SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

Nome: _____ Matricola:_____ fila:____ posto: ____

ESERCIZIO A-1 (4 punti)

Si consideri un processore che dispone dei registri speciali PC (program counter) e PS (program status), di un banco di registri riservato allo stato utente che comprende i registri generali R1, R2, R3, R4 e lo stack pointer SP, e di 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'.
Al tempo t, quando sono presenti nel sistema il processo padre Pi e il suo processo figlio Pj, i registri del processore, il descrittore di Pi e di Pj e lo stack del nucleo hanno i contenuti mostrati in tabella:

DESCRITTORE DI P_I	
Stato	Pronto
PC	4000
PS	16F2
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

DESCRITTORE DI P_J	
Stato	Esec.
PC	8000
PS	16F2
SP	A000
R1	AAA1
R2	AAA2
R3	AAA3
R4	AAA4

PROCESSORE: Registri speciali			
PC	8100	PS	16F2

STACK DEL NUCLEO	
……	……
1016	AAAA
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	A020
R1	AA11
R2	AA22
R3	AA33
R4	AA44

REG. STATO SUPERV.	
SP'	1016
R'1	0000
R'2	0001
R'3	0002
R'4	0003

Al tempo t il processo Pj termina la propria esecuzione tramite la chiamata di sistema exit(). Il modello di terminazione è simile al modello di Unix, per cui il processo che termina passa il codice di terminazione al processo padre, il quale lo riceve tramite la primitiva wait().
All'istante di tempo t+1, quando termina la chiamata exit(), passa in esecuzione il processo Pi. La prima istruzione eseguita da Pi è l'istruzione contenuta nella locazione 4000 che è l'istruzione SVC (Supervisor Call) tramite la quale Pi invoca la chiamata di sistema wait(), per attendere la terminazione del processo figlio Pj.
L'invocazione della chiamata di sistema wait() al tempo t+1 determina l'intervento del nucleo, che esegue la funzione di servizio. Supponendo che il vettore di interruzione associato alla chiamata di sistema 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 caricamento dell'istruzione SVC contenuta nella locazione 4000 del processo Pi
b) 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;
c) 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;
d) 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.

SOLUZIONE

a) Contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo all'inizio della fase di estrazione dell'istruzione SVC contenuta all'indirizzo logico 4000 del processo Pi (nota: per semplicità si suppone che tutte le istruzioni abbiano la stessa lunghezza, uguale a un byte),

DESCRITTORE DI P_I	
Stato	Esec
PC	4000
PS	16F2
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

DESCRITTORE DI P_J	
Stato	Zombie
PC	8000
PS	16F2
SP	A000
R1	AAA1
R2	AAA2
R3	AAA3
R4	AAA4

STACK DEL NUCLEO	
……	……
1016	AAAA
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

REG. STATO SUPERV.	
SP'	1016
R'1	0000
R'2	0001

PROCESSORE: Registri speciali				
PC	4001		PS	16F2

R'3	0002
R'4	0003

b) 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	
Stato	Esec
PC	4000
PS	16F2
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

DESCRITTORE DI P_J	
Stato	Zombie
PC	8000
PS	16F2
SP	A000
R1	AAA1
R2	AAA2
R3	AAA3
R4	AAA4

STACK DEL NUCLEO	
……	……
1016	AAAA
1015	4001
1014	16F2
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

REG. STATO SUPERV.	
SP'	1014
R'1	0000
R'2	0001
R'3	0002
R'4	0003

PROCESSORE: Registri speciali			
PC	0425	PS	275E

c) 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	
Stato	Esec
PC	4000
PS	16F2
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

STACK DEL NUCLEO	
……	……
1016	AAAA
1015	4001
1014	16F2
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

REG. STATO SUPERV.	
SP'	1014
R'1	??
R'2	??
R'3	??
R'4	??

PROCESSORE: Registri speciali			
PC	0425+ ??	PS	275E

d) 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	4000
PS	16F2
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

STACK DEL NUCLEO	
……	……
1016	AAAA
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	F000
R1	1112
R2	2221
R3	3331
R4	4441

REG. STATO SUPERV.	
SP'	1016
R'1	Invariato
R'2	Invariato
R'3	Invariato
R'4	Invariato

PROCESSORE: Registri speciali			
PC	4001	PS	16F2

SOLUZIONE

a) Contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di caricamento dell'istruzione SVC contenuta nella locazione 4000 del processo Pi

DESCRITTORE DI P_I	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

PROCESSORE: Registri speciali			
PC		PS	

STACK DEL NUCLEO	
……	……
1016	
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	
R1	
R2	
R3	
R4	

REG. STATO SUPERV.	
SP’	
R’1	
R’2	
R’3	
R’4	

b) 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	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

PROCESSORE: Registri speciali			
PC		PS	

STACK DEL NUCLEO	
……	……
1016	
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	
R1	
R2	
R3	
R4	

REG. STATO SUPERV.	
SP’	
R’1	
R’2	
R’3	
R’4	

c) 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	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	
R4	

PROCESSORE: Registri speciali			
PC		PS	

STACK DEL NUCLEO	
……	……
1016	
1015	
1014	
1013	
1012	
1011	
1010	

REG. STATO UTENTE	
SP	
R1	
R2	
R3	
R4	

REG. STATO SUPERV.	
SP’	
R’1	
R’2	
R’3	
R’4	

d) 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	
PC	
PS	
SP	
R1	

DESCRITTORE DI P_J	
Stato	
PC	
PS	
SP	
R1	

STACK DEL NUCLEO	
……	……
1016	
1015	
1014	
1013	

REG. STATO UTENTE	
SP	
R1	
R2	
R3	
R4	

SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

R2		R2		1012			
R3		R3		1011		REG. STATO SUPERV.	
R4		R4		1010		SP'	
						R'1	
						R'2	
PROCESSORE: Registri speciali						R'3	
PC		PS				R'4	

ESERCIZIO A-2 (4 punti)

Si consideri un gioco di gruppo che consiste in una sequenza di partite, ciascuna con 5 partecipanti. Ogni partita è una gara ad eliminazione, i cui partecipanti sono eliminati uno alla volta fino a quando rimane un solo giocatore, che è il vincitore. L'ammissione al gioco può essere richiesta da un numero arbitrario di giocatori. I richiedenti rimangono in attesa fino a quando, terminata una partita, si può comporre un nuovo gruppo di 5 partecipanti. Precisamente, se al termine di una partita vi sono almeno 5 richiedenti in attesa, si ammettono immediatamente al gioco i primi 5; altrimenti i richiedenti rimangono in attesa fino a quando, con l'arrivo di una o più nuove richieste, si raggiunge il numero 5. Al termine della partita si ripete l'ammissione dei giocatori con le stesse modalità, e inizia una nuova partita.

Ogni giocatore è un'istanza del thread *Giocatore*. I thread si sincronizzano con i meccanismi dello standard POSIX, utilizzando i seguenti dati condivisi da tutti i thread:

- *GiocatoriInAttesa*: intero non negativo; valore iniziale 0;
- *GiocatoriAmmessi*: intero non negativo; valore iniziale 0;
- *PartitaInCorso:* booleano; valore iniziale 0;

e inoltre la variabile *P_mutex,* di tipo *mutex* e la variabile *AttesaGiocatori* di tipo *condition.*

Si chiede di completare il codice della generica istanza del thread *Giocatore,* inserendo le opportune funzioni di sincronizzazione. Si tenga presente che, dopo l'ammissione, la partecipazione al gioco è formalizzata con la funzione PartecipaAllaPartita (), che restituisce il valore 1 quando il giocatore è eliminato oppure esce dal gioco come vincitore.

SOLUZIONE

```
Giocatore
{
   <fa altre cose>;
   // decide di giocare una partita: prologo per l'ammissione al gioco
   pthread_mutex_lock(&P_mutex);
   GiocatoriInAttesa++;
   while  (PartitaInCorso==1 or GiocatoriInAttesa <5)  pthread_cond_wait(&AttesaGiocatori, &P_mutex);
   // il giocatore si sospende se è in corso una partita o, in caso contrario, se
   // non si è ancora costituito un gruppo di 5 giocatori in attesa di partecipare al gioco
   GiocatoriAmmessi++;
   if GiocatoriAmmessi<5 pthread_cond_signal(&AttesaGiocatori);
   // i giocatori ammessi riattivano a catena altri giocatori in attesa fino a costituire il gruppo di 5
   //  la variabile GiocatoriInAttesa è aggiornata dopo la riattivazione di tutti i 5 giocatori da ammettere
   else          {
        // GiocatoriAmmessi= 5
        PartitaInCorso=1; GiocatoriInAttesa= GiocatoriInAttesa – 5;
        }
   pthread_mutex_unlock(&P_mutex);
   // la partita è iniziata e il giocatore partecipa al gioco
   UscitaDalGioco==0;
   while (UscitaDalGioco==0) UscitaDalGioco= PartecipaAllaPartita ();
        // La funzione PartecipaAllPartita restituisce UscitaDalGioco=1 quando il giocatore e' eliminato
        // oppure quando è il  vincitore
   // epilogo del gioco
   pthread_mutex_lock(&P_mutex);
   GiocatoriAmmessi - -;
   if (GiocatoriAmmessi== 0)  {
        PartitaInCorso= 0;
        pthread_cond_signal(&AttesaGiocatori);
        / riattiva, se esiste, un giocatore sospeso, che ripeterà il test iniziale
        }
```

```
    pthread_mutex_unlock(&P_mutex);
    < fa altre cose >.
}
```

SOLUZIONE

```
Giocatore
{
    <fa altre cose>;
    // decide di giocare una partita: prologo per l'ammissione al gioco
    ……
    GiocatoriInAttesa++;
    while  (PartitaInCorso==1 or GiocatoriInAttesa <5)
    ……
    // il giocatore si sospende se è in corso una partita o, in caso contrario, se
    // non si è ancora costituito un gruppo di 5 giocatori in attesa di partecipare al gioco
    GiocatoriAmmessi++;
    if GiocatoriAmmessi<5
    ……
    // i giocatori ammessi riattivano a catena altri giocatori in attesa fino a costituire il gruppo di 5
    //  la variabile GiocatoriInAttesa è aggiornata dopo la riattivazione di tutti i 5 giocatori da ammettere
    else          {
          // GiocatoriAmmessi= 5
          GiocatoriInAttesa= GiocatoriInAttesa – 5; PartitaInCorso=1;
          }
    ……
    // la partita è iniziata e il giocatore partecipa al gioco
    UscitaDalGioco==0;
    while (UscitaDalGioco==0) UscitaDalGioco=PartecipaAllaPartita ();
          // La funzione PartecipaAllPartita restituisce UscitaDalGioco=1 quando il giocatore e' eliminato
          // oppure quando è il  vincitore
    // epilogo del gioco
    ……
    GiocatoriAmmessi - -;
    if (GiocatoriAmmessi== 0)  {
          PartitaInCorso= 0;
          ……
          / riattiva, se esiste, un giocatore sospeso, che ripeterà il test iniziale
          }
          ……
    < fa altre cose >.
}
```

ESERCIZIO A-3 (3 punti)

In un sistema vengono generati 5 processi (A,B,C,D,E), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

Processo	Durata	Tempo di arrivo
A	45	0
B	55	13
C	15	21
D	5	25
E	15	29

Tutti i processi avanzano senza mai sospendersi.

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) con la politica di scheduling Shortest Remaining Time First (SRTF) con prerilascio.

In particolare la politica di scheduling seleziona per l'esecuzione il processo con minor tempo residuo in esecuzione.

SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

SOLUZIONE

Tempo t=	Evento	Processo in esecuzione	Tempo residuo processo in esecuzione	Coda pronti (con tempo residuo di esecuzione)
0	Generato processo A	A	45	-
13	Generato processo B	A	32	B(55)
21	Generato processo C	C	15	A(24), B(55)
25	Generato processo D	D	5	C(11), A(24), B(55)
29	Generato processo E	D	1	C(11), E(15),A(24), B(55)
30	Termina D	C	11	E(15),A(24), B(55)
41	Termina C	E	15	A(24), B(55)
56	Termina E	A	24	B(55)
80	Termina A	B	55	-
135	Termina B	-	-	-

Di conseguenza:

Processo	Tempo di arrivo nel sistema	TERMINA ESECUZIONE	TEMPO DI PERMANENZA NEL SISTEMA
A	0	80	80
B	13	135	122
C	21	41	20
D	25	30	5
E	29	56	27

SOLUZIONE

Tempo t=	Evento	Processo in esecuzione	Tempo residuo processo in esecuzione	Coda pronti (con tempo residuo di esecuzione)
0	Generato processo A	A	45	-

Di conseguenza:

Processo	Tempo di arrivo nel sistema	TERMINA ESECUZIONE	TEMPO DI PERMANENZA NEL SISTEMA
A	0		
B	13		
C	21		
D	25		
E	29		

ESERCIZIO A-4 (2 punti)

In un sistema con risorse R1, R2 e R3 (tutte di molteplicità 2), sono presenti i processi P1, P2 e P3 che inizialmente non possiedono risorse e successivamente avanzano senza interagire reciprocamente e alternandosi nello stato di esecuzione con velocità arbitrarie.

Nel corso della propria esistenza, ciascun processo esegue una propria sequenza di richieste, che possono intercalarsi in modo arbitrario con quelle degli altri processi. Dopo aver ottenuto e utilizzato le risorse che richiede, ogni processo termina rilasciando tutte le risorse ottenute.

Si consideri la sequenza di richieste sotto riportate:

Processo	Prima richiesta	Seconda richiesta	Terza richiesta	Quarta richiesta	Terminazione
P1	1 istanza di R3	1 istanza di R2	1 istanza di R3	1 istanza di R1	Rilascia tutto
P2	1 istanza di R3	1 istanza di R2	1 istanza di R2	1 istanza di R1	Rilascia tutto
P3	1 istanza di R2	1 istanza di R1	1 istanza di R1	-	Rilascia tutto

Dire se i processi evitano la possibilità di stallo e motivare la risposta.

SOLUZIONE
I processi non evitano lo stallo perché richiedono le risorse in modo non ordinato.
Una sequenza che provoca stallo è la seguente:
P1-R3, P2-R3, P1-R2, P1-R3 (sospeso), P2-R2, P2-R2 (sospeso), P1-R2 (sospeso)
L'ordinamento delle risorse ai fini della politica di prevenzione dello stallo è: R3, R2, R1

SOLUZIONE
I processi evitano lo stallo [SI/NO]?_____

Se NO: Una sequenza che provoca stallo è la seguente:

_____

Se SI: si previene lo stallo perché:

_____

ESERCIZIO A-5 (2 punti)
In un sistema UNIX un processo genera un processo figlio eseguendo il seguente frammento di codice

```
printf("%s", "Hello ");
a=exec("prova");
a = fork( );
if (a>0)
        printf("%s", " Old Plain ");
else
        printf("%s", " Brave New ");
printf("%s", " World!\n");
```

Dire cosa stampano per effetto di questo codice il processo padre e il processo figlio.

SOLUZIONE

Processo padre:
stampa "Hello ", poi, se la exec ha successo, eseguirà il programma prova.
Altrimenti (la exec fallisce) ci sono due casi:

1) se la fork ha successo stampa "Old Plain World!",
2) altrimenti stampa solo "World!".

Processo figlio:
se la exec fallisce e la fork ha successo stampa " Brave New World!"
altrimenti non stampa niente

SOLUZIONE

Processo padre:

_____

_____

Processo figlio:

_____

_____

ESERCIZIO B-1 (4 punti)

Un sistema simile a Unix gestisce la memoria con segmentazione e caricamento in partizioni variabili e prevede lo swapping dei processi. La memoria virtuale di ogni processo è suddivisa in 2 segmenti, uno per il codice e uno per i dati e la pila. La memoria fisica, che ha la capacità di 35 Mbyte, riserva permanentemente al sistema operativo la partizione con origine 0 e lunghezza 3 e lascia i rimanenti 32 Mbyte disponibili per il caricamento dei processi.

Per l'assegnazione delle partizioni ai processi si adotta la politica *best fit.* In mancanza della partizione, o delle partizioni necessarie a un processo, si esegue lo *swap out* di uno o più segmenti, selezionati con la seguente politica:

- per i segmenti dati, si considera il tempo T trascorso dal caricamento;
- per i segmenti codice, si considera il tempo T trascorso dall'ultimo inserimento del relativo puntatore nella *Text Structure* di un processo;
- i segmenti vengono scaricati in sequenza, in ordine decrescente di *T;*
- a parità di *T,* tra un segmento codice e un segmento dati si seleziona il segmento codice.

Lo swap in avviene, con politica *FIFO*, appena la disponibilità di memoria lo consente.

Immediatamente prima del tempo *t=10* (notazione: *t=10 -)* sono presenti i seguenti processi (tutte le lunghezze sono espresse in *Mbyte*):

- il processo A, generato e caricato al tempo 1, che occupa:
  - la partizione A1 con origine 3 e lunghezza 4, riservata al codice;
  - la partizione A2 con origine 9 e lunghezza 4, riservata ai dati e alla pila;
- il processo B, generato e caricato al tempo 2, che occupa:
  - la partizione B1 con origine 13 e lunghezza 6, riservata al codice;
  - la partizione B2 con origine 25 e lunghezza 6, riservata ai dati e alla pila.

Al tempo t=10 il processo A esegue una *fork* generando il processo AA.

Al tempo t=11 il processo B esegue una *fork* generando il processo BB.

Al tempo t=12 il processo BB esegue una *exec* . Le esigenze di memoria sono di 4 per il codice e di 5 Mbyte, complessivi per i dati e la pila;

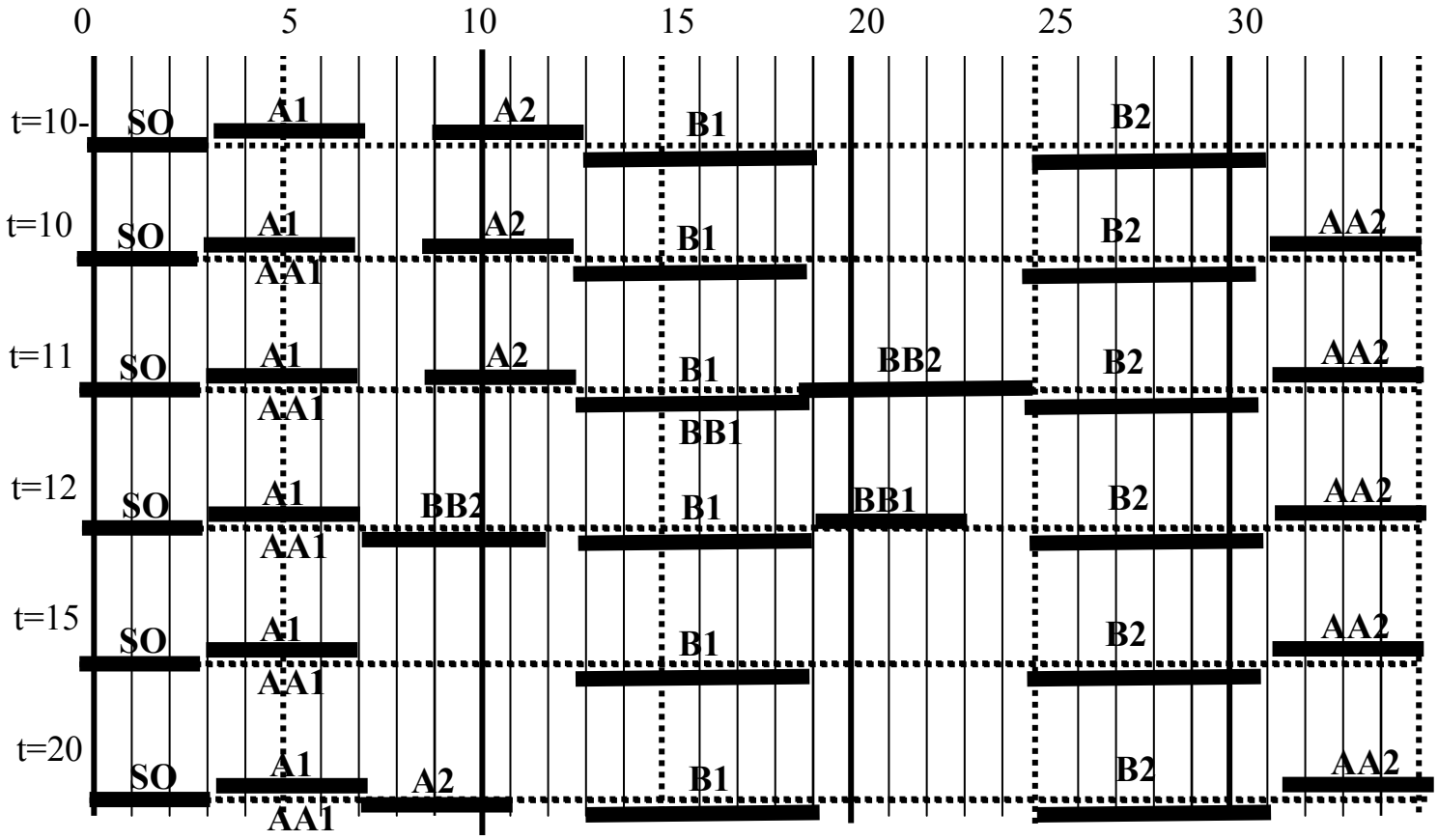
Al tempo t=15 il processo BB esegue una *exit.*

Al tempo t=20 il processo B esegue una *wait.*

Utilizzando il grafico sotto riportato, mostrare come evolve l'occupazione della memoria immediatamente dopo i tempi 10, 11, 12 e 20.

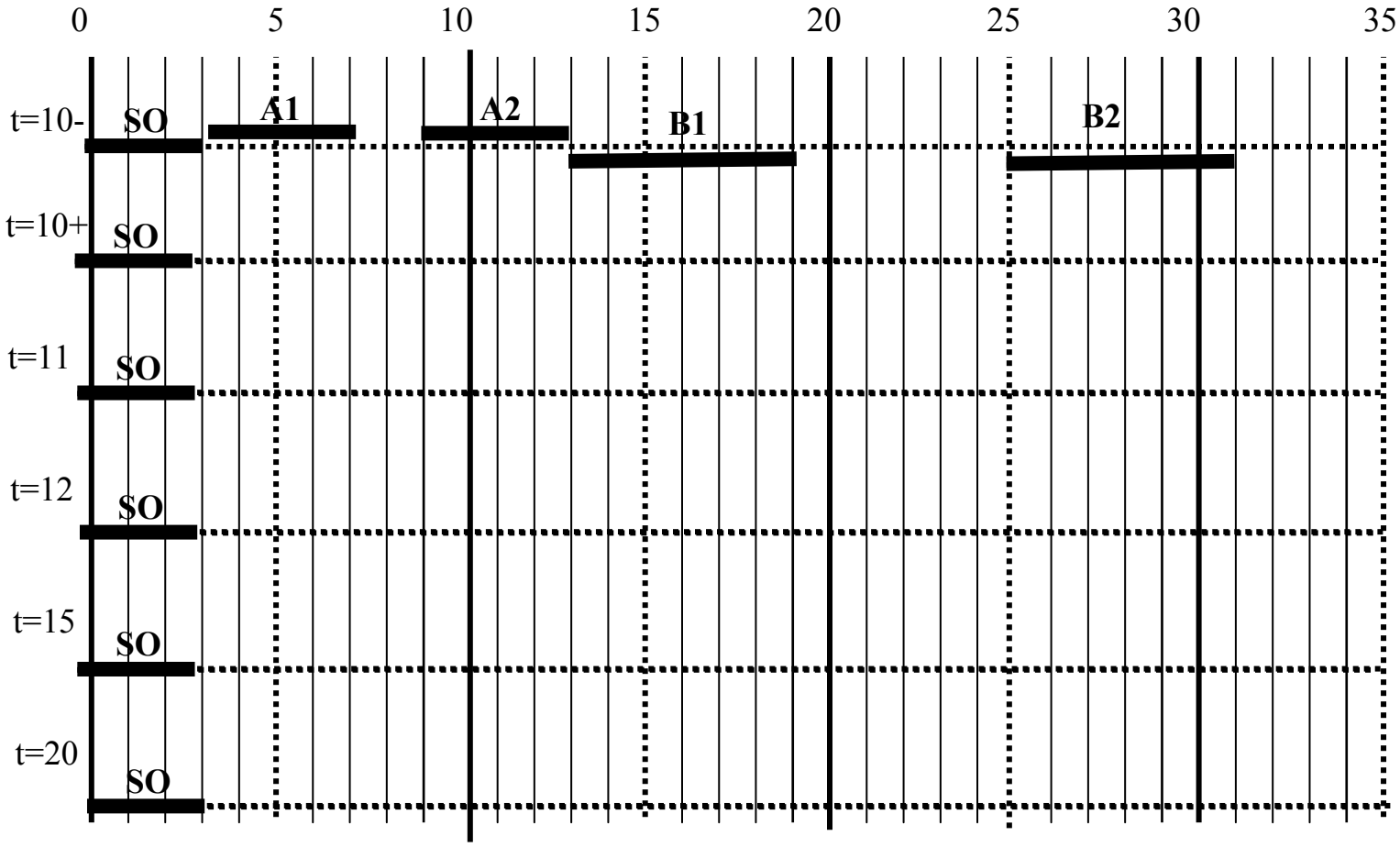
SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

SOLUZIONE



SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

SOLUZIONE



SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

ESERCIZIO B-2 (4 punti)

In un file system UNIX i blocchi del disco hanno ampiezza di 2Kbyte e i puntatori ai blocchi sono a 32 bit. Gli i-node contengono, oltre agli altri attributi, 4 indirizzi diretti e 3 indirizzi indiretti.
Si consideri il file rappresentato dal seguente i-node:

ind	0	1	2	3	4	5	6
Blocco fisico	101	103	105	107	109	108	106

Alcuni frammenti dei blocchi 106, 108, 109, 310, 311 e 200 sono riportati nel seguito.

Blocco fisico 106:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	200	201	202	203	204	205	206	207	208	209	…

Blocco fisico 108:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	311	312	313	314	315	316	317	308	309	310	…

Blocco fisico 109:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	421	422	423	424	425	426	427	418	419	420	…

Blocco fisico 310:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	531	532	533	535	535	536	537	538	539	520	…

Blocco fisico 311:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	646	647	648	649	640	641	642	643	644	645	…

Blocco fisico 200:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	725	726	727	728	729	720	721	722	723	724	…

Dire in quali blocchi fisici del disco sono contenuti i seguenti byte del file:

- Byte numero 4096
- Byte numero 7890
- Byte numero 12098
- Byte numero 16842
- Byte numero 28000
- Byte numero 1072000

SOLUZIONE

Indirizzo	BLOCCO LOGICO	BLOCCO FISICO	Raggiunto attraverso:			
			INDIRIZZO DIRETTO	BLOCCO INDIRETTO PRIMO LIVELLO	BLOCCO INDIRETTO SECONDO LIVELLO	BLOCCO INDIRETTO TERZO LIVELLO
4096	2	105	2	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
7890	3	107	3	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
12098	5	422		Blocco 109 Ind. nel blocco 1	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
16842	8	425		Blocco 109 Ind. nel blocco 4	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
28000	13	420		Blocco 109 Ind. nel blocco 9	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
1072000	523	643		Blocco 108 Ind. nel blocco 0	Blocco 311 Ind. nel blocco 7	Blocco ..... Ind. nel blocco.....

SOLUZIONE

Indirizzo	BLOCCO LOGICO	BLOCCO FISICO	Raggiunto attraverso:			
			INDIRIZZO DIRETTO	BLOCCO INDIRETTO PRIMO LIVELLO	BLOCCO INDIRETTO SECONDO LIVELLO	BLOCCO INDIRETTO TERZO LIVELLO
4096				Blocco	Blocco	Blocco .....

				Ind. nel blocco.....	Ind. nel blocco.....	Ind. nel blocco.....
7890				Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
12098				Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
16842				Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
28000				Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
1072000				Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....

ESERCIZIO B-3 (3 punti)

In un sistema UNIX il processo P esegue in sequenza le seguenti chiamate di sistema:
apre in lettura il file *F*, che contiene la seguente stringa di lunghezza 62:
`QUINTO_APPELLO:_CALENDARIO_DEGLI_ORALI-MODALITA'_DI_ISCRIZIONE`
esegue con successo una FORK che genera il figlio P1
esegue con successo una FORK che genera il figlio P2
esegue una WAIT immediatamente seguita da un'altra WAIT.
Dopo l'esecuzione della seconda WAIT avanzano i processi P1 e P2, intercallando le seguenti chiamate di sistema:

1. P1 esegue EXEC
2. P1 esegue READ(F, & buff 1_1, 16)
3. P2 esegue EXEC
4. P2 esegue READ(F, & buff 2_1, 23)
5. P2 esegue CLOSE F
6. P1 esegue READ(F, & buff 1_2, 23)
7. P2 esegue OPEN F
8. P2 esegue READ(F, & buff 2_2, 14)
9. P1 stampa la concatenazione dei contenuti di buff 1_1e buff 1_2;
10. P2 stampa la concatenazione dei contenuti di buff 2_1e buff 2_2.

Si chiede:

- il contenuto di buff 1_1 dopo l'operazione 2);
- il contenuto di buff 2_1 dopo l'operazione 4);
- il contenuto di buff 1_2 dopo l'operazione 6);
- il contenuto di buff 2_2 dopo l'operazione 8);
- la stringa stampata da P1 con l'operazione 9;
- la stringa stampata da P2 con l'operazione 10.

SOLUZIONE

* Dopo l'operazione 2) Buff 1_1 contiene: `QUINTO_APPELLO:_`

* Dopo l'operazione 4) Buff 2_1 contiene: `CALENDARIO_DEGLI_ORALI-`

* Dopo l'operazione 6) Buff 1_2 contiene: `MODALITA'_DI_ISCRIZIONE`

* Dopo l'operazione 8) Buff 2_2 contiene: `QUINTO_APPELLO`

* La stringa stampata da P1 con l'operazione 9 è: `QUINTO_APPELLO:_ MODALITA'_DI_ISCRIZIONE`

* La stringa stampata da P2 con l'operazione 10 è: `CALENDARIO_DEGLI_ORALI- QUINTO_APPELLO`

SOLUZIONE

* Dopo l'operazione 2) Buff 1_1 contiene:

* Dopo l'operazione 4) Buff 2_1 contiene:

* Dopo l'operazione 6) Buff 1_2 contiene:

* Dopo l'operazione 8) Buff 2_2 contiene:

* La stringa stampata da P1 con l'operazione 9 è:

* La stringa stampata da P2 con l'operazione 10 è:

ESERCIZIO B-4 (2 punti)
Data la seguente matrice di protezione:

	File1	File2	File3	Stampante	Scanner
Utente 20	R	W	R	W	R
Utente 21	RW		RW	W	
Gruppo 81	RW	WX	RWX		R
Gruppo 33	W			W	R

Convertirla nella corrispondente lista di controllo degli accessi

SOLUZIONE

File 1: <Utente 20:R>→<Utente 21:RW>→<Gruppo 81:RW>→<Gruppo 33:W>
File 2: <Utente 20:W>→<Gruppo 81:WX>
File 3: <Utente 20:R>→<Utente 21:RW>→<Gruppo 81:RWX>
Stampante: <Utente 20:W>→<Utente 21:W>→<Gruppo 33:W>
Scanner: <Utente 20:R>→<Gruppo 81:R>→<Gruppo 33:R>

SOLUZIONE

_____

_____

_____

_____

_____

ESERCIZIO B-5 (2 punti)
Un disco RAID di livello 1 è composto da 6 dischi fisici indipendenti, individuati dagli indici da 0 a 5. Ogni disco esegue i comandi pendenti in maniera indipendente dagli altri. I blocchi del disco virtuale hanno ampiezza uguale a quella dei blocchi fisici, e le strip coincidono con i blocchi.
I dischi 3, 4 e 5 sono ridondanti, e ogni loro settore contiene la copia dei settori omologhi dei dischi non ridondanti di indice 0, 1 e 2. Pertanto il blocco *b* del disco virtuale *V* è mappato nel blocco *b* div 3 dei dischi fisici di indice *b* mod 3 e *b* mod 3 + 3.
Quando un disco di indice *i* si guasta, il suo controllore rileva e notifica il guasto non appena riceve un comando di lettura o scrittura. Se il comando è di lettura, questo viene indirizzato al disco *i*+3 mod 6, e, in ogni caso (indipendentemente dal tipo di comando), viene avviato il processo di recovery del disco *i*.
Al tempo t il disco fisico di indice 2 è guasto e il disco virtuale V riceve un comando di scrittura che interessa i blocchi logici di indice 1516 e 6785. Per ognuna di queste operazioni si chiede:
1. L'indice dei dischi fisici e l'indice del blocco fisico sul quale è mappato il blocco virtuale.
2. Se l'operazione dà luogo ad una notifica di guasto.
3. Quante operazioni vengono eseguite sui dischi fisici.

SISTEMI OPERATIVI E LABORATORIO, CORSI A e B Prova del 2/7/2010 CORSO |A| |B|

SOLUZIONE

Blocco 1516:

1. È allocato sul blocco fisico 505 dei dischi fisici 1 e 4.
2. Dato che l'operazione non coinvolge il disco 2 non dà luogo una notifica di guasto
3. Vengono eseguite due operazioni di scrittura, sul blocco fisico 505 dei dischi fisici 1 e 4.

Blocco 6785:

1. È allocato sul blocco fisico 2261 dei dischi fisici 2 e 5.
2. Dato che l'operazione coinvolge il disco 2 dà luogo una notifica di guasto
3. Viene eseguita una sola operazione di scrittura, sul blocco fisico 2261 del disco fisico 5.

SOLUZIONE

Blocco 1516:

1. È allocato sui blocchi fisici: _____ dei dischi fisici _____.
2. dà luogo a una notifica di guasto [SI/NO]? _____
3. Vengono eseguite le seguenti operazioni:_____

Blocco 6785:

4. È allocato sui blocchi fisici: _____ dei dischi fisici _____.
5. dà luogo a una notifica di guasto [SI/NO]? _____
6. Vengono eseguite le seguenti operazioni:_____