Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ___ Col _____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2010/2011 - PROVA DI CONCORRENZA

25 luglio 2011

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

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione in tutti i fogli prima di svolgere ogni altro esercizio. Scrivere esclusivamente a penna senza abrasioni. E' vietato l'uso delle penne cancellabili, della matita, dei coprenti bianchi per la correzione (bianchetto) e la scrittura in colore rosso (riservato alla correzione).
Il compito e' formato da due fogli, quattro facciate compresa questa. Le soluzioni che si vogliono sottoporre per la correzione devono essere scritte negli spazi bianchi di questi fogli. Non verranno corretti altri supporti.
E' obbligatorio consegnare il compito, e' possibile chiedere che esso non venga valutato scrivendo “NON VALUTARE” in modo ben visibile nella prima facciata.

Esercizio 1: Il monitor da implementare deve fornire tre procedure entry:

procedure entry void put(T value);

procedure entry T get_from_head();

procedure entry T get_from_tail();

I valori che i processi passano come parametro alla chiamata *put* devono essere letti da altri processi con le chiamate *get*, in ordine fifo (*get_from_head*) o lifo (*get_from_tail*). In altre parole *get_from_head* legge il valore indicato dalla prima chiamata *put* che non sia ancora stato letto, la *get_from_tail* l'ultimo valore usato come parametro di una *put* che non sia stato ancora letto.

Il tipo generico “T” richiede molta memoria, quindi si chiede di non creare buffer di elementi di tipo T.

La chiamata *put* e' bloccante fino a quando un processo non legga il valore passato come parametro (cioe' quando l'altro processe legge con una *get*, *put* si sblocca).

Ovviamente quando un processo chiama una delle due *get* e non ci sono processi in attesa di completare la *put*, il processo che richiede la *get* deve attendere. Se vi sono piu' processi in attesa di fare la *get* (*from head* or *from tail*), vengono riattivati in ordine fifo man mano che arrivano processi a fare richieste di tipi *put*.

Esercizio 2: siano dati tre processi "stampa" come segue:

```
stampa(x):(x=A,T,R) process
while (true) {
    synchro(x);
    print(x);
}
```

scrivere la fuzione synchro() facendo uso di semafori che consenta di avere in output una sequenza infinita di TARATATA (i.e. TARATATATARATATATARATATATARATATATARATATA.....)

Esercizio 3: Dato un servizio di message passing asincrono (NON completamente asincrono) e senza fare uso di un server implementare un servizio di message passing sincrono LIFO. (cioe' dati asend/arecv, implementare lssend lsrecv)
I messaggi possono essere di grandi dimensioni, si chiede quindi che l'implementazione non crei una coda con tutti i messaggi in attesa di essere ricevuti.

Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ___ Col _____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2010/2011 – PARTE GENERALE
17 giugno 2011

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

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione in tutti i fogli prima di svolgere ogni altro esercizio. Scrivere esclusivamente a penna senza abrasioni. E' vietato l'uso delle penne cancellabili, della matita, dei coprenti bianchi per la correzione (bianchetto) e la scrittura in colore rosso (riservato alla correzione).
Il compito e' formato da due fogli, quattro facciate compresa questa. Le soluzioni che si vuole sottoporre per la correzione devono essere scritte negli spazi bianchi di questi fogli. Non verranno corretti altri supporti.
E' obbligatorio consegnare il compito, e' possibile chiedere che esso non venga valutato scrivendo “NON VALUTARE” in modo ben visibile nella prima facciata.
Per svolgere questo compito occorre solo una penna e un documento di identità valido. La consultazione o anche solo la disponibilità di altro materiale comporterà l'annullamento del compito (con la stessa penalizzazione di punteggio della grave insufficienza per la prossima esercitazione scritta).

Esercizio 1: Il seguente File System di tipo UNIX contiene molteplici errori. Si chiede al candidato di trovare gli errori e di correggerli come farebbe il programma fsck (file system check) (blocco da 1k, 16 i-node, 16 blocchi).

I-node[0](radice): size=60;link=6;type='d';direct[0]=4

I-node[1]: size=44;link=1;type='l';direct[0]=7

I-node[2]: size=28;link=2;type='d';direct[0]=0

I-node[3]: size=1026;link=3;type='f';direct[0]=15,direct[1]=5

I-node[6]: size=2050;link=1;type='f';direct[0]=15,direct[2]=1

I-node[8]: size=16;link=2;type='d';direct[0]=3

I-node[11]: size=44;link=4;type='d';direct[0]=14

I-node[12]: size=32;link=2;type='f';direct[0]=9

data[0]={(".",2),("..",0),("passwd",12)}

data[1]="cc"

data[3]={(".",8),("..",0)}

data[4]={(".",0),("..",0),("boot",6),("lost+found",8),("etc",2),("tmp",11)}

data[5]="dd"

data[7]="/etc/passwd"

data[9]="root:x:0:0:root:/root:/bin/bash\n"

data[12]="aaaa....." (1024 volte a)

data[14]={(".",11),("..",0),("data",1),("p2",12)}

data[15]="bbbb....." (1024 volte b)

bit-map i-node liberi: 0xeab2 (il bit meno significativo indica l'i-node 0)

bitmap blocchi liberi: 0x2d62(il bit meno significativo indica il blocco 0)

Esercizio 2: Sia dato un algoritmo di rimpiazzamento MAXPAGE che sostituisce sempre la pagina di numero maggiore.

a) Mostrare una stringa infinita di riferimenti (relativi ad un insieme finito di pagine) nella quale MAXPAGE si comporti come MIN con una memoria di 3 frame e che generi infiniti page fault.

b) L'algoritmo e' a stack?

Esercizio 3: Detti rispettivamente R il numero di riga e C il numero di colonna, soddisfare la richiesta corrispondente a (R+C)%5 (esercizio obbligatorio):

0) Quali sono le caratteristiche di un kernel per sistemi multiprocessore. Cosa significa che un kernel e' di tipo SMP (symmetric multi processor).
1) Cosa e' un microkernel e quale e' la differenza con un kernel monolitico
2) Cosa e' una macchina virtuale e in quali casi si utilizza.
3) Cosa e' un modulo del kernel e in quali casi viene utilizzato.
4) Cosa significa generare (compilare) un kernel e a cosa serve generare kernel specifici

NOTA: tutti gli esercizi verranno valutati solo se le risposte saranno corredate da motivazioni e dimostrazioni scritte in Italiano o in Inglese (corretto o almeno comprensibile). La presenza nella soluzione di un esercizio di solo codice sorgente/tabelle/scarabocchi/simboli vari comporta la non valutazione dell'esercizio.