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

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
PROVA SCRITTA DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2022/2023
11 settembre 2023

Esercizio -1: Essere iscritti su AlmaEsami 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 tre fogli, sei 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.
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 (verrà automaticamente valutato gravemente insufficiente).

Esercizio c.1: Il monitor AB gestisce l'accodamento limitato di due tipi di dati A e B. Il monitor prevede 3 procedure entry:
`void AB.add2a(int a0, int a1)`
`void AB.addb(int b0)`
`void AB.geta2b(int *a0, int *b0, int *b1).`
`add2a` aggiunge 2 elementi di tipo A, `addb` aggiunge un elemento di tipo B, `geta2b` restituisce un elemento di tipo A e due di tipo B.
Il monitor può memorizzare al massimo MAX elementi di tipo A e MAX di tipo B. (MAX >= 2).
Se non sono disponibili almeno un elemento di tipo A e due di tipo B la funzione `geta2b` deve attendere.
Gli elementi devono essere restituiti in ordine FIFO così come le richieste pendenti per geta2b devono essere esaudite in ordine FIFO.

Esercizio c.2: Un servizio di message passing asincrono esteso (XAMP) fornisce 4 chiamate:

- `void xsend(msg_t msg, pid_t pid)`: spedisce il messaggio `msg` al processo `pid`
- `msg_t xrecv(pid_t pid)`: riceve un messaggio da `pid` (o da chiuque se `pid==0`). E' una receive completamente asincrona: se non c'è nessun messaggio da `pid` (o da chiunque se `pid==0`), xrecv restituisce NULL, non attende.
- `int xcount(pid_t pid)`: ritorna il numero di messaggi in attesa di essere ricevuti provenienti da `pid` (o da chiunque se `pid==0`).
- `void xexpect(pid_t pid)`: attende che vi sia almeno un messaggio in attesa di essere ricevuto proveniente da `pid` (o da chiunque se `pid==0`). Se un tale elemento esiste al momento della chiamata, `xexpect` non è bloccante.

Il servizio XAMP ha lo stesso potere espressivo del message passing asincrono? (quello definito a lezione).

Esercizio g.1: Sia dato un sistema con uno spazio indirizzabile di 4096 byte (12 bit) e la ampiezza della pagina gestita dalla MMU sia configurabile in due modi diversi: 8 bit (pagine di 256 byte) o 9 bit (pagine di 512 byte).
La memoria fisica installata è di 4096 byte e viene gestita con un meccanismo di memoria virtuale tramite paginazione a richiesta. Il sistema operativo occupa gli indirizzi da 0 a 2999 (in esadecimale BB9). Gli indirizzi (FISICI) liberi da suddividere in frame per la paginazione vanno quindi da 3000 (BBA) a 4095 (FFF).
Quella che segue e' la sequenza degli indirizzi (LOGICI) ai quali un processo accede durante la propria esecuzione. Gli indirizzi sono espressi in forma esadecimale (per comodità di calcolo).
`112 0ab 221 322 432 **011** 1aa 3bb 2cc 444.`
Tutti gli accessi sono in lettura tranne il sesto (**011**, evidenziato in grassetto) che è un accesso in scrittura.
a) generare la lista dei riferimenti nell'ipotesi che le pagine siano di 256 byte.
b) Calcolare lo stato della memoria dopo ogni accesso in memoria e contare i page fault usando l'algoritmo di rimpiazzamento FIFO (nel caso di pagine a 256 byte)
c) generare la lista dei riferimenti nell'ipotesi che le pagine siano di 512 byte.
d) calcolare lo stato della memoria dopo ogni accesso in memoria e contare i page fault usando l'algoritmo di rimpiazzamento FIFO (nel caso di pagine a 512 byte).
e) calcolare i byte scambiati con la memoria secondaria nel caso b e nel caso d.

Esercizio g.2: rispondere alle seguenti domande (*motivando opportunamente le risposte!*):
a) Perché è necessario l'algoritmo della media esponenziale nel calcolo dello scheduler shortest job first? La media esponenziale dipende da un parametro (normalmente indicato con alpha, di valore fra zero e uno). A cosa serve questo parametro?
b) Dato un array di 4 dischi di uguale dimensioni confrontare l'uso di raid10 e di raid6. Qual è la capacità utilizzabile nei due casi? Qual è il livello di fault tolerance? Quali sono le prestazioni?
c) Perché i metodi di sincronizzazione tipo test&set (detti anche spinlock) assumono grande rilevanza nella scrittura di sistemi operativi multiprocessore?
d) Il file system ext2 è più efficiente nell'accesso diretto a file di piccole dimensioni rispetto a quelli di grandi dimensioni. Perché?