COGNOME ..... NOME ..... MATRICOLA ..... CORSO |A| |B|

ESERCIZIO A-1 (4 punti)

Si consideri un processore che dispone dei seguenti registri:
- i registri speciali PC (program counter) e PS (program status)
- un banco di registri riservato allo stato utente, che comprende i registri generali R1, R2, R3, R4 e lo stack pointer SP,
- un ulteriore banco di registri riservato allo stato supervisore, che comprende i registri generali R'1, R'2, R'3, R'4 e lo stack pointer SP.

Il sistema gestisce il processore con politica Round Robin. Al tempo t, quando sono presenti nel sistema (tra gli altri) il processo Pi, in stato di esecuzione, e il processo Pj, che occupa la prima posizione della Coda Pronti, il timer lancia l'interruzione *TimerInt* che segnala l'esaurimento del quanto di tempo. Immediatamente dopo il tempo *t*, quando l'interruzione viene riconosciuta, i registri del processore, i descrittori di Pi e Pj e gli stack di Pi, di Pj e del nucleo hanno i contenuti mostrati in figura. Anche il vettore associato all'interruzione *TimerInt* è mostrato in figura.

L'interruzione determina l'intervento del nucleo, che esegue una funzione di servizio comprendente lo scheduler.

Supponendo che il vettore di interruzione associato all'interruzione *TimerInt* sia 0425 e che la parola di stato del nucleo sia 275E, si chiede:

a) il contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione della prima istruzione della funzione di servizio;
b) il contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione dell'istruzione IRET con la quale termina la funzione di servizio;
c) il contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

DESCRITTORE DI P_I	
Stato	Esec
PC	2E31
SP	2873
PS	16F2
R1	1234
R2	56CC
R3	0000
R4	0000

DESCRITTORE DI P_J	
Stato	Pronto
PC	A12C
SP	A275
PS	16F2
R1	25CC
R2	0000
R3	0056
R4	AA38

STACK DEL NUCLEO	
……	……
1016	23BB
1015	070A
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	2997
R1	6649
R2	02CE
R3	D410
R4	73FF
REG. STATO SUPERV.	
SP'	1015
R'1	0012
R'2	AACC
R'3	2345
R'4	789A

PROCESSORE: Registri speciali			
PC	2F00	PS	16F2

SOLUZIONE

a) contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione della prima istruzione della funzione di servizio:

DESCRITTORE DI P_I		DESCRITTORE DI P_J		STACK DEL NUCLEO		REG. STATO UTENTE	
Stato	Esec	Stato	Pronto	……	……	SP	2997
PC	2E31	PC	A12C	1016	23BB	R1	6649
SP	2873	SP	A275	1015	070A	R2	02CE
PS	16F2	PS	16F2	1014	2F00	R3	D410
R1	1234	R1	25CC	1013	16F2	R4	73FF
R2	56CC	R2	0000	1012			
R3	0000	R3	0056	1011		REG. STATO SUPERV.	
R4	0000	R4	AA38	1010		SP’	1013
						R’1	0012
						R’2	AACC
PROCESSORE: Registri speciali						R’3	2345
PC	0425	PS	275E			R’4	789A

b) contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione dell’istruzione IRET con la quale termina la funzione di servizio;

DESCRITTORE DI P_I		DESCRITTORE DI P_J		STACK DEL NUCLEO		REG. STATO UTENTE	
Stato	Pronto	Stato	Esec	……	……	SP	A275
PC	2F00	PC	A12C	1016	23BB	R1	25CC
SP	2997	SP	A275	1015	070A	R2	0000
PS	16F2	PS	16F2	1014	A12C	R3	0056
R1	6649	R1	25CC	1013	16F2	R4	AA38
R2	02CE	R2	0000	1012			
R3	D410	R3	0056	1011		REG. STATO SUPERV.	
R4	73FF	R4	AA38	1010		SP’	1013
						R’1	…
						R’2	…
PROCESSORE: Registri speciali						R’3	…
PC	….	PS	…..			R’4	…

c) contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione dell’istruzione eseguita subito dopo la IRET.

DESCRITTORE DI P_I		DESCRITTORE DI P_J		STACK DEL NUCLEO		REG. STATO UTENTE	
Stato	Pronto	Stato	Esec	……	……	SP	A275
PC	2F00	PC	A12C	1016	23BB	R1	25CC
SP	2997	SP	A275	1015	070A	R2	0000
PS	16F2	PS	16F2	1014		R3	0056
R1	6649	R1	25CC	1013		R4	AA38
R2	02CE	R2	0000	1012			
R3	D410	R3	0056	1011		REG. STATO SUPERV.	
R4	73FF	R4	AA38	1010		SP’	1015
						R’1	…
						R’2	…
PROCESSORE: Registri speciali						R’3	…
PC	A12C	PS	16F2			R’4	…

ESERCIZIO A.2 (4 punti)

In un sistema operativo che realizza i thread a livello kernel, i thread T11 e T12 del processo P cooperano scambiando messaggi attraverso un buffer circolare di n celle (numerate da 0 a n-1), ciascuna capace di contenere un messaggio.
Un possible schema di cooperazione, realizzato con i semafori CelleLibere (valore iniziale n) e MessaggiGiacenti (valore iniziale 0) e con le variabili IndDeposito e IndPrelievo (interi nell'intervallo [0,n), con IndDeposito ≠ IndPrelievo)) è il seguente:

Thread T11

```
……
ProduceMessaggio(mesaggio);
wait(CelleLibere);
DepositaNelBuffer(messaggio, IndDeposito);
IndDeposito ++ %n;
signal(MessaggiGiacenti);
……
```

Thread T12

```
……
wait(MessaggiGiacenti);
PrelevaDalBuffer(mess, IndPrelievo);
IndDeposito ++ %n;
signal(CelleLibere);
ConsumaMessaggio(mess);
……
```

Trasformare il precedente scherma usando i meccanismi mutex e condition della libreria p_thread (con operazioni p_thread_mutex_lock, p_thread_mutex_unlock, p_thread_condition_wait e p_thread_condition_signal) utilizzando:
- per il primo meccanismo la variabile MutexBuffer (di tipo pthread_mutex)
- per il secondo la variabile AttesaBuffer e AttesaPrelievo di tipo pthread_condition
- le variabili condivise CelleLibere (valore iniziale n) e, analogamente allo schema precedente, le variabili IndDeposito e IndPrelievo (interi nell'intervallo [0,n), con IndDeposito ≠ IndPrelievo)).

SOLUZIONE

Thread T11

```
……
ProduceMessaggio(messaggio);

p_thread_mutex_lock (MutexBuffer);
if CelleLibere== 0
   p_thread_condition_wait(AttesaDeposito,
                           MutexBuffer);
DepositaNelBuffer(messaggio, IndDeposito);
CelleLibere -- ;
IndDeposito ++ %n;
p_thread_condition_signal(AttesaPrelievo);
p_thread_mutex_unlock(MutexBuffer);
……
```

Thread T12

```
……
p_thread_mutex_lock (MutexBuffer);
if CelleLibere== n
   p_thread_condition_wait(AttesaPrelievo,
                           MutexBuffer);
PrelevaDalBuffer(mess, IndPrelievo);
CelleLibere ++ ;
IndPrelievo ++ %n;
p_thread_condition_signal(AttesaDeposito);
p_thread_mutex_unlock (MutexBuffer);
ConsumaMessaggio(mess);
……
```

Esercizio A.3 (3 punti)

In un sistema operativo che realizza i thread a livello kernel, i thread T1 e T2 del processo P si scambiano messaggi attraverso una pila a tre posizioni (la pila è indicizzata dalla variabile puntatore inizializzata a 0). Si consideri la seguente soluzione realizzata con il semaforo Mutex, inizializzato ad 1, il semaforo SemThread1 inizializzato a 3 e utilizzato dal thread T1 per bloccarsi quando non ci sono elementi liberi nella pila e il semaforo SemThread2 inizializzato a 0 e utilizzato dal thread T2 per bloccarsi quando non ci sono elementi da leggere dalla pila.

Thread T1:

```
While (True) {
      messaggio = produce_messaggio();

      wait(&Mutex);
      wait(&SemThread1);
      Pila[puntatore]= messaggio;
      puntatore ++;
      signal(&SemThread2);

      signal(&Mutex);

}
```

Thread T2:

```
While (True) {
      wait(&Mutex);
      wait(&SemThread2);
      puntatore --;
      mess= Pila[puntatore];
      signal(&SemThread1);

      signal(&Mutex);
      consuma_messaggio(mess);

}
```

La soluzione è corretta? Motivare la risposta.

SOLUZIONE

La soluzione è errata
Motivo: quando T1 si sospende perché la pila è piena (oppure quando T2 si sospende perché la pila è vuota) tiene bloccata la mutua esclusione e i due thread vanno in stallo.

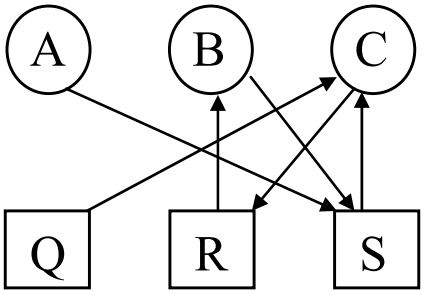
Esercizio A.4 (2 punti)

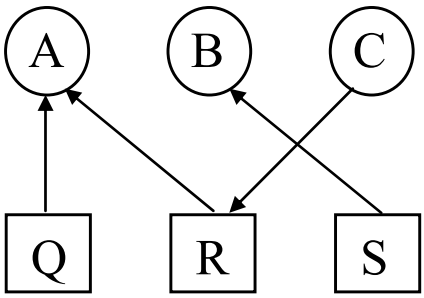
Dati tre processi A,B,C che osservano il paradigma di "possesso e attesa" e quattro risorse singole Q, R, S, utilizzabili in mutua esclusione e senza possibilità di prerilascio, supponiamo che il gestore assegni le risorse al processo richiedente alla sola condizione che la risorsa sia disponibile. Inizialmente tutte le risorse sono disponibili.
Si considerino le seguenti sequenze di richieste e rilasci:

Sequenza S1:	Sequenza S2:
1) A richiede Q; 2) C richiede S; 3) C richiede Q; 4) B richiede R; 5) B richiede S; 6) A rilascia Q; 7) A richiede S; 8) C richiede R.	1) A richiede R; 2) B richiede Q, 3) C richiede S; 4) C richiede R; 5) A richiede Q; 6) C rilascia S; 7) B rilascia Q; 8) B richiede S;

Queste sequenze provocano stallo? Motivare le risposte rappresentando nel grafo lo stato (assegnazioni e richieste pendenti) al termine di ciascuna sequenza.

SOLUZIONE





1) stallo [SI/NO] ? SI
2) stallo [SI/NO] ? NO

Esercizio A.5 (2 punti)

Dire quali delle seguenti parti di un processo Unix restano residenti in memoria principale anche quando il processo è nello stato di Swapped.

	residente/non residente
Stack	non residente
Codice	non residente
Dati	non residente
User Structure	non residente
Process Structure	residente
Text STructure	residente

PARTE B

Esercizio B.1 (4 punti)

SISTEMI OPERATIVI, CORSI A e B - TERZO APPELLO 2007- 8/6/2007

Un disco con 2 facce, 200 settori per traccia e 100 cilindri ha un tempo di seek proporzionale al numero di cilindri attraversati e pari a 2ms per ogni cilindro. Il periodo di rotazione è di 20 msec: conseguentemente il tempo impiegato per percorrere un settore è di 0,1 msec.
A un certo tempo (convenzionalmente indicato come t= 0) termina l'esecuzione dei comandi sul cilindro 61 e sono pervenute, nell'ordine, le seguenti richieste di lettura o scrittura:

- clindro 59, faccia 0, settore 100
- clindro 9, faccia 1, settore 120
- clindro 98, faccia 0, settore 55

Successivamente arrivano i seguenti, ulteriori comandi di lettura:

- al tempo 20: clindro 77, faccia 1, settore 5
- al tempo 40: clindro 64, faccia 0, settore 60
- al tempo 50: clindro 77, faccia 0, settore 5

Calcolare il tempo necessario per eseguire tutte queste operazioni supponendo che si adotti la politica di scheduling *Shortest Seek Time First* (SSTF).
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.
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.
Per il ritardo rotazionale dopo un'operazione di *seek* si assume sempre il valore di caso peggiore, pari a un intero periodo di rotazione (20 msec). Si assuma inoltre che i comandi sullo stesso cilindro vengano eseguiti in ordine FIFO.

SOLUZIONE

op. su cilindro:	59	settore:	100						
inizio:	0	seek:	4	rotazione:	20	percorrenza:	0,1	fine:	24,1
op. su cilindro:	77	settore:	5	faccia 1					
inizio:	24,1	seek:	36	rotazione:	20	percorrenza:	0,1	fine:	80,2
op. su cilindro:	77	settore:	5	faccia 0					
inizio:	80,2	seek:	0	rotazione:	19,9	percorrenza:	0,1	fine:	100,2
op. su cilindro:	64	settore:	60						
inizio:	100,2	seek:	26	rotazione:	20	percorrenza:	0,1	fine:	146,3
op. su cilindro:	98	settore:	55						
inizio:	146,3	seek:	68	rotazione:	20	percorrenza:	0,1	fine:	234,4
op. su cilindro:	9	settore:	120						
inizio:	234,4	seek:	178	rotazione:	20	percorrenza:	0,1	fine:	432,5

Esercizio B.2 (4 punti)

Si consideri un sistema che gestisce la memoria utilizzando un algoritmo di sostituzione LRU locale e attribuendo a ogni processo un numero `MaxBlocchi` di blocchi a disposizione del working set. Questo numero è definito dinamicamente e viene incrementato o decrementato dal processo di sistema `WorkingSetManager`, che interviene periodicamente e applica la seguente politica:

- se esistono pagine con distanza passata maggiore o uguale a 10, scarica tutte queste pagine e riduce di 1 il valore di `MaxBlocchi`;
- se tutte le pagine hanno distanza passata minore di 7, incrementa di 1 il valore di `MaxBlocchi`.

Al verificarsi di un errore di pagina si ha il seguente comportamento:

- se il numero di pagine attualmente caricate è minore di `MaxBlocchi`, la pagina riferita viene caricata assegnando il primo dei blocchi dalla lista `BlocchiDisponibili`.
- se il numero di pagine attualmente caricate è uguale a `MaxBlocchi`, viene scaricata la pagina prescelta dall'algoritmo di sostituzione e la pagina riferita viene caricata al suo posto.

A un certo istante, quando il tempo virtuale del processo A è t=10, la tabella delle pagine dei processo A (che riporta anche il tempo virtuale dell'ultimo riferimento) è la seguente:

Pagina	Blocco	Ultimo riferimento
0	-	
1	6	2
2	-	
3	-	
4	-	
5	8	9
6	-	
7	10	4
8	15	5
9	-	
Processo A		

Il valore di `MaxBlocchi` attualmente assegnato al processo A è `MaxBlocchi= 5` e i primi elementi della lista `BlocchiDisponibili` sono `20→25→15→` ….

A partire da questo istante il processo A avanza ed esegue i seguenti riferimenti alla memoria:

- al tempo 10 riferisce la pagina 5
- al tempo 11 riferisce la pagina 3
- al tempo 12 riferisce la pagina 0
- al tempo 13 riferisce la pagina 5
- al tempo 14 riferisce la pagina 0
- al tempo 15 riferisce la pagina 9

Dopo quest'ultimo riferimento interviene il processo `WorkingSetManager`, che applica la politica sopra definita.
Si chiede:

a) i caricamenti e gli eventuali scaricamenti di pagine eseguiti ai tempi 10, 11, 12, 13, 14, 15;
b) la configurazione della tabella delle pagine del processo A subito prima dell'intervento del processo `WorkingSetManager`
c) il valore di `MaxBlocchi` assegnato al processo A dopo l'intervento del processo `WorkingSetManager`;
d) la configurazione della tabella delle pagine del processo A quando termina l'intervento del processo `WorkingSetManager`

SOLUZIONE

- tempo 10: scaricata la pagina - ; caricata la pagina - nel blocco -
- tempo 11: scaricata la pagina - ; caricata la pagina 3 nel blocco 20
- tempo 12: scaricata la pagina 1 ; caricata la pagina 0 nel blocco 6
- tempo 13: scaricata la pagina - ; caricata la pagina - nel blocco -
- tempo 14: scaricata la pagina - ; caricata la pagina - nel blocco -
- tempo 15: scaricata la pagina 7 ; caricata la pagina 9 nel blocco 10

Nuove configurazioni della tabella delle pagine del processo A:

Pagina	Blocco	Ultimo riferimento
0	6	14
1	-	-
2	-	
3	20	11
4	-	
5	8	13
6	-	
7	-	-
8	15	5
9	10	15
Processo A		

Subito prima dell'intervento del processo `WorkingSetManager`

Pagina	Blocco	Ultimo riferimento
0	6	14

1	-	-
2	-	
3	20	11
4	-	
5	8	13
6	-	
7	-	-
8	-	-
9	10	15
Processo A		

Al termine dell'intervento del processo `WorkingSetManager`

Valore di `MaxBlocchi` assegnato al processo A dopo l'intervento del processo `WorkingSetManager`;

`MaxBlocchi= 4`

Esercizio B.3 (3 punti)

In un sistema UNIX, vengono eseguite in sequenza le seguenti operazioni:

a) il processo P apre il file F, la cui lunghezza corrente è di 250 caratteri;
b) il processo P legge 100 caratteri dal file F;
c) il processo P esegue con successo una *fork,* generando il processo P1;
d) il processo P1 esegue la chiamata *read* per leggere 200 caratteri dal file F;
e) il processo Q, padre del processo P, apre il file F;
f) il processo Q legge 150 caratteri dal file F.

Si cbiede:

- il valore del puntatore di lettura del processo P dopo l'operazione b);
- il valore del puntatore di lettura del processo P1 dopo l'operazione c);
- il numero di caratteri letti dal processo P1 con l'operazione d)
- il valore del puntatore di lettura del processo P1 dopo l'operazione d);
- il valore del puntatore di lettura del processo Q dopo l'operazione e);
- il valore del puntatore di lettura del processo Q dopo l'operazione f).

SOLUZIONE

- valore del puntatore di lettura del processo P dopo l'operazione b): 100
- valore del puntatore di lettura del processo P1 dopo l'operazione c): 100
- numero di caratteri letti dal processo P1 con l'operazione d): 150
- valore del puntatore di lettura del processo P1 dopo l'operazione d): 250
- valore del puntatore di lettura del processo Q dopo l'operazione e): 0
- valore del puntatore di lettura del processo Q dopo l'operazione f): 150

Esercizio B.4 (2 punti)

Dire quali delle seguenti tabelle di un processo Unix sono modificate dalla chiamata di sistema Open:

	SI/NO
process structure:	NO
user structure:	SI
Text Table:	NO
Tabella dei file aperti di sistema:	SI
Tabella dei file attivi:	SI

Esercizio B.5 (2 punti)

Un disco RAID di livello 4 è composto da 4 dischi fisici, numerati da 0 a 3. Le strip corrispondono a blocchi.
Il disco 3 è ridondante e il suo blocco di indice *i* contiene la parità dei blocchi di indice *i* dei dischi non ridondanti, cioè dei dischi 0, 1 e 2.
I blocchi di indice 7 dei dischi non ridondanti contendono rispettivamente:
Disco 0: 1 1 0 0 …
Disco 1: 1 0 0 1 …
Disco 2: 1 1 1 1 …
Disco 3: 1 0 1 0 …
Si chiede:

1) Come cambia il contenuto del blocco di indice 7 del disco ridondante se, in seguito a una scrittura, il contenuto del blocco 7 del disco 2 diviene 0 0 1 1 …
2) Quali blocchi è necessario leggere per modificare il contenuto del blocco di indice 7 del disco 2
3) Quali blocchi è necessario scrivere per modificare il contenuto del blocco di indice 7 del disco 2

SOLUZIONE

1) Contenuto del blocco di indice 7 del disco ridondante dopo la scrittura nel blocco 7 del disco 2:

Disco 0: 1 1 0 0 …
Disco 1: 1 0 0 1 …
Disco 2: 0 0 1 1 …
Disco 3: 0 1 1 0 …

2) E' necessario leggere i i blocchi: di indice 7 dei dischi 2 e 3

3) E' necessario scrivere i blocchi di indice 7 dei dischi 2 e 3