esercizio n. 2 – memoria virtuale

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF: *3 bit* spiazzamento fisico: *12 bit* NPL: *3 bit* spiazzamento logico: *12* bit

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria virtuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 ... CY0 ...), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 4 K DX:12 K PX:4 K
CY:16 K DY: 4 K PY:4 K

indir. virtuale	prog. X	prog. Y
0	*CX0*	*CY0*
1	*DX0*	*CY1*
2	*DX1*	*CY2*
3	*DX2*	*CY3*
4		*DY0*
5		
6		
7	*PX0*	*PY0*

(c) Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R e S.

A pagina seguente sono indicate due serie di eventi; la prima serie termina all'istante t_0, la seconda all'istante t_1.

Si compilino le tabelle che descrivono i contenuti della memoria fisica e della MMU agli istanti t_0 e t_1, utilizzando la notazione CP0 CP1 DP0 PP0 ... CQ0 ... per indicare le pagine virtuali dei processi; nelle tabelle della MMU si aggiunga anche il NPV effettivo con la notazione CP0/0 ecc ...; in tutte le tabelle si usi la notazione (CP0) ecc ... per indicare un dato non più valido e la notazione CP0 = CQ0 ecc ... per indicare che una pagina fisica contiene più di una pagina logica.

Si considerino valide le seguenti ipotesi relative al sistema:

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 0AAA
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 39F2
- il numero di pagine residenti **R** vale **4**
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- in assenza di indicazioni esplicite relative all'accesso alle pagine, le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo; inoltre la pagina di codice caricata più di recente è acceduta continuamente
- al momento di una "fork" viene duplicata solamente la pagina di pila caricata più recentemente, ma tutte le pagine virtuali del padre sono considerate condivise con il figlio
- dopo una "fork", se uno dei due processi padre o figlio scrive in una pagina condivisa, la nuova pagina fisica che viene allocata appartiene al processo che ha eseguito la scrittura
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene **sempre** in sequenza, senza buchi, a partire dalla pagina fisica 0
- le righe della tabella della MMU vengono allocate ordinatamente man mano che vengono allocate le pagine di memoria virtuale
- gli eventi influenzanti la MMU partono da una situazione di tabella vergine
- se è richiesta una nuova riga di MMU, si utilizza **sempre** la prima riga libera

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede a pagine nel seguente ordine: dati 1, dati 0, pila
3. P crea due pagine dati dinamiche tramite BRK
4. creazione del processo Q e lancio del programma Y ("fork" di Q ed "exec" di Y)
5. Q salta all'istruzione di indirizzo 2B35, poi accede alla pila
6. Q crea 2 pagine di pila

Si compilino le tabelle sotto, al tempo t_0.

situazione al tempo t_0

memoria fisica	
indir. fisico	**pagine allocate**
0	*CP0*
1	*PP0*
2	*(DP1) DP4*
3	*(DP0) DP5*
4	*(CQ3) PQ2*
5	*PQ0*
6	*CQ2*
7	*PQ1*

MMU			
proc.	**NPV**	**NPF**	**valid bit**
P	*CP0/0*	*0*	*1*
P	*PP0/7*	*1*	*1*
P	*(DP1/2) DP4/5*	*2*	*1*
P	*(DP0/0 DP5/6*	*3*	*1*
Q	*(CQ3/3) PQ2/5*	*4*	*1*
Q	*PQ0/7*	*5*	*1*
Q	*CQ2/2*	*6*	*1*
Q	*PQ1/6*	*7*	*1*
			0
			0
			0
			0

(e) A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

7. P termina ("exit" di P)
8. Q esegue una fork e crea il processo R
9. R esegue una fork e crea il processo S

Si compilino le tabelle sotto, al tempo t_1.

situazione al tempo t_1

memoria fisica	
indir. fisico	**pagine allocate**
0	*(CP0) PR2*
1	*(PP0) PS2*
2	*(DP1) (DP4)*
3	*(DP0) (DP5)*
4	*(CQ3) PQ2*
5	*PQ0 = PR0 = PS0*
6	*CQ2 = CR2 = CS2*
7	*PQ1 = PR1 = PS1*

MMU			
proc.	**NPV**	**NPF**	**valid bit**
(P) R	*(CP0/0) PR2/5*	*(0) 0*	*(1)(0)1*
(P) R	*(PP0/7) PR0/7*	*(1) 5*	*(1)(0)1*
(P) R	*(DP1/2) (DP4/5) CR2/2*	*(2) 6*	*(1)(0)1*
(P) R	*(DP0/0) (DP5/6) PR1/6*	*(3) 7*	*(1)(0)1*
Q	*(CQ3/3) PQ2/5*	*4*	*1*
Q	*PQ0/7*	*5*	*1*
Q	*CQ2/2*	*6*	*1*
Q	*PQ1/6*	*7*	*1*
S	*PS2/5*	*1*	*(0) 1*
S	*PS0/7*	*5*	*(0) 1*
S	*CS2/2*	*6*	*(0) 1*
S	*PS1/6*	*7*	*(0) 1*

Esercizio n. 3 – memoria virtuale

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 K byte, quella logica di 32 K byte e la pagina ha taglia di 4 K byte. Si chiede di svolgere i punti seguenti:

(a) **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi:

NPF: *3 bit* Spiazzamento fisico: *12 bit* NPL: *3 bit* Spiazzamento logico: *12* bit

(b) **Si inserisca** nella tabella a fianco la struttura in pagine della memoria virtuale di due programmi X e Y (mediante la notazione CX0 CX1 DX0 PX0 … CY0 …), sapendo che la dimensione iniziale dei segmenti di tali programmi è la seguente:

CX: 12 K DX: 8 K PX: 4 K
CY: 8 K DY: 8 K PY: 4 K

Inoltre ambedue i programmi hanno un segmento condiviso CON di 4 K byte allocato subito dopo l'area dati

indir. virtuale	prog. X	prog. Y
0	*CX0*	*CY0*
1	*CX1*	*CY1*
2	*CX2*	*DY0*
3	*DX0*	*DY1*
4	*DX1*	*CON*
5	*CON*	
6		
7	*PX0*	*PY0*

(c) Nel sistema vengono creati alcuni processi, indicati con P, Q e R

A pagina seguente sono indicate due serie di eventi; la prima serie termina all'istante di tempo t_0, la seconda all'istante di tempo t_1 (con $t_1 > t_0$).

Nota bene: la tabella illustra la struttura del programma (X e Y) e pertanto le pagine riportano il nome del programma (non dei processi che poi eseguiranno il programma, i quali non sono ancora noti).

Si compilino le tabelle che descrivono i contenuti della memoria fisica e della MMU agli istanti di tempo t_0 e t_1, utilizzando la notazione CP0 CP1 DP0 PP0 … CQ0 … per indicare le pagine virtuali dei processi; nelle tabelle della MMU si aggiunga anche il NPV effettivo con la notazione CP0/0, ecc…; in tutte le tabelle si usi la notazione (CP0) ecc… per indicare un dato non più valido e la notazione CP0 = CQ0 ecc… per indicare che una pagina fisica contiene più di una pagina logica.

Si considerino valide le seguenti ipotesi relative al sistema:

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè si caricano le pagine su richiesta senza scaricare le precedenti fino a raggiungere il numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 2CAF
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 19F2
- il numero **R** di pagine residenti vale **4**
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo, con questa unica eccezione: se è residente una sola pagina di codice, di certo quella è stata utilizzata recentemente
- al momento di una "fork" viene duplicata solamente la pagina di pila caricata più di recente, ma tutte le pagine virtuali del padre sono considerate condivise con il figlio
- dopo una "fork", se uno dei due processi padre o figlio scrive in una pagina condivisa, la nuova pagina fisica allocata appartiene al processo che ha fatto la scrittura
- l'allocazione delle pagine virtuali nelle pagine fisiche avviene **sempre** in sequenza, senza buchi, a partire dalla pagina fisica 0
- le righe della tabella della MMU vengono allocate ordinatamente in contemporanea all'allocazione delle pagine di memoria virtuale
- gli eventi influenzanti la MMU partono da una situazione di tabella vergine
- se è richiesta una nuova riga di MMU si utilizza **sempre** la prima riga libera (partendo dall'alto)

(d) A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. creazione del processo P e lancio del programma X ("fork" di P ed "exec" di X)
2. P accede alle 2 pagine dati (prima la 0 poi la 1)
3. creazione del processo Q come figlio di P ("fork" eseguita da P)
4. P salta all'istruzione di indirizzo virtuale (esadecimale) 0F24
5. Q scrive nella pagina dati DQ1
6. Q crea una nuova pagina di pila

Si compilino le tabelle sotto, al tempo t_0.

situazione al tempo t_0

memoria fisica	
indir. fisico	**pagine allocate**
0	*CP2 = CQ2*
1	*(PP0) CP0*
2	*DP0 = DQ0*
3	*DP1 = (DQ1)*
4	*(PQ0) PQ1*
5	*DQ1*
6	
7	

unità di gestione memoria - MMU			
proc.	**NPV**	**NPF**	**bit di validità**
P	*CP2/2*	*0*	*1*
P	*(PP0/7) CP0/0*	*1*	*1*
P	*DP0/3*	*2*	*1*
P	*DP1/4*	*3*	*1*
Q	*CQ2/2*	*0*	*1*
Q	*(PQ0/7) PQ1/6*	*4*	*1*
Q	*DQ0/3*	*2*	*1*
Q	*DQ1/4*	*(3) 5*	*1*

(e) A un certo istante di tempo $t_1 > t_0$ sono terminati, nell'ordine, gli eventi seguenti:

7. P termina ("exit" di P)
8. creazione del processo R e lancio del programma Y ("fork" di R ed "exec" di Y)
9. R accede al segmento CON
10. Q accede al segmento CON

Si compilino le tabelle sotto, al tempo t_1.

situazione al tempo t_1

memoria fisica	
indir. fisico	**pagine allocate**
0	*(CP2) = CQ2*
1	*(CP0) CR1*
2	*(DP0) = (DQ0)*
3	*(DP1) PR0*
4	*PQ1*
5	*DQ1*
6	*CON*
7	

unità di gestione memoria - MMU			
proc.	**NPV**	**NPF**	**bit di validità**
(P) R	*(CP2/2) CR1/1*	*(0) 1*	*(0) 1*
(P) R	*(CP0/0) PR0/7*	*(1) 3*	*(0) 1*
(P) R	*(DP0/3) CON/4*	*(2) 6*	*(0) 1*
(P) Q	*(DP1/4) CON/5*	*(3) 6*	*(0) 1*
Q	*CQ2/2*	*0*	*1*
Q	*(PQ0/7) PQ1/6*	*4*	*1*
(Q)	*(DQ0/3)*	*(2)*	*(1) 0*
Q	*DQ1/4*	*(3) 5*	*1*

Nota bene: le tabelle ai punti (d) e (e) illustrano la struttura dei processi P, Q e R, pertanto le pagine riportano i nomi di tali processi (non dei programmi X e Y eseguiti); per capire esattamente l'evoluzione della tabella di MMU e come compilarne le righe, si considerino attentamente le specifiche date al punto (c).

Esercizio n. 3 – Memoria Virtuale

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica è di 64 Kbyte, l'indirizzo logico è di 16 bit, la dimensione di pagina è di 4 Kbyte. Si svolgano i punti seguenti:

(a) **Si definisca** la struttura dell'indirizzo fisico dando la lunghezza in bit dei campi costituenti:

NPF: 4 *bit* _____ Spiazzamento fisico: 12 *bit* _____

(b) I tre programmi PGP, PGQ e PGR, che verranno eseguiti dai tre processi P, Q e R, rispettivamente, hanno la struttura di segmentazione iniziale seguente (e condividono il segmento COND):

CP: 8 K	DP: 4 K	PP: 8 K	COND: 4 K
CQ: 12 K	DQ: 4 K	PQ: 4 K	COND: 4 K
CR: 8 K	DR: 4 K	PR: 4 K	COND: 4 K

La dimensione complessiva dello spazio di indirizzi di ogni processo è di 64 K. Il segmento pila di ogni processo inizia dal fondo di tale spazio e cresce verso l'indirizzo iniziale. Onde permettere la crescita dello Heap (dati dinamici, funzione `malloc`), nei processi P, Q e R il segmento condiviso COND è allocato lasciando 2, 1 e 2 pagine libere dopo i segmenti dati DP, DQ e DR, rispettivamente.

Si definisca in tabella 1A il significato delle varie pagine di memoria logica, tramite la notazione usuale: CP0, CP1, DP0, PP0, ..., CQ0, ..., COND0, ... (ecc).

Indirizzo di pagina virtuale	Processo **P**	Processo **Q**	Processo **R**
0	*CP0*	*CQ0*	*CR0*
1	*CP1*	*CQ1*	*CR1*
2	*DP0*	*CQ2*	*DR0*
3		*DQ0*	
4		*DQ1*	
5	*COND0*	*COND0*	*COND0*
6			
7			
8			
9			
A			
B	*PP4*		
C	*PP3*		
D	*PP2*		
E	*PP1*		
F	*PP0*	*PQ0*	*PR0*

1A) Struttura della Memoria Logica

Indirizzo fisico	Pagine allocate t_0
0	***occupata***
1	*CP0*
2	*CP1*
3	***occupata***
4	*DP0*
5	*COND0*
6	*PP0*
7	*PP1*
8	*CQ0*
9	*CQ1*
A	*CQ2*
B	***occupata***
C	*DQ0*
D	*PQ0*
E	
F	

Indirizzo fisico	Pagine allocate t_1
0	***occupata***
1	*(CP0) PP2*
2	*CP1*
3	***(occ)*** *CR0*
4	*(DP0) PP3*
5	*COND0*
6	*(PP0) PP4*
7	*PP1*
8	*(CQ0) DQ1*
9	*CQ1*
A	*CQ2*
B	***(occ)*** *CR1*
C	*DQ0*
D	*PQ0*
E	*DR0*
F	*PR0*

1B) Memoria Fisica agli istanti di tempo t_0 e t_1

(c) **Si indichi** quanto spazio (in pagine) è disponibile per la chiamata a funzione nei processi:

P: 8 *pagine*_____ R: 9 *pagine* _____

(d) A un certo istante di tempo t_0 si è conclusa la seguente sequenza di eventi:

1. lancia un processo diverso da P, Q, R e occupa le pagine di indirizzo fisico 0, 3, B
2. lancia il processo P (fork di P ed exec di PGP)
3. lancia il processo Q (fork di Q ed exec di PGQ)

Si indichi l'allocazione fisica delle pagine all'istante di tempo t_0, compilando tabella 1B (parte sinistra). Valgono le ipotesi seguenti:

- il numero di pagine residenti per ciascun processo vale **R** = 6
- le pagine virtuali vanno allocate in quelle fisiche consecutivamente, a partire dalla pagina fisica 0
- l'algoritmo di sostituzione è LRU (prima si dealloca una pagina e poi si fa la nuova assegnazione)
- in ciascun processo, le pagine meno usate sono le seguenti: la prima di codice, la prima di dati e la prima di pila (sono elencate in ordine di frequenza d'uso crescente)

Si scriva "occupata" per riferirsi a una pagina attribuita a un altro processo (diverso da P, Q e R).

(e) A un certo istante di tempo $t_1 > t_0$ si è conclusa la seguente sequenza di eventi aggiuntivi:

4. dealloca le pagine di indirizzo fisico 3 e B
5. lancia il processo R (fork di R ed exec di PGR)
6. alloca tre nuove pagine di pila per P
7. alloca una nuova pagina di dati per Q

Si indichi l'allocazione fisica delle pagine all'istante di tempo t_1, compilando tabella 1B (parte destra) nelle medesime ipotesi elencate nel punto precedente (d) più l'ipotesi seguente: se occorre scegliere una pagina fisica libera, si prende sempre quella avente numero minore.

(f) **Si indichi** il contenuto della tabella delle pagine della MMU all'istante di tempo t_1, compilando la tabella predisposta sotto. Valgono le ipotesi seguenti:

- gli eventi influenzanti la MMU descritti ai punti (d, e) partono da una situazione di tabella vergine
- ogniqualvolta è richiesta una nuova riga va utilizzata la prima libera (senza curarsi se essa sia una riga vergine oppure sia stata liberata)

In prima colonna si annoti il pid mediante i simboli seguenti:

- "P", "Q" o "R" se la riga è utilizzata e la pagina è attribuita al processo P, Q o R, rispettivamente
- "?" se la riga è utilizzata ma la pagina è attribuita a un processo diverso da P, Q o R
- "NS" se la riga è libera (perché non è mai stata utilizzata oppure è stata liberata)

Si indichi anche il valore assunto dal bit di validità di pagina (valore 1 se la pagina è caricata).

PID	Num. Pag. Virt.	Num. Pag. Fis.	Bit di Validità
?	*non noto*	*0*	*1*
? R	*non noto 0 (CR0)*	*3 3*	*1 1*
? R	*non noto 1 (CR1)*	*B B*	*1 1*
P	*0 (CP0) D (PP2)*	*1 1*	*1 1*
P	*1 (CP1)*	*2*	*1*
P	*2 (DP0) C (PP3)*	*4 4*	*1 1*
P	*5 (COND0)*	*5*	*1*
P	*F (PP0) B (PP4)*	*6 6*	*1 1*
P	*E (PP1)*	*7*	*1*
Q	*0 (CQ0) 4 (DQ1)*	*8 8*	*1 1*
Q	*1 (CQ1)*	*9*	*1*
Q	*2 (CQ2)*	*A*	*1*
Q	*3 (DQ0)*	*C*	*1*
Q	*5 (COND0)*	*5*	*1*
Q	*F (PQ0)*	*D*	*1*
R	*2 (DR0)*	*E*	*1*
R	*5 (COND)*	*5*	*1*
R	*F (PR0)*	*F*	*1*

Nota bene: in rosso sono indicate le (potenziali) variazioni intervenute all'istante di tempo t_1; indicare tali variazioni anche in tabella 1A è del tutto facoltativo, benché possa essere chiarificante.