 4 Mbyte con tempo di completamento di 5 sec;
- t= 0+ 3ε : Processo P4, che occupa 0,4 Mbyte con tempo di completamento di 1 sec;
- t= 0+ 4ε : Processo P5, che occupa 2 Mbyte con tempo di completamento di 4 sec;
- t= 0+ 5ε : Processo P6, che occupa 0,7 Mbyte con tempo di completamento di 3 sec;
- t= 0+ 6ε : Processo P7, che occupa 3 Mbyte con tempo di completamento di 6 sec;
- t= 0+ 7ε : Processo P8, che occupa 0,2 Mbyte con tempo di completamento di 5 sec;
- t= 0+ 8ε : Processo P9, che occupa 1,9 Mbyte con tempo di completamento di 2 sec;
- t= 0+ 9ε : Processo P10, che occupa 1,2 Mbyte con tempo di completamento di 4 sec;

Dove ε è trascurabile. Il caricamento dei processi nelle rispettive partizioni avviene con politica FIFO. Una volta caricato in memoria, ogni processo mantiene l'assegnazione fino alla sua terminazione.
Il processore viene gestito con politica FIFO. Si suppone che, una volta caricati, i processi avanzino fino ala terminazione senza mai sospendersi.
Si chiede a quale tempo avvengono il caricamento in memoria e l'uscita dal sistema di ogni processo.

SOLUZIONE

PROC	LUNGH	CARICATO al tempo t=	NELLA PARTIZIONE	IN ESECUZIONE al tempo t=	TERMINA al tempo t=
P1	0,3	0	Part 2	0	3

ESERCIZIO M-2 Rilocazione statica con caricamento in partizioni fisse, tasso di frammentazione

In un sistema operativo che utilizza la rilocazione statica e gestisce la memoria con partizioni fisse, la memoria fisica ha un'ampiezza di 10 Mbyte ed è configurata con le seguenti partizioni:

- Partizione 1, ampiezza 1 Mbyte, riservata al Sistema Operativo;
- Partizione 2, ampiezza 0,5 Mbyte, disponibile per processi di ampiezza a con $a \leq 0{,}5$ Mbyte;
- Partizione 3, ampiezza 1 Mbyte, disponibile per processi di ampiezza a con $0{,}5 < a \leq 1$ Mbyte;
- Partizione 4, ampiezza 2,5 Mbyte, disponibile per processi di ampiezza a con $1 < a \leq 2{,}5$ Mbyte;
- Partizione 5, ampiezza 5 Mbyte, disponibile per processi di ampiezza a con $2{,}5 < a \leq 5$ Mbyte.

A un certo sono caricati in memoria i seguenti processi:

- Processo P1, che occupa 0,3 Mbyte;
- Processo P2, che occupa 0,9 Mbyte
- Processo P3, che occupa 2 Mbyte:
- Processo P4, che occupa 3 Mbyte

Si chiede di valutare il tasso di frammentazione della memoria, definito come rapporto tra la somma delle lunghezze dei frammenti inutilizzati e l'ampiezza complessiva della memoria riservata ai processi, e di indicare inizio e fine di ciascun frammento.

SOLUZIONE

(Inizio e fine riferiti a inizio partizione)

- Frammento Part 2 inizio ….. fine …..
- Frammento Part 3 inizio ….. fine …..
- Frammento Part 4 inizio ….. fine …..
- Frammento Part 5 inizio ….. fine …..

Lunghezza complessiva dei frammenti (Mbyte) ……..

Tasso di frammentazione della memoria : ……..

ESERCIZIO M-3 Rilocazione dinamica e caricamento con partizioni variabili

In un sistema che gestisce la memoria con rilocazione dinamica e caricamento in partizioni variabili, il sistema operativo occupa una partizione con origine 0 e lunghezza 7. Inoltre prima del tempo t=10 sono stati caricati in memoria i seguenti processi:

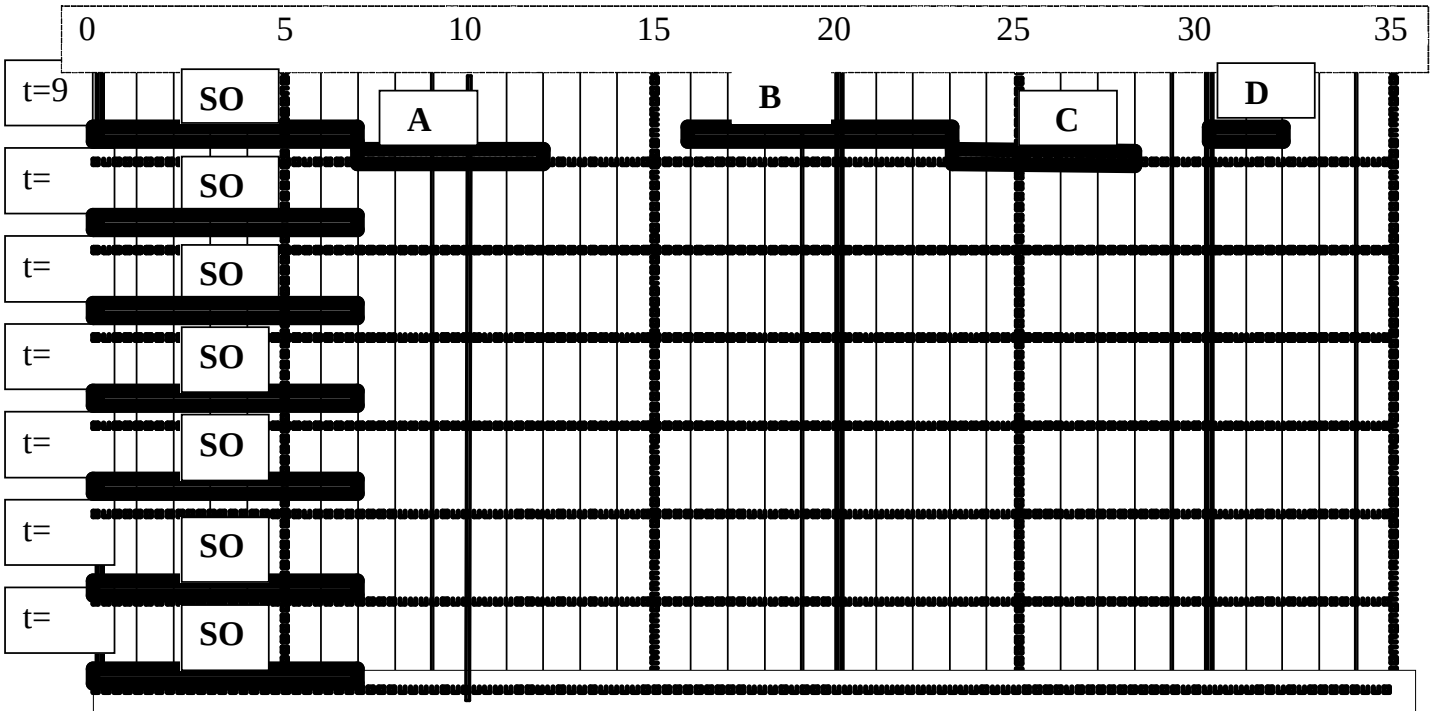
- Al tempo 1 il processo C, che occupa una partizione con origine 23 e lunghezza 5;
- Al tempo 5 il processo B, che occupa una partizione con origine 16 e lunghezza 7;
- Al tempo 7 il processo A, che occupa una partizione con origine 7 e lunghezza 5;
- Al tempo 9 il processo D, che occupa una partizione con origine 30 e lunghezza 2.

Prima del tempo 10 sono generati nell'ordine anche i seguenti processi, che rimangono in attesa di caricamento:

- il processo E che richiede una partizione di lunghezza 10;
- il processo F che richiede una partizione di lunghezza 5;
- il processo G che richiede una partizione di lunghezza 4.

I processi in attesa di caricamento sono inseriti in una coda *FIFO*. Tutti i tempi sono espressi in *msec* e tutte le lunghezze delle partizioni sono espresse in *Mbyte*.

A ogni processo caricato in memoria viene revocata l'assegnazione dopo che sono trascorsi 10 *msec* dal caricamento. Quando un processo viene scaricato è inserito nell'ultima posizione della coda *FIFO* e il sistema operativo carica (se possibile) uno o piu' dei processi in attesa di caricamento, adottando la politica FIFO per la scelta dei processi da caricare e la politica *First-Fit* per l'assegnazione delle partizioni. Utilizzando il grafico sotto riportato, mostrare come evolvono l'occupazione della memoria e la coda FIFO dei processi in attesa di caricamento a seguito dei primi 6 interventi del sistema operativo, a partire da quello che avviene al tempo 11. Durante questo periodo nessuno dei processi termina.



t= 9: coda E -> F -> G;

t= …: scaricato …
caricato --
coda

t= …: scaricato …
caricati …
coda …..

t= … : scaricato …
caricato …
coda

t= …: scaricato;
caricato
coda

t= …: scaricato …
caricato --
coda

t= …: scaricato …
caricati …
coda …..

t= … : scaricato …
caricato …
coda

ESERCIZIO M-4 Rilocazione dinamica e caricamento in partizioni variabili

In un sistema che gestisce la memoria con rilocazione dinamica e caricamento in partizioni variabili, il sistema operativo occupa una partizione con origine 0 e lunghezza 7. Prima del tempo 0 sono stati generati e caricati in memoria i seguenti processi

- il processo C, che risiede in memoria da 7 *msec* ed occupa una partizione con origine 23 e lunghezza 5;
- il processo A, che risiede in memoria da 6 *msec* ed occupa una partizione con origine 7 e lunghezza 5;
- il processo D, che risiede in memoria da 5 *msec* ed occupa una partizione con origine 30 e lunghezza 2.
- il processo B, che risiede in memoria da 2 *msec* ed occupa una partizione con origine 16 e lunghezza 7;

Inoltre ai tempi 0, 1 e 2 sono generati rispettivamente i seguenti processi, che fino al tempo 3 rimangono in attesa di caricamento:

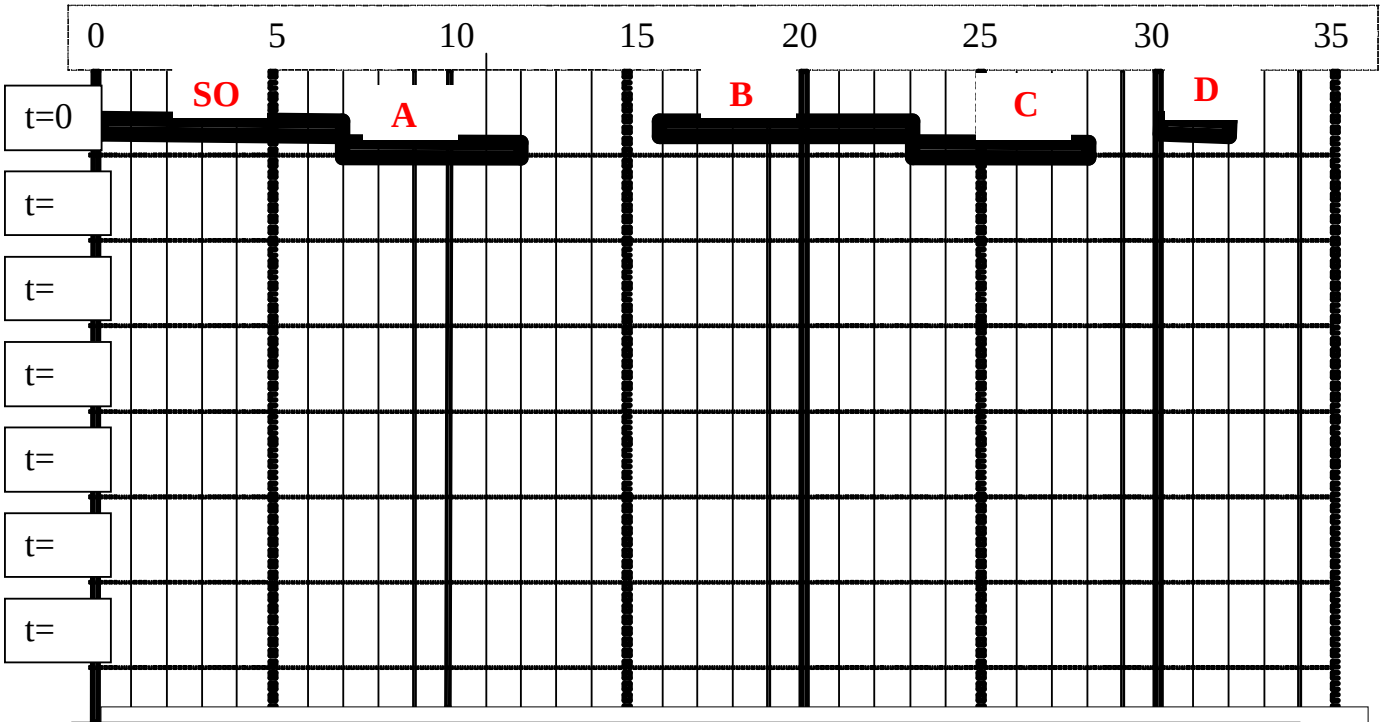
- il processo E che richiede una partizione di lunghezza 10;
- il processo F che richiede una partizione di lunghezza 5;
- il processo G che richiede una partizione di lunghezza 6.

Il codice dei processi è rilocabile. I processi in attesa di caricamento sono inseriti in una coda FIFO. Tutti i tempi sono espressi in *msec* e tutte le lunghezze delle partizioni sono espresse in *Mbyte*.

A ogni processo caricato in memoria viene revocata l'assegnazione quando sono trascorsi 10 *msec* dal caricamento. In questa circostanza interviene lo scheduler, che inserisce il processo scaricato nell'ultima posizione della coda dei processi in attesa di caricamento e, finchè il primo processo di questa coda trova una partizione libera di ampiezza sufficiente, lo rimuove dalla coda e lo carica in memoria. Se nessuno dei processi in coda può essere caricato con questa politica, si esegue preliminarmente il compattamento della memoria (verso il basso) e si ripete il tentativo di caricamento La politica per l'assegnazione delle partizioni è la *First Fit.*

Utilizzando il grafico sotto riportato, mostrare come evolvono l'occupazione della memoria e la coda dei processi in attesa di caricamento a seguito dei primi 6 interventi del sistema operativo, a partire da quello che avviene al tempo 3.

Si suppone che nessuno dei processi terminii in questo intervallo di tempo.



t= 2 coda E,F ,G;	t= : scaricati compatt? caricati : coda:	t=: scaricati: compatt? caricati : coda	t= scaricati: compatt? caricati : coda	t= scaricati: compatt? caricati : coda
t= : scaricati compatt? caricati : coda:	t=: scaricati: compatt? caricati : coda	t= scaricati: compatt? caricati : coda	t= scaricati: compatt? caricati : coda	

ESERCIZIO M-5 Rilocazione dinamica e caricamento in partizioni variabili

In un sistema operativo che gestisce la memoria con rilocazione dinamica e caricamento in partizioni variabili, la memoria fisica ha un'ampiezza di 35 Mbyte. La partizione 1, della lunghezza di 7 Mbyte, è riservata al sistema operativo, mentre il resto della memoria è disponibile per il caricamento dei processi.
Al tempo *t* sono caricati in memoria i seguenti processi:

- Processo A, nella partizione con origine 7 Mbyte e lunghezza 5 Mbyte;
- Processo B, nella partizione con origine 16 Mbyte e lunghezza 7 Mbyte;
- Processo C, nella partizione con origine 23 Mbyte e lunghezza 5 Mbyte;
- Processo D, nella partizione con origine 30 Mbyte e lunghezza 2 Mbyte;

Successivamente sii verificano i seguenti eventi (in alternativa):

1. Al tempo *t+2* termina il processo B; quindi al tempo *t+ 5* viene generato il processo E il cui spazio virtuale occupa 10 Mbyte;
2. Al tempo *t+2* termina il processo C; quindi al tempo *t+ 5* viene generato il processo F il cui spazio virtuale occupa 5 Mbyte;
3. Al tempo *t+2* viene generato il processo G il cui spazio virtuale occupa 4 Mbyte; quindi al tempo *t+5* termina il processo A.

Per l'assegnazione delle partizioni si utilizza la politica *first fit.*
Per individuare le partizioni libere si utilizza una lista, i cui elementi sono coppie del tipo *(Lunghezza della partizione, Origine della partizione successiva)*
Si chiede, nelle tre ipotesi, la configurazione della lista delle partizioni libere al tempo *t*, al tempo *t+2* e al tempo *t+5*,

SOLUZIONE

Al tempo *t*:

- Partizioni libere: (0rigine 12, lungh, 4); (origine 28, lungh 2), (origine 32, lungh 3)
- Lista (*Ø, 12)* → *(4, 28)* → *(2, 32)* → *(3, Ø)*

Ipotesi 1

Al tempo *t+2:*

- Partizioni libere:
  Lista:
  Al tempo *t+5:*
- Partizioni libere:
  Lista:

Ipotesi 2

Al tempo *t+2:*

- Partizioni libere:
  Lista:
  Al tempo *t+5:*
- Partizioni libere:
  Lista:

Ipotesi 3

Al tempo *t+2:*

- Partizioni libere:
  Lista:
  Al tempo *t+5:*
- Partizioni libere:
  Lista:

ESERCIZIO M-6 Rilocazione dinamica e caricamento in partizioni variabili

In un sistema operativo che gestisce la memoria con rilocazione dinamica e caricamento in partizioni variabili, la memoria fisica ha un'ampiezza di 35 Mbyte. La partizione 1, della lunghezza di 7 Mbyte, è riservata al sistema operativo, mentre il resto della memoria è disponibile per il caricamento dei processi.
Al tempo *t* sono caricati in memoria i seguenti processi:

- Processo A, nella partizione con origine 7 Mbyte e lunghezza 5 Mbyte;
- Processo B, nella partizione con origine 16 Mbyte e lunghezza 7 Mbyte;
- Processo C, nella partizione con origine 23 Mbyte e lunghezza 5 Mbyte;
- Processo D, nella partizione con origine 30 Mbyte e lunghezza 2 Mbyte;

Successivamente si verificano i seguenti eventi (in alternativa):

1. Al tempo *t+2* termina il processo B; quindi al tempo *t+ 5* viene generato il processo E il cui spazio virtuale occupa 10 Mbyte;
2. Al tempo *t+2* termina il processo C; quindi al tempo *t+ 5* viene generato il processo F il cui spazio virtuale occupa 5 Mbyte;
3. Al tempo *t+2* viene generato il processo G il cui spazio virtuale occupa 4 Mbyte; quindi al tempo *t+5* termina il processo A.

Per l'assegnazione delle partizioni si utilizza la politica *first fit.*
Nelle 3 ipotesi, si chiede di valutare al tempo *t+5* il tasso di frammentazione della memoria, definito come rapporto tra la somma delle lunghezze dei frammenti inutilizzati e l'ampiezza complessiva della memoria riservata ai processi.

SOLUZIONE

Alternativa 1)
Al tempo *t+2* termina il processo B; quindi al tempo *t+ 5* viene generato il processo E il cui spazio virtuale occupa 10 Mbyte;

Al tempo *t:*

SO	SO	SO	SO	SO	SO	SO	A	A	A	A	A					B	B	B	B	B	B	B	C	C	C	C	C			D	D			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+2:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+5:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

- Frammento 1:
- Frammento 2:
- Frammento 3:

Lunghezza complessiva dei frammenti (Mbyte): ……

Tasso di frammentazione della memoria (%):……

ESERCIZIO M-6, Alternativa 2)

Al tempo *t+2* termina il processo C; quindi al tempo *t+ 5* viene generato il processo F il cui spazio virtuale occupa 5 Mbyte;

Al tempo *t:*

SO	SO	SO	SO	SO	SO	SO	A	A	A	A	A					B	B	B	B	B	B	B	C	C	C	C	C			D	D			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+2:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+5:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

- Frammento 1:
- Frammento 2:
- Frammento 3:

Lunghezza complessiva dei frammenti (Mbyte): ……

Tasso di frammentazione della memoria (%):……

ESERCIZIO M-6 , Alternativa 3)

Al tempo *t+2* viene generato il processo G il cui spazio virtuale occupa 4 Mbyte; quindi al tempo *t+5* termina il processo A.

Al tempo *t:*

SO	SO	SO	SO	SO	SO	SO	A	A	A	A	A					B	B	B	B	B	B	B	C	C	C	C	C			D	D			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+2:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Al tempo *t+5:*

SO	SO	SO	SO	SO	SO	SO																												
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

- Frammento 1:
- Frammento 2:
- Frammento 3:

Lunghezza complessiva dei frammenti (Mbyte): ……

Tasso di frammentazione della memoria (%):……

ESERCIZIO M-7 Segmentazione

In un sistema UNIX, la memoria virtuale di ogni processo comprende i segmenti *codice*, *dati* e *pila*. Il caricamento nella memoria fisica avviene con la tecnica delle partizioni variabili.
Il processo ProcK, che occupa le partizioni di origine 21300, 40500 e 62000 e lunghezze 10000, 7000 e 2000 rispettivamente per il codice, i dati e la pila, esegue la chiamata di sistema *fork* che genera il processo ProcJ. I dati e la pila del processo ProcJ sono caricati rispettivamente a partire dagli indirizzi 70000 e 77000.
Si chiedono i contenuti dei registri base e limite dei segmenti codice, dati e pila quando è in esecuzione il processo ProcK e quando è in esecuzione il processo ProcJ.

SOLUZIONE

1) Quando è in esecuzione il processo ProcK:

Segmento codice: registro base ……. registro limite …….

Segmento dati: registro base ……. registro limite …….

Segmento pila: registro base ……. registro limite …….

2) Quando è in esecuzione il processo ProcJ:

Segmento codice: registro base ……. registro limite …….

Segmento dati: registro base ……. registro limite …….

Segmento pila: registro base ……. registro limite …….

ESERCIZIO M-8 Segmentazione

In un sistema UNIX, la memoria virtuale di ogni processo comprende i segmenti *codice*, *dati* e *pila*. Il caricamento nella memoria fisica avviene con la tecnica delle partizioni variabili.
Il processo ProcK, che occupa le partizioni di origine 21300, 40500 e 62000 e lunghezze 10000, 7000 e 2000 rispettivamente per il codice, i dati e la pila, esegue la chiamata di sistema *fork* che genera il processo ProcJ. Al momento della chiamata sono libere 3 partizioni, che iniziano rispettivamente agli indirizzi 77000, 90000 e 400000 e hanno rispettivamente lunghezze 12000, 10000. e 20000, L'assegnazione delle partizioni avviene con politica *first fit.*
Si chiede:

1. l'origine e la lunghezza delle partizioni che rimangono libere dopo l'esecuzione della *fork ;*
2. i contenuti dei registri base e limite dei segmenti codice, dati e pila quando è in esecuzione il processo ProcK
3. i contenuti dei registri base e limite dei segmenti codice, dati e pila quando è in esecuzione il processo ProcJ.

SOLUZIONE

1) Partizioni libere dopo l'esecuzione della fork:

a) origine ……. lunghezza …….
b) origine ……. lunghezza …….
c) origine ……. lunghezza …….

2) Quando è in esecuzione il processo ProcK:

Segmento codice: registro base ……. registro limite …….
Segmento dati: registro base ……. registro limite …….
Segmento pila: registro base ……. registro limite …….

2) Quando è in esecuzione il processo ProcJ:

Segmento codice: registro base ……. registro limite …….
Segmento dati: registro base ……. registro limite …….
Segmento pila: registro base ……. registro limite …….

ESERCIZIO M-9 Segmentazione

In un sistema che gestisce la memoria con spazio logico suddiviso in segmento codice e segmento dati (comprensivo della pila) e caricamento in partizioni variabili con rilocazione dinamica, i registri base e limite del segmento codice hanno a un certo istante i valori B_1= 12.000 e L_1= 28.000, mentre i registri base e limite del segmento dati hanno i valori B_2= 45.000 e L_2= 25.000.

Supponendo che il processo in esecuzione estragga istruzioni dagli indirizzi logici 7.000, 30.000 e 25.000, che riferiscono dati con indirizzi logici 47.000, 40.000 e 20.000, dire se ognuno dei precedenti indirizzi logici è legittimo e, in caso affermativo, calcolare il corrispondente indirizzo fisico.

SOLUZIONE

Istruzioni:

Indirizzo logico	Legittimo?	Indirizzo fisico
1. …….	…….	…….
2. …….	…….	…….
3. …….	…….	…….

Dati:

1. …….	…….	…….
2. …….	…….	…….
3. …….	…….	…….

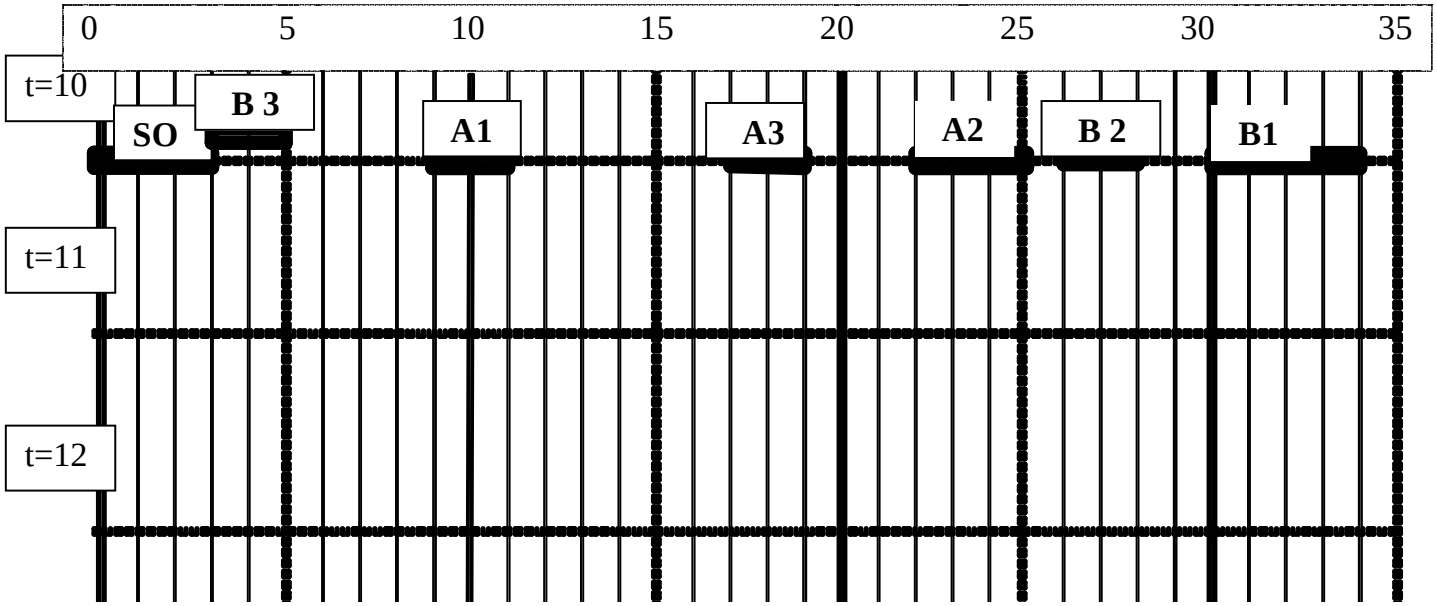
ESERCIZIO M-10 – Segmentazione in UNIX

In un sistema simile a Unix che gestisce la memoria con segmentazione e caricamento in partizioni variabili, il sistema operativo occupa una partizione con origine 0 e lunghezza 3. Inoltre al tempo t=10 sono stati caricati in memoria i seguenti processi:

- il processo A, che occupa tre partizioni:
  - una partizione A1 con origine 9 e lunghezza 2 per il codice
  - una partizione A2 con origine 22 e lunghezza 3 per i dati
  - una partizione A3 con origine 17 e lunghezza 2 per lo stack
- il processo B, che occupa tre partizioni:
  - una partizione B1 con origine 30 e lunghezza 4 per il codice
  - una partizione B2 con origine 26 e lunghezza 2 per i dati
  - una partizione B3 con origine 3 e lunghezza 2 per lo stack

Tutte le lunghezze delle partizioni sono espresse in *Mbyte* e che la memoria ha dimensione 35, inoltre il gestore della memoria adotta politica *best fit* per l'assegnazione delle partizioni.
Al tempo t=11 il processo A esegue una `fork` e viene creato il processo C. Successivamente al tempo 12 il processo C esegue una `exec` per eseguire un nuovo codice. Si supponga che il nuovo codice occupi una partizione C1 di lunghezza 3,e che utilizzi inizialmente una partizione C2 di lunghezza 4 per i dati e una partizione C3 di lunghezza 1 per lo stack.
Utilizzando il grafico sotto riportato, mostrare come evolve l'occupazione della memoria ai tempi 11 e 12.

SOLUZIONE



ESERCIZIO M-11 LRU globale

In un sistema che gestisce la memoria con paginazione, sono presenti i processi A, B e C. Lo stato di occupazione della memoria al tempo 12 è descritto dalla seguente *CoreMap*, dove per ogni blocco si specifica nell'ordine: il processo a cui appartiene la pagina caricata, l'indice della pagina e il tempo al quale è avvenuto l'ultimo riferimento alla pagina stessa . Ad esempio, nel blocco 11 è caricata la pagina 6 del processo B, il cui ultimo riferimento è avvenuto al tempo 5.

SO	SO	SO	SO	SO	SO	B,3 11	A,1 2	B,0 10	C,1 3		B,6 5	C,7 8	A,4 12	C,3 6	A,5 9	C,5 7	B,2 1	A,7 4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Per la gestione della memoria si utilizza un algoritmo di sostituzione LRU globale.
Quali pagine vengono scaricate dalla memoria e caricate in memoria ai tempi t= 13, t=14 e t=15 nelle seguenti ipotesi (alternative):

(a) Il processo A riferisce la pagina 2 al tempo 13, la pagina 6 al tempo 14 e la pagina 4 al tempo 15;
(b) Il processo A riferisce la pagina 5 al tempo 13, la pagina 3 al tempo 14 e la pagina 8 al tempo 15;
(c) Il processo B riferisce la pagina 2 al tempo 13, la pagina 7 al tempo 14 e la pagina 6 al tempo 15;
(d) Il processo B riferisce la pagina 1 al tempo 13, la pagina 7 al tempo 14 e la pagina 0 al tempo 15;
(e) Il processo C riferisce la pagina 0 al tempo 13, la pagina 6 al tempo 14 e la pagina 4 al tempo 15;
(f) Il processo C riferisce la pagina 3 al tempo 13, la pagina 4 al tempo 14 e la pagina 0 al tempo 15.

SOLUZIONE

	t=	Pagina scaricata	Pagina caricata ...	Nel blocco
(a)	13	-	A2	10
	14			
	15			
(b)	13			
	14			
	15			
(c)	13			
	14			
	15			
(d)	13			
	14			
	15			
(e)	13			
	14			
	15			
(f)	13			
	14			
	15			

ESERCIZIO M-12 LRU locale

In un sistema che gestisce la memoria con paginazione, sono presenti i processi A, B e C. Lo stato di occupazione della memoria al tempo 11 è descritto dalla seguente tabella, dove per ogni blocco si specifica nell'ordine: il processo a cui appartiene la pagina caricata, l'indice della pagina e il tempo al quale è avvenuto l'ultimo riferimento alla pagina stessa . Ad esempio, nel blocco 11 è caricata la pagina 6 del processo B, il cui ultimo riferimento è avvenuto al tempo 5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SO	SO	SO	SO	SO	SO		A,1 2	B,0 10	C,1 3		B,6 5	C,7 8		C,3 6	A,5 9	C,5 7	B,2 1	A,7 4

Le tabelle delle pegine dei tre processi sono le seguenti (con notazione evidente; in parentesi è replicato il tempo al quale è avvenuto l'ultimo riferimento)

Pagina	Blocco
0	-
1	7 (2)
2	-
3	-
4	-
5	15 (9)
6	-
7	18 (4)

Processo A

Pagina	Blocco
0	8 (10)
1	-
2	17 (1)
3	-
4	-
5	-
6	11 (5)
7	-

Processo B

Pagina	Blocco
0	-
1	9 (3)
2	-
3	14 (6)
4	-
5	16 (7)
6	-
7	12 (8)

Processo C

Per la gestione della memoria si utilizza un algoritmo di sostituzione LRU locale. Il working set assegnato al processo A (inteso come numero di blocchi a disposizione del processo, anche se momentaneamente non occupati) ha dimensione 3; analogamente quelli assegnati ai processi B e C hanno dimensione 5.
Quali pagine vengono scaricate dalla memoria e caricate in memoria al tempo t= 12 e al tempo 13 nelel seguenti ipotesi (alternative):

(g) Il processo A riferisce la pagina 2 al tempo 12 e la pagina 6 al tempo 13;
(h) Il processo A riferisce la pagina 5 al tempo 12 e la pagina 3 al tempo 13;
(i) Il processo B riferisce la pagina 2 al tempo 12 e la pagina 7 al tempo 13;
(j) Il processo B riferisce la pagina 1 al tempo 12 e la pagina 7 al tempo 13;
(k) Il processo C riferisce la pagina 0 al tempo 12 e la pagina 6 al tempo 13;
(l) Il processo C riferisce la pagina 3 al tempo 12 e la pagina 4 al tempo 13;

SOLUZIONE

(a) t=12: scaricata … caricata…. nel blocco … t=13: scaricata … caricata…. nel blocco

(b) t=12: scaricata … caricata…. nel blocco t=13: scaricata … caricata…. nel blocco

(c) t=12: scaricata … caricata…. nel blocco t=13: scaricata … caricata…. nel blocco

(d) t=12: scaricata … caricata…. nel blocco t=13: scaricata … caricata…. nel blocco

(e) t=12: scaricata … caricata…. nel blocco t=13: scaricata … caricata…. nel blocco

(f) t=12: scaricata … caricata…. nel blocco t=13: scaricata … caricata…. nel blocco

ESERCIZIO M-13 Algoritmo Second Chance

In un sistema che gestisce la memoria con paginazione, sono presenti i processi A, B e C. Lo stato di occupazione della memoria è descritto dalla *Core Map*, dove per ogni blocco si specifica nell'ordine: il processo a cui appartiene la pagina caricata, l'indice della pagina caricata e il bit di pagina riferita.
Al tempo *t* la configurazione della *CoreMap* è quella riportata in figura, dove, per esempio, nel blocco 11 è caricata la pagina 7 del processo C con bit di pagina riferita uguale a 1.
L'algoritmo di sostituzione è il *Second Chance* (globale) e prima di ogni invocazione il puntatore è posizionato sul blocco successivo alla *vittima* individuata nell'invocazione precedente.

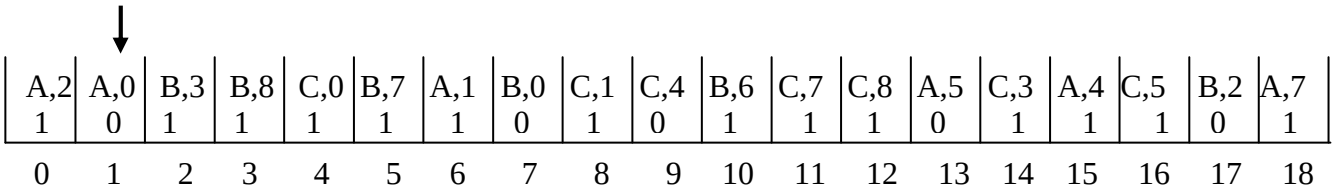
A,2 1	A,0 0	B,3 1	B,8 1	C,0 1	B,7 1	A,1 1	B,0 0	C,1 1	C,4 0	B,6 1	C,7 1	C,8 1	A,5 0	C,3 1	A,4 1	C,5 1	B,2 0	A,7 1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Mostrare come si modifica la *CoreMap* se al tempo *t* si verifica (in alternativa) uno dei seguenti eventi:

1. Il puntatore è posizionato sul blocco 1 e il processo A riferisce la pagina 5
2. Il puntatore è posizionato sul blocco 2 e il processo C riferisce la pagina 2
3. Il puntatore è posizionato sul blocco 7 e il processo B riferisce la pagina 5
4. Il puntatore è posizionato sul blocco 18 e il processo A riferisce la pagina 3.

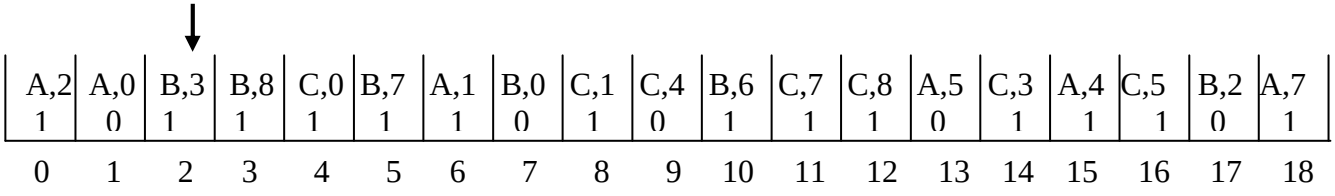
SOLUZIONE

1) Il puntatore è posizionato sul blocco 1 e il processo A riferisce la pagina 5
Configurazione iniziale:

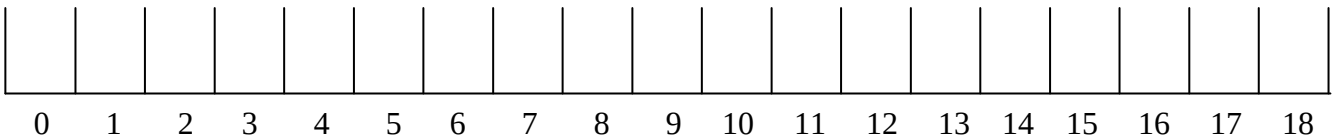


Configurazione finale:

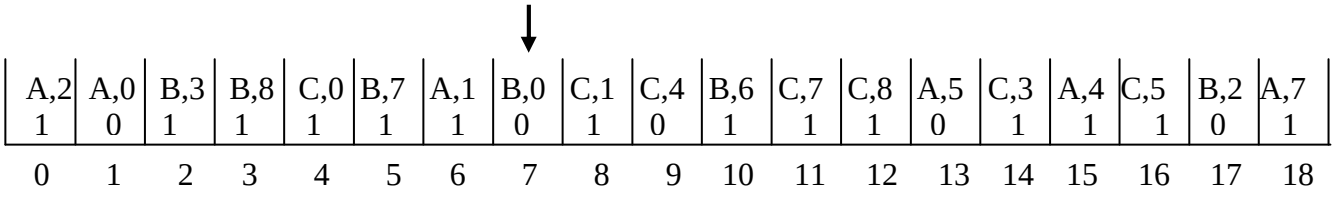
2) Il puntatore è posizionato sul blocco 2 e il processo C riferisce la pagina 2
Configurazione iniziale:



Configurazione finale:

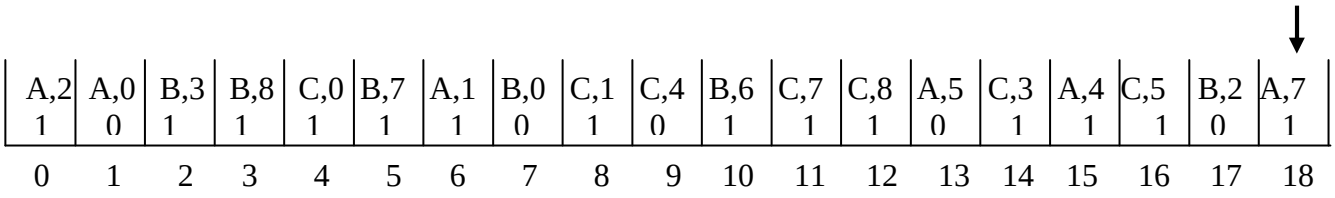


3) Il puntatore è posizionato sul blocco 2 e il processo C riferisce la pagina 2
Configurazione iniziale:



Configurazione finale:

4) Il puntatore è posizionato sul blocco 18 e il processo A riferisce la pagina 3.
Configurazione iniziale:



Configurazione finale:

ESERCIZIO M-14 – Tabelle delle pagine

Si consideri un sistema dove gli indirizzi logici hanno la lunghezza di 32 bit e le pagine logiche e fisiche hanno ampiezza di 4 kByte.Per la gestione della memoria con paginazione dinamica si utilizzano tabelle delle pagine a 2 livelli. La tabella di primo livello comprende 2^{10} elementi.
Gli elementi di ogni tabella di primo o secondo livello occupano 4 byte, di cui 1 è riservato agli indicatori (pagina caricata, riferita, modificata, ecc.) mentre i rimanenti individuano un indice di blocco fisico.
Si chiede:

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

SOLUZIONE

1. lunghezza del campo offset : ……
2. lunghezza (numero di elementi) di ogni tabella di secondo livello: ……
3. spazio occupato in memoria da ogni tabella di secondo livello (numero di byte) : ……
4. massima dimensione della memoria fisica : ……

ESERCIZIO M- 15 Tabella delle pagine

Un sistema gestisce la memoria con paginazione e usa indirizzi logici di 32 bit, blocchi di memoria di 1 kbyte e tabelle delle pagine a due livelli. La tabella delle pagine di 1° livello (o *directory)* e quelle di 2° livello hanno uguale dimensione. Ogni elemento della tabella di primo livello e di quelle di secondo livello contiene l'indice di un blocco di memoria e 2 bit di controllo.
La memoria fisica ha una capacità di 4 Gbyte (= 2^{32} byte)
Domande:

1. Nell'indirizzo logico, quanti bit sono riservati all'offset nella tabella di 1° livello, quanti all'offset nella tabella di 2° livello e quanti all'offset nella pagina logica?
2. Quanti bit sono necessari per codificare un indice di blocco e quanti bit contiene ogni elemento dei tabella di primo o di secondo livello?
3. Quanti byte e quanti blocchi di memoria occupa ogni tabella?

SOLUZIONE

Il numero di blocchi fisici è ……

1. Offset in Tab 1° livello (numero di bit): ……

   Offset in Tab 2° livello (numero di bit): ……

   Offset in pagina logica (numero di bit): ……

   Bit necessari per codificare un indice di blocco: ……

   Lunghezza in bit di ogni elemento di tabella: ……

2. Numero di byte occupati da ogni tabella:

   Numero di blocchi occupati da ogni tabella: ……

ESERCIZIO M-16 -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=2*, 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* 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 11. I primi 6 blocchi della memoria fisica sono riservati al sistema operativo e sono ignorati dall'algoritmo di sostituzione.

Proc							C	A	B	A		B	C	D	D		D	B	A	B		C		C
Pag							0	1	0	2		6	3	1	2		6	2	7	3		7		2
Rif							0	1	0	0		1	1	0	1		0	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*

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 fino a 3 pagine o, in alternativa, esegue lo *swapout* di un processo.
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.
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 B e C, che riferiscono nell'ordine le pagine B0, B1, B6, C2, C0, C4, B2, B5
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 D e B, che riferiscono nell'ordine le pagine D3, D6, D2, D1, B0, B2, B3, B0;
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

SOLUZIONE

(Modificare 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+1*

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*

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*

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*

ESERCIZIO M-17 – Working Set Manager

Si consideri un sistema che gestisce la memoria con paginazione a domanda, applicando un algoritmo di sostituzione LRU localee una politica di controllo dinamico del Working Set. Per ogni processo sono definiti i seguenti dati:

- l'intero *MaxBlocchi:* massimo numero di blocchi disponibili per il caricamento del *Working Set*, che viene ridefinito periodicamente dal demone *WorkingSetManager;*
- l'intero *PagineResidenti*, uguale al numero di pagine attualmente caricate in memoria e variabile nel tempo;
- la tabella delle pagine di ogni processo, con campi *Pagina, Blocco, R (bit di pagina riferita) e DP (Distanza Passata).*

Quando un processo avanza, per ogni pagina riferita:

- se il riferimento determina *Page Fault*, la pagina riferita viene caricata nel blocco libero individuato dal primo elemento della lista *BlocchiDisponibili* (che su suppone sempre non vuota), viene incrementata la variabile *PagineResidenti* e viene definito *DP= 0;*
- in ogni caso, si assegna *R= 1.*

Il processo *WorkingSetManager*, che interviene periodicamente, esegue le seguenti operazioni per ogni processo:
Fase 1) per ogni pagina residente in memoria aggiorna la distanza passata con il seguente algoritmo:

- se *R= 0* assegna *DP= DP+ 1;*
- altrimenti assegna *R= 0* e *DP= 0*

Fase 2) se *PagineResidenti> MaxBlocchi:*

- scarica dalla memoria principale *PagineResidenti- MaxBlocchi*, selezionandole in ordine decrescente del valore di *DP;*
- se per l'ultima pagina scaricata si ha *DP< 7*, assegna *MaxBlocchi= MaxBlocchi + 1;*

se *PagineResidenti< MaxBlocchi*, assegna *MaxBlocchi= MaxBlocchi – 1.*

Al tempo T1, subito dopo un intervento di *WorkingSetManager*, per il processo *P* si ha ***MaxBlocchi* = 7** e la tabella delle pagine ha la seguente configurazione:

Pagina	Blocco	R	DP
0	-	-	-
1	6	0	2
2	-	-	-
3	-	-	-
4	-	-	-
5	8	0	5
6	-	-	-
7	10	0	4
8	15	0	8
9	-	-	-
10	20	0	9
11	-	-	-
12	30	0	7

Dopo *T1* e prima del tempo *T2* il processo *P* avanza e riferisce nell'ordine le seguenti pagine: ***5, 3, 0, 5, 0, 9, 3, 8, 9, 11***
Al tempo *T2* entra in esecuzione *WorkingSetManager*, che applica la politica sopra definita.
Al tempo *T4* termina l'intervento di *WorkingSetManager*
Supponendo che il contenuto della lista *BlocchiDisponibili* al tempo T1 sia → 50 → 51 → 52 → 53 → 54 → …., si chiede:

a) il valore di *PagineResidenti* e la configurazione della Tabella delle Pagine del processo P al tempo *T2*;
b) la configurazione della Tabella delle Pagine del processo *P* al tempo *T3*, quando termina la Fase 1 di *WorkingSetManager*;
c) il valore di *PagineResidenti* e di *MaxBlocchi* e la configurazione della Tabella delle Pagine del processo *P* al tempo *T4*, quando termina la Fase 2 di *WorkingSetManager*.

SOLUZIONE

Tempo T2: Tabella delle Pagine

Pagina	Blocco	R	DP
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Pagine Residenti= ……..

Tempo T3: Tabella delle Pagine

Pagina	Blocco	R	DP
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Tempo T4: Tabella delle Pagine

Pagina	Blocco	R	DP
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Pagine Residenti= ……..;
MaxBlocchi= ……