

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2007/2008
CONCORRENZA - 13 Giugno 2008

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione prima di svolgere ogni altro esercizio.

Esercizio 1: Un santuario in Nepal e' alla fine di un ponte di corda. Il ponte e' l'unico modo per andare o tornare dal santuario. Sul ponte possono stare al massimo MAXPONTE persone e tutte devono procedere nello stesso senso. Il santuario contiene al massimo MAXSANTUARIO persone.

I fedeli sono infiniti e seguono il seguente algoritmo:

```
#define ANDATA 0
#define RITORNO 0
fedele: process [1..infinito]
    santuario.entraponte(ANDATA)
    ... attraversa il ponte ...
    santuario.esciponte(ANDATA)
    ... medita ...
    santuario.entraponte(RITORNO)
    ... attraversa il ponte ...
    santuario.esciponte(RITORNO)
```

Scrivere il monitor santuario con l'avvertenza non solo di seguire le specifiche ma di risolvere eventuali problemi di deadlock e starvation qualora ve ne fossero. (indicare nel caso quali siano e come sono stati risolti).

Esercizio 2: E' possibile implementare un meccanismo di entrata/uscita da sezione critica usando le seguenti operazioni atomiche?

- a) $F(X, Y) = \langle Z = \text{random}(2); X1 = Y \ \& \ Z; Y = X \ | \ Z; X = X1; \rangle$
b) $F(X, Y, Z) = \langle Z = \text{random}(2); Y = X; X = X \ \& \ Z \rangle$

Esercizio 3: Si considerino le seguenti primitive di message-passing semi-sincrono.

int sssend(id dest, msg msg, int tout); Invia il messaggio msg al destinatario dest. Il mittente viene sospeso in attesa della ricezione o del timeout tout. In caso di invio avvenuto con successo ritorna 0. In caso di timeout ritorna 1 e il destinatario non riceve il messaggio.

msg ssreceive(id snd); Riceve un messaggio dal mittente snd. Snd puo' essere -1 per ricevere da chiunque.

Rispondere alle seguenti domande:

- a) E' possibile implementare le primitive sssend() e ssreceive() basandosi solo sulle primitive completamente sincrone send() e receive()? Nota bene: I processi possono accedere in lettura al clock di sistema, ma non hanno alcuna primitiva per sospendersi per un tempo specificato. Il processo che fa la sssend() non deve fare attesa attiva.
- b) E' possibile implementare le primitive completamente sincrone send() e receive() basandosi solo sulle primitive sssend() e ssreceive()?
- c) E' possibile implementare la sleep(int n) (che sospende un processo per n secondi) basandosi solo sulle primitive sssend() e ssreceive(), evitando attesa attiva da parte del processo?

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
 CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2007/2008
 PARTE GENERALE – 13 Giugno 2008

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione prima di svolgere ogni altro esercizio.

Esercizio 1: Sia dato un buffer di 16K gestito con il metodo delle buddy list e unita' di allocazione minima 1K. Mostrare una sequenza di operazioni che porti ad avere 8K liberi frammentati in blocchi da 1K. Mostrare la storia dello stato della memoria a seguito di ogni operazione.

Esercizio 1bis: Un sistema operativo gestisce un file system basato su i-nodes mantenendo in memoria principale la tabella degli i-node (precaricata al boot) e la bitmap dei blocchi liberi. La tabella degli i-node ha spazio per 32 entry. L'i-node 0 descrive la directory radice. La dimensione di un blocco fisico e' di 1KB. La tabella degli i-node corrente e':

	inode[0]	inode[1]	inode[2]	inode[3]	inode[4]	inode[8]	inode[9]
size:	210	90	40	11	45	1094	1024
type:	d	d	d	l	d	f	f
...							
direct[0]:	3	20	6	31	8	14	1
direct[1]:	0	0	0	0	0	16	0
direct[2]:	0	0	0	0	0	0	0
direct[3]:	0	0	0	0	0	0	0
indirect[0]:	0	0	0	0	0	0	0

La bitmap dei blocchi liberi e': 0x801F69CB (il bit meno significativo rappresenta il blocco 0)

Parte del contenuto corrente del disco e':

block[1]	block[3]	block[6]	block[8]	block[14]	block[16]	block[20]	block[31]
000000	. 0	. 2	. 4	0110010	0110010	. 1	/local/bin
111111	.. 0	.. 0	.. 2	1001000	0110010	.. 0	
000000	bin 3	bin 4	a.out 8	
111111	local 2						
000000	etc 1						
....	big 9						

Mostrare I cambiamenti al contenuto del disco, alla bitmap dei blocchi liberi e alla tabella degli i-node apportati dall'esecuzione di ognuno di questi comandi:

- a) ln /bin/a.out /etc/
- b) rm /bin/a.out
- c) cat "ciao" > /bin/a.out
- c) d) for n in seq 5; do cat /big >> /huge; done
- ci) e) ln -s /etc/a.out /etc/b.out
- cii) f) rm /etc/a.out

Esercizio 2: Sia dato l'algoritmo di rimpiazzamento MOD che in caso di page fault sostituisca il frame n%size, dove n e' l'indice del riferimento corrente nella stringa dei riferimenti in memoria e size sia il numero di frame. (suggerimento per chiarezza: contate i riferimenti e i frame a partire da 0).

E' un algoritmo a stack? Fornire una dimostrazione a supporto della risposta. Verra' valutata la semplicita' e compattezza della dimostrazione.

Esercizio 3: Sia x l'ultima e y la penultima cifra del vostro numero di matricola. Descrivere con quali componenti hardware e con quali altri componenti del sistema operativo interagisce direttamente o indirettamente il componente $(y*10+x)\%4$ del sistema operativo. Dettagliare quali sono le chiamate o gli eventi che scatenano tali interazioni.

- 0. Disk manager
- 1. File system manager

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.

2. Scheduler
3. Memory manager



