SISTEMI OPERATIVI
4 luglio 2012

Cognome:_____ **Nome:**_____
Matricola:_____

1. Ricordate che non potete usare calcolatrici o materiale didattico.
2. Ricordate che potete consegnare al massimo tre prove scritte per anno accademico

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (5 punti)

Tre processi P_A, P_B e P_C eseguono il seguente codice:

Shared **Var** semaphore mutex = 1; (valore iniziale)
semaphore done = 0; (valore iniziale)

P_A:	P_B:	P_C:
repeat forever:	**repeat forever**:	**repeat forever**:
wait(mutex)	**wait**(done)	**wait**(done)
<A>	**wait**(mutex)	**wait**(done)
signal(mutex)	<B>	**wait**(mutex)
signal(done)	**signal**(mutex)	<C>
	signal(done)	**signal**(mutex)

a1)
L'esecuzione concorrente di P_A, P_B e P_C produce una sequenza (di lunghezza indefinita) di chiamate alle procedure A, B e C. Quali delle sequenze qui sotto riportate possono essere la porzione iniziale di sequenze prodotte dall'esecuzione concorrente di P_A, P_B e P_C? (marcate le sequenze che scegliete con una croce nello spazio apposito)
A2)
In ciascuna delle sequenze restanti, che non possono essere prodotte dall'esecuzione concorrente dei tre processi, cerchiate/sottolineate la lettera che rappresenta l'ultima procedura che può effettivamente essere eseguita nell'esecuzione concorrente di P_A, P_B e P_C

1. [X] A,A,B,C,A,A,C,A,A …
2. [ ] A,B,A,C,A,A,**C**,B,A …
3. [ ] A,B,**B**,C,A,A,C,B,A…
4. [X] A,B,A,A,C,A,B,B,C …

b)
Riportate lo pseudocodice della prima "soluzione" al "*problema dei 5 filosofi*" vista a lezione.

filosofo *i*:

```
do{
        wait(bacchetta[i])
        wait(bacchetta[i+1 mod 5])
                …
                mangia
                …
        signal(bacchetta[i]);
        signal(bacchetta[i+1 mod 5]);
                …
                pensa
                …
        }while (true)
```

semaphore *bacchetta*[*i*] = 1;

c) Perché la soluzione riportata al punto b) non è esente da deadlock?

Perché se tutti i filosofi riescono a prendere ciascuno una sola delle due bacchette (*bacchetta[i]*) si crea un'attesa circolare.

d)
All'interno di un sistema operativo, un certo processo P è correntemente in stato di "Ready to Run", e si sa che, una acquisita la CPU non dovrà più rilasciarla volontariamente prima di aver terminato la propria esecuzione (in altre parole, non deve più eseguire operazioni di I/O, di sincronizzazione o di comunicazione con altri processi).
Quale/quali, tra gli algoritmi di scheduling **FCFS**, **SJF preemptive, SJF non-preemptive**, **round robin** garantisce/garantiscono che il processo P riuscirà a portare a termine la propria computazione? (motivate la vostra risposta, assumendo che SJF possa effettivamente essere implementato)

FCFS, e round robin. Infatti, nel caso di SJF preemptive e non, potrebbe sempre arrivare in coda di ready un processo che deve usare la CPU per un tempo minore di quanto rimane da eseguire a P.

e) Se invece P si trova già nello stato "Running", quali degli algoritmi sopra riportati garantiscono la terminazione di P?

FCFS, round robin e SJF non-preemtpive.

ESERCIZIO 2 (5 punti)

In un SO la tabella delle pagine può contenere al massimo 2048 (decimale) entry, e l'offset massimo all'interno di una pagina è FFFF (esadecimale).

a) Il SO potrebbe dover adottare un sistema di paginazione a due livelli (motivate la vostra risposta)?

No. Dobbiamo implementare una paginazione a due (o più) livelli se la tabella delle pagine più grossa del sistema non stà in un solo frame. Nel nostro caso, un frame grande 2^{16} byte che contenga 2048 entry permette di assegnare a ciascuna entry 32 byte per scrivere il numero di un frame, uno spazio più che

sufficiente per un qualsiasi sistema reale (32 byte = 256 bit, per cui possono essere indirizzati fino a 2^{256} frame della memoria fisica, ben al di la delle dimensioni della memoria fisica dei sistemi attuali)

b) Quale dimensione può avere lo spazio di indirizzamento fisico di questo sistema per poter dire con certezza che il sistema deve implementare la memoria virtuale?

Lo spazio di indirizzamento fisico deve essere inferiore a quello logico. Poiché lo spazio di indirizzamento logico è di $2^{11} * 2^{16} = 2^{27}$ byte, lo spazio di indirizzamento fisico dovrà essere al massimo grande 2^{26} byte (in realtà anche con uno spazio fisico di 2^{26} byte sarebbe necessario implementare la memoria virtuale, in quanto una parte della RAM sarebbe comunque occupata dal SO)

c) Descrivete brevemente un vantaggio e uno svantaggio del binding dinamico degli indirizzi:

Vantaggio: è possibile spostare il codice dei programmi in esecuzione da una parte all'altra della RAM senza dover ricompilare o ricaricare il codice stesso

Svantaggio: il calcolo degli indrizzi fisici a partire da quelli logici viene fatto a run time, e rallenta l'esecuzione dei programmi (anche in presenza di hardware specifico per limitare l'overhead computazionale)

d1) Un sistema che implementa la memoria virtuale e che adotta una politica di rimpiazzamento delle pagine locale può sofrire di trashing? (motivate la risposta)
d2) Un sistema che implementa la memoria virtuale e che ha uno spazio di indirizzamento fisico maggiore di quello logico può soffrire di trashing? (motivate la risposta)

d1) Si. Se ci sono troppo processi nel sistema, ciascuno avrà pochi frame a disposizione e produrrà continuamente dei page fault, sostituendo una propria pagina che sarà molto probabilmente riferita nell'immediato futuro.
d2) Si. Se un sistema implementa la memoria virtuale può soffrire di trashing indipendentemente dalla dimensione degli spazi di indirizzamento logici e fisici.

e)
Perché l'uso delle librerie dinamiche è preferibile all'uso delle librerie statiche?

Perché permettono di risparmiare spazio in RAM. Infatti, una sola copia della libreria può essere condivisa da tutti i programmi che la usano, e la libreria viene caricata in RAM solo se il programma che la usa chiama una delle subroutine della libreria stessa. (di fatto viene risparmiato anche spazio sull'hard disk).

ESERCIZIO 3 (4 punti)

a) Dite a cosa serve e descrivete brevemente la FAT.
(Se preferite, potete rispondere alla domanda con un esempio, disegnando la FAT relativa ad un hard disk da 16 blocchi che contiene un unico file memorizzato nei blocchi 9, 3, 5, 13. Indicate quali assunzioni fate per indicare i blocchi vuoti e i blocchi di fine file)

E' un array con un numero di entry pari al numero di blocchi dell'hard disk, e permette un'implementazione efficiente dell'allocazione concatenata dei file: se X e Y sono due blocchi appartenenti al file F, con il blocco X che punta al blocco Y, allora nella entry X della FAT c'è scritto il

numero Y. Se Z è l'ultimo blocco del file, nella entry Z della FAT viene scritto un marker di fine file. Se K è un blocco libero dell'hard disk, nella entry K della FAT viene scritto un marker di "blocco libero"

b) Una cartella unix DIR ha un link counter pari a 5. Che informazioni ci da questo valore, e come viene ricavata questa informazione?

Ci dice che DIR contiene 3 sottocartelle. Questo valore viene ricavato dal fatto che quando DIR è stata creata, il suo link conunter valeva 2 (il nome DIR all'interno della directory padre di DIR, e "." All'interno di DIR). Ogni qual volta viene creata una nuova sottocartella di DIR, il link counter di DIR viene incrementato di uno, a causa dell'entry ".." all'interno della sottocartella appena creata.

c) In una cartella A di un sistema Unix viene creato un nuovo file di testo di nome *pippo*. Assumendo che la working directory sia A, indicate i comandi necessari per eseguire le seguenti operazioni (assumendo che ogni comando tiene conto di quelli eseguiti prima di lui)

1) creare un nuovo hard link di nome *pluto* a *pippo*: ln pippo pluto
2) creare un nuovo hard link di nome *topolino* a *pluto*: ln pluto topolino
3) creare un nuovo symbolic link di nome *minnie* a *topolino*: ln –s topolino minnie
4) copiare *pluto* in *paperino (paperino* prima non esisteva*)*: cp pluto paperino
5) rimuovere pippo: rm pippo
6) qual è il valore del link counter di *topolino* dopo il comando 3)? 3
7) qual è il valore del link counter di *pluto* dopo il comando 4)? 3