SISTEMI OPERATIVI, CORSI A e B PRIMA VERIFICA INTERMEDIA - 3/4 /2007 COMPITO 2

COGNOME E NOME ..... MATRICOLA ..... FILA POSTO

ESERCIZIO 1 (4 punti)

Si consideri un processore che dispone dei seguenti registri:

- i registri speciali PC (program counter), PS (program status) e SP(stack pointer);
- i registri R1 e R2, utilizzati sia nello stato utente che in quello supervisore.

Inoltre riserva un'area di memoria (appartenente al nucleo) per il vettore di interruzione e per lo stack del nucleo. Il vettore di interruzione contiene, per ogni interruzione, un indirizzo nell'area di memoria del nucleo e una parola di stato.

- Al riconoscimento di un'interruzione, l'hardware salva tutti i registri nello stack del nucleo (azzerando contemporaneamente i registri R1 e R2) e salta alla funzione di servizio dell'interruzione.
- Nella fase di esecuzione dell'istruzione IRET, l'hardware ripristina tutti i registri dallo stack del nucleo.

Il sistema operativo utilizza uno scheduling a priorità (vengono messi in esecuzione i processi con valore maggiore di priorità) e con prerilascio.

Al tempo T i soli processi presenti nel sistema sono P1 che è in esecuzione e il processo P2 che è bloccato in attesa del completamento di un'operazione di input/output sul disco. In un istante successivo arriva un'interruzione dal disco che segnala il completamento dell'operazione di input/output e che attiva la routine di gestione dell'interruzione, la quale riattiva il processo P2. Completata la gestione dell'interruzione la routine invoca lo schedulatore che termina il suo compito con l'istruzione IRET.

Nell'istante in cui arriva l'interruzione dal dispositivo che ha completato l'operazione di input/output, i registri del processore, i descrittori di P1 e P2 e lo stack del nucleo hanno i contenuti mostrati in figura, che mostra anche il contenuto dell'elemento del vettore di interruzione associato alle interruzioni da programma.

Descrittore di P1	
Stato	In esec.
priorità	3
PC	3E4F
PS	6544
SP	1209
R1	C100
R2	0002

Descrittore di P2	
Stato	bloccato
priorità	2
PC	894F
PS	9887
SP	7652
R1	1E34
R2	EE00

Stack del nucleo	
0FFF	……
1000	
1001	
1002	
1003	
1004	

Registri	
PC	B323
PS	AF92
SP	6500
R1	4617
R2	7676

SP nucleo	0FFF

Vettore di interruzione	
INDIRIZZO	2098
PAROLA DI STATO	1111

Siano:

1) T+x l'istante in cui viene estratta la prima istruzione della routine di gestione dell'interruzione
2) T+y l'istante in cui viene estratta l'istruzione IRET
3) T+z l'istante in cui termina l'esecuzione dell'istruzione IRET

Si chiede:

1) Lo stato del processore nell'istante T+x
2) Il contenuto dei registri, dei descrittori e dello stack del nucleo nell'istante T+x;
3) Il contenuto dei registri, dei descrittori e dello stack del nucleo nell'istante T+y;
4) Il contenuto dei registri, dei descrittori e dello stack del nucleo nell'istante T+z;
5) Lo stato del processore nell'istante T+z

SOLUZIONE

1) Stato del processore: SUPERVISORE

Nelle tabelle seguenti: riportare solo i contenuti che cambiano

2)

Descrittore di P1	
Stato	In esec.
priorità	3
PC	invariato
PS	invariato
SP	invariato
R1	invariato
R2	invariato

Descrittore di P2	
stato	bloccato
priorità	2
PC	invariato
PS	invariato
SP	invariato
R1	invariato
R2	invariato

Stack del nucleo	
0FFF	……
1000	B323
1001	AF92
1002	6500
1003	4617
1004	7676

Registri	
PC	2098
PS	1111
SP	6500
R1	0
R2	0
SP nucleo	1004

3)

Descrittore di P1	
Stato	In esec.
priorità	3
PC	B323
PS	AF92
SP	6500
R1	4617
R2	7676

Descrittore di P2	
stato	Pronto
priorità	2
PC	invariato
PS	invariato
SP	invariato
R1	invariato
R2	invariato

Stack del nucleo

0FFF	……
1000	B323
1001	AF92
1002	6500
1003	4617
1004	7676

Registri

PC	????
PS	????
SP	6500
R1	????
R2	????
SP nucleo	1004

4)

Descrittore di P1	
Stato	In esec.
priorità	3
PC	B323
PS	AF92
SP	6500
R1	4617
R2	7676

Descrittore di P2	
stato	Pronto
priorità	2
PC	invariato
PS	invariato
SP	invariato
R1	invariato
R2	invariato

Stack del nucleo

0FFF	……
1000	
1001	
1002	
1003	
1004	

Registri

PC	B323
PS	AF92
SP	6500
R1	4617
R2	7676
SP nucleo	0FFF

5) Stato del processore: UTENTE

ESERCIZIO 2 (4 punti)

In un sistema operativo che realizza i threads a livello utente, i thread T_1, T_2 e T_3 del processo P competono in un gioco di carte. I tre threads avanzano senza sincronizzarsi ed eseguono tutti lo stesso codice:

```
1. While (true) {
2.     lock (&Chiave);
3.           MescolaCarte(&Mazzo);
4.           ThreadYield();
5.           EstraiCarta(&Carta);
6.     unlock (&Chiave);
7.     ThreadYield();
8.     UtilizzaCarta(Carta);
9.     lock (&Chiave);
10.          InserisciCarta(&Mazzo,Carta);
11.    unlock (&Chiave);
12.    ThreadYield();
       }
```

Lo scheduling dei thread avviene senza prerilascio.

Si fanno le seguenti ipotesi:

- `lock(&Chiave)` esegue `ThreadYield()` nel caso in cui il valore di `chiave` sia `0` (occupato), mentre `unlock(&Chiave)` non esegue `ThreadYield()` quando assegna il valore 1 alla `chiave`.
- le operazioni `MescolaCarte`, `EstraiCarta`, `UtilizzaCarta` e `InserisciCarta` richiedono 10 msec. Tutte le altre operazioni richiedono un tempo trascurabile.
- Al tempo t T_2 è in stato di pronto e deve eseguire l'istruzione 1.
- Al tempo t T_3 è in stato di pronto e deve eseguire l'istruzione 1.
- al tempo t , l'ordinamento dei thread in coda pronti è T_2 -> T_3 ;

Se il thread T_1 esegue l'istruzione 4 al tempo t , quanto tempo impiega T_3 per completare l'istruzione 3?

SOLUZIONE

(riempire una riga della tabella ad ogni nuova assegnazione del processore ai thread)

Tempo	Thread in esecuzione	Prossima istruzione T1	Prossima istruzione T2	Prossima istruzione T3	Coda pronti
T	T2	5	1	1	T3 -> T1
T	T3	5	2	1	T1 -> T2
t	T1	5	2	2	T2 -> T3
t + 10	T2	8	2	2	T3 -> T1
t + 20	T3	8	5	2	T1 -> T2
t + 20	T1	8	5	2	T2 -> T3
t + 30	T2	9	5	2	T3 -> T1

t + 40	T3	9	8	2	T1 -> T2
t + 50	T1	9	8	5	T2 -> T3

T3 completa l'istruzione 10 al tempo t+50

ESERCIZIO 3 (4 punti)
Una biblioteca dispone di 5 copie di un libro, che gli utenti possono prendere in prestito Gli mutenti sono thread di uno stesso processo, conformi allo standard POSIX, che si sincronizzano mediante il meccanismo *mutex* (sul quale sono definite le operazioni `pthread_mutex_lock` e `pthread_mutex_unlock`), e il meccanismo *condition* (sul quale sono definite le operazioni `pthread_cond_wait` e `pthread_cond_signal`).
Per accedere al prestito, i thread condividono la variabile *CopieDisponibili,* con valore iniziale 5, la variabile *MutexPrestito* di tipo *mutex*, e la variabile *PrestitoDisponibile,* di tipo *condition.*
A meno delle necessarie operazioni di scincronizzazione, ogni thread che utilizza il parcheggio esegue la seguente sequenza di operazioni:

```
<l'utente entra entyra nella biblioteca per richiedere il prestito >
if CopieDisponibili = 0
//l'utente attende la disponibilità di una copia//
//l'utente può ottenere una copia in prestito//
<il bibliotecario preleva una copia e la consegna >
CopieDisponibili - -
<l'utente utilizza il libro>
<l'utente riconsegna il libro>
CopieDisponibili ++
<l'utente esce dalla bibliorteca>
```

Si chiede di inserire le corrette operazioni di sincronizzazione con i meccanismi *mutex* e *condition*.

SOLUZIONE

```
<l'utente entra entra nella biblioteca per richiedere il prestito >
pthread_mutex_lock(&MutexPrestito)
if CopieDisponibili = 0 pthread_cond_wait(&PrestitoDisponibile,&MutexPrestito);
//l'utente attende la disponibilità di una copia//
//l'utente può ottenere una copia in prestito //
<il bibliotecario preleva una copia e la consegna >
CopieDisponibili - -
pthread_mutex_unlock(&MutexPrestito);
<l'utente utilizza il libro>
pthread_mutex_lock(&MutexPrestito);
<l'utente riconsegna il libro>
<il bibliotecario ripone il libro nello scaffale>
CopieDisponibili ++
pthread_cond_signal(&PrestitoDisponibile);
pthread_mutex_unlock(&MutexPrestito);
<l'utente esce dalla biblioteca>
```

ESERCIZIO 4 (4 punti)

Un sistema con processi A, B, C, D, E e risorse dei tipi R1, R2, R3, R4, rispettivamente di molteplicità [4, 5, 5, 6], utilizza l'algoritmo del banchiere per evitare lo stallo. Il sistema ha raggiunto lo stato (sicuro) mostrato nelle tabelle seguenti.

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

Esigenza residua (esclusa l'assegnazione attuale)				
	R1	R2	R3	R4
A	0	1	2	0
B	2	0	1	2
C	1	0	0	0
D	0	4	1	1
E	1	1	1	0

Molteplicità			
R1	R2	R3	R4
4	5	5	6

Disponibilità			
1	2	1	1

Si considerino ora i seguenti casi (in alternativa):

a. il processo D richiede un'istanza della risorsa R3
b. Il processo B richiede un'istanza della risorsa R1

Dire in ognuno dei due casi se la richiesta è accettata dal sistema operativo.

SOLUZIONE

a) Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R3 al processo D:

Assegnazione attuale				
	R1	R2	R3	R4
A			1	2
B	2			1
C		2		1
D			3	
E	1	1	1	1

Esigenza residua (esclusa l'assegnazione attuale)				
	R1	R2	R3	R4
A	0	1	2	0
B	2	0	1	2
C	1	0	0	0
D	0	4	0	1
E	1	1	1	0

Molteplicità			
R1	R2	R3	R4
4	5	5	6

Disponibilità			
1	2	0	1

a1) Il processo C può terminare
La disponibilità di {R1, R2, R3, R4} diviene { 1,4,2,0}
a2) Il processo D può terminare
La disponibilità di {R1, R2, R3, R4} diviene { 1,4,3,2
a3) Il processo A può terminare
La disponibilità di {R1, R2, R3, R4} diviene {1,4,4,4}
a4) Il processo E può terminare
La disponibilità di {R1, R2, R3, R4} diviene {2,5,5,5}
a5) Il processo B…… può terminare
La disponibilità di {R1, R2, R3, R4} diviene {4,5,4,6}

Di conseguenza: assegnazione assegnata? SI

b) Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R1 al processo B:

Assegnazione attuale				
	R1	R2	R3	R4
A	0	0	1	2

Esigenza residua (esclusa l'assegnazione attuale)				
	R1	R2	R3	R4
A	0	1	2	0

Molteplicità			
R1	R2	R3	R4
4	5	5	6

B	3	0	0	1
C	0	2	0	1
D	0	0	2	0
E	1	1	1	1

B	1	0	1	2
C	1	0	0	0
D	0	4	1	1
E	1	1	1	0

Disponibilità			
0	2	1	1

Di conseguenza: assegnazione assegnata? NO

ESERCIZIO 5 (4 punti)

In un sistema che gestisce il processore con la politica RoundRobin e quanto di tempo di 10 msec. vengono generati 5 processi (A,B,C), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

Processo	Durata	Tempo di arrivo
A	25	0
B	18	7
C	13	15

Calcolare il tempo di permanenza nel sistema di ogni processo (definito come differenza tra il tempo di completamento dell'esecuzione e il tempo di arrivo nel sistema), supponendo che si verifichino (in sequenza) i seguenti eventi:

1. Al tempo T+19 si sospende il processo in esecuzione
2. Al tempo T+33 viene riattivato il processo sospeso

E supponendo che nessun altro processo si sospenda.
Mostrare come si evolve il sistema a ogni riassegnazione del processore e comunque al tempo T+33 (dopo la riattivazione del processo sospeso).

SOLUZIONE

Tempo T+	proc. in esecuzione	coda pronti	Processi sospesi
0	A (25)	-	-
7	A (18)	B (18)	-
10	B (18)	A (15)	-
15	B (13)	A (15) , C (13)	-
19	A (15)	C (13)	B (9)
29	C (13)	A (5)	B (9)
33	C (9)	A (5), B (9)	-
39	A (5)	B (9), C(3)	-
44	B (9) *	C (3)	-
53	C (3) **	-	-
56	-	-	-

*A Termina all'istante 44
** B termina all'istante 53
*** C termina all'istante 56

Tempi di parmanennza nel sistema:
Processo A: 44- 0= 44; Processo B: 53- 7= 46; Processo C: 56- 15= 41;

ESERCIZIO 6 (2 punti)

Si consideri un sistema nel quale è definito il semaforo *sem1* e i processi P1, P2 e P3.
Al tempo *t* il semaforo *sem1* ha la seguente configurazione:
Sem1: valore 1, coda ∅
Allo stesso tempo, la CodaPronti contiene i processi P2 -> P3 e il processo P1 è in esecuzione.
Lo scheduler dei processi non prevede il prerilascio del processore.
Come si modificano il semaforo *sem1* e la CodaPronti e quale processo è in esecuzione se si verificano (in alternativa) le due seguenti sequenze di eventi:

a) P1 esegue *wait (Sem1)* e successivamente il processo in esecuzione esegue *wait(Sem1);*
b) P1 esegue *signal (Sem1)* e successivamente il processo in esecuzione esegue *wait(Sem1);*

SOLUZIONE

	Sequenze di eventi	In	Coda	Sem1

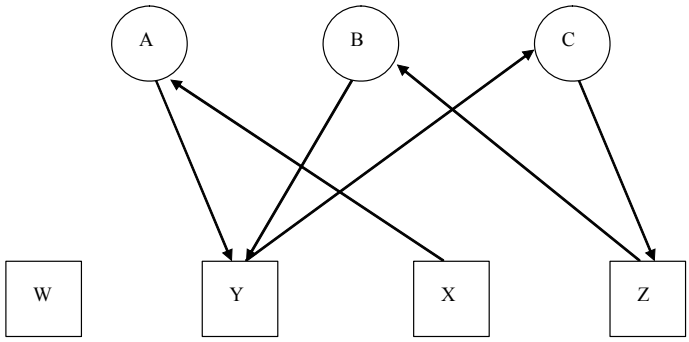
		Esecuzione	Pronti	
a-1	P1 esegue *wait (Sem1)*	P1	P2-> P3	0, ∅
a-2	Il processo in esecuzione esegue *wait (Sem1)*	P2	P3	0, P1
b-1	P1 esegue *signal (Sem1)*	P1	P2-> P3	2, ∅
b-2	Il processo in esecuzione esegue *wait (Sem1)*	P1	P2-> P3	1, ∅

ESERCIZIO 7 (2 punti)
Un sistema con 3 processi (A,B e C) e quattro risorse singole (W, X, Y e Z) esegue la seguente sequenza di assegnazioni e rilasci:

1) B richiede W
2) A richiede X
3) B richiede Z
4) C richiede Y
5) B rilascia W
6) A richiede Y
7) B richiede Y
8) C richiede Z

mostrare lo stato raggiunto dal sistema al termine della sequenza e dire se il sistema è in stallo.

SOLUZIONE



Il sistema è in stallo [si/no] ? SI, perché si verifica attesa circolare tra i processi B, C e le risorse Y, Z.

ESERCIZIO 8 (2 punti)

Quali delle seguenti operazioni possono essere eseguite solo in stato supervisore ?

	Solo Supervisore
Istruzione SVC	
Modificare il PID di un processo	SI
Istruzione TSL	
Provocare la terminazione di un processo figlio (segnale SIGKILL) in Unix	
Modificare i registri di controllo di una unità DMA	SI

ESERCIZIO 9 (2 punti)
Nel sistema UNIX, si considerino i processi P, F1 e F2, dove F1 e F2 sono figli di P, che non ha altri figli.
In quale stato si trovano il processo F1 e il processo F2 al tempo 26 e al tempo 28 nelle seguenti ipotesi:

1. Il processo P esegue la chiamata di sistema `wait` unicamente al tempo 27 e i processi F1 ed F2 eseguono la chiamata di sistema `exit` rispettivamente al tempo 10 e al tempo 20;
2. Il processo P esegue chiamate di sistema `wait` al tempo 18 e al tempo 27, e i processi F1 ed F2 eseguono la chiamata di sistema `exit` rispettivamente al tempo 10 e al tempo 20;

SOLUZIONE

1.	Tempo 26	Stato di F1: ZOMBIE	Stato di F2: ZOMBIE
	Tempo 28	Stato di F1: TERMINATO	Stato di F2: ZOMBIE
2.	Tempo 26	Stato di F1: TERMINATO	Stato di F2: ZOMBIE
	Tempo 28	Stato di F1: TERMINATO	Stato di F2: TERMINATO

ESERCIZIO 10 (2 punti)

Si confrontino le primitive di sincronizzazione `wait` e `signal` con le omonime chiamate di sistema UNIX.
Si chiede se l'esecuzione di queste primitive o chiamate di sistema può produrre l'effetto di:

1) modificare il valore di un semaforo
2) modificare la coda di un semaforo
3) provocare la transizione di stato del processo in esecuzione
4) modificare la user structure del processo in esecuzione.

SOLUZIONE

	Può modificare il valore di un semaforo?	Può modificare la coda di un semaforo?	Può provocare la transizione di stato del processo in esecuzione?	Modifica il Signal Handler Array del processo?
Primitiva di sincronizzazione `wait`	SI	SI	SI	NO
Chiamata di sistema `wait di UNIX`	NO	NO	SI	NO
Primitiva di sincronizzazione `signal`	SI	SI	SI	NO
Chiamata di sistema `signal di UNIX`	NO	NO	NO	SI