SISTEMI OPERATIVI – 15 luglio 2019 corso A

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

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

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (7 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(done)	**wait**(done)	**wait**(mutex)
wait(mutex)	**wait**(mutex)	<C>
<A>	<B>	**signal**(mutex)
signal(mutex)	**signal**(mutex)	**signal**(done)
	signal(done)	

a)
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.
a1: Quale/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 la/le sequenza/e che scegliete con una croce nello spazio apposito)
a2: Qual è il valore della variabile semaforica "done" immediatamente dopo la terminazione della/delle sequenze che avete selezionato nella risposta precedente?

1. [X] C,B,A,C,C,B,A,C,A done = 1
2. [ ] C,C,C,A,B,A,B,A,B done = ???
3. [X] C,A,C,B,A,C,C,A,A done = 0
4. [ ] C,C,B,A,A,C,A,A,C done = ???

b) Elencate almeno quattro ragioni per cui, in un moderno sistema operativo time sharing che implementa la memoria virtuale si può verificare un context switch tra due processi utente:

scade il quanto di tempo del processo running
entra in coda di ready un processo con priorità maggiore di quello in stato running
processo running esegue una operazione di I/O
processo running esegue una wait e si addormenta sul semaforo
processo running genera una trap e viene terminato
processo running genera page fault

processo running termina di eseguire il suo codice (in tutti i casi eccetto il secondo, se c'è almeno un altro processo utente in RQ)

c) Che cosa vuol dire che un algoritmo di schedulino soffre di starvation?

Che non garantisce che, in un tempo finito, un processo in coda di ready venga selezionato per entrare in esecuzione e terminare.

d) descrivete brevemente le tre proprietà che deve possedere una corretta soluzione al problema della sezione critica.

Si vedano i lucidi della sezione 6.2

e) Che vantaggio da l'uso dei thread al posto dei processi?

Un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, il context switch tra peer thread e la creazione di un nuovo peer thread richiedono molto meno tempo delle corrispondenti operazioni sui processi.

ESERCIZIO 2 (7 punti)

In un sistema, la tabella delle pagine più grande occupa 8 Mbyte, equivalenti a 1024 frame, e un indirizzo fisico è scritto su 29 bit.

a) Quanto è grande lo spazio di indirizzamento logico del sistema? (motivate numericamente la vostra risposta)

Un frame/pagina è grande $2^{(23-10)} = 2^{13} = 8192$ byte, e quindi nel sistema ci sono (29-13) 2^{16} frame, per cui sono necessari 2 byte per scrivere il numero di un frame. Dunque la tabella delle pagine più grande del sistema ha $2^{23}/2$ entry, pari al numero di pagine dello spazio logico, e quindi lo spazio di indirizzamento logico è grande $2^{(22+13)} = 2^{35}$ byte.

b) Assumendo che lo spazio di indirizzamento logico e fisico del sistema non cambino, quale dimensione minima dovrebbero avere le pagine di questo sistema per essere certi di non dover ricorrere ad una paginazione a più livelli?

Poniamo 35 = m + n (m = bit usati per scrivere un numero di pagina, n = bit usati per scrivere l'offset).
Allora il numero di entry della PT più grande (cioè 2^m entry), moltiplicato per la dimensione di una entry (cioè 16 bit, o due byte) deve poter essere contenuto in una pagina/frame, ossia: $2^m \cdot 2^1 \leq 2^n$
Da cui: $m + 1 \leq n$. Poiché $m = 35 - n$, risolvendo il semplice sistema si ha $n \geq 18$, ossia le pagine devono almeno essere grandi $2^{18} = 256$ Kbyte.

c) Come funziona e **perché si usa** una inverted page table? (se vi sembra utile, usate anche un disegno)

si veda la sezione 8.5.3 dei lucidi usati a lezione.

d) perché i sistemi operativi moderni non usano l'allocazione contigua dello spazio in RAM a partizioni fisse o a partizioni variabili?

Non usano l'allocazione a partizioni fisse perché limita a priori il grado di programmazione e soffre in modo eccessivo del problema della frammentazione interna. Non usano l'allocazione a partizioni variabili

perché soffre del problema della frammentazione esterna e costringe periodicamente al ricompattamento dello spazio in RAM.

ESERCIZIO 3 (6 punti)

a) In Unix che cosa succede nelle strutture interne al sistema, quando viene creato un nuovo link fisico ad un file A già esistente?

Viene semplicemente incrementato di 1 il link counter dell'index-node associato al file A. Una entry viene aggiunta nella cartella in cui è stato creato il nuovo link fisico.

b) perché sono necessari i link simbolici?

Per permettere di costruire collegamenti fra le directory, tra le quali non sono ammessi i link fisici (in Unix, i link simbolici permettono anche link fra file che stanno su partizioni diverse degli hard disk).

c) In quale caso, e perché, l'allocazione indicizzata dello spazio in memoria secondaria è particolarmente **svantaggiosa**?

Nel caso di file molto piccoli, che ad esempio occupano solo un blocco, perché per tenere traccia di quel blocco si preca quasi completamente il blocco indice, che va comunque allocato.

d) In quali casi, rispettivamente, è meglio usare un sistema RAID nella configurazione 0, 1 o 5?

Usiamo il RAID 0 se abbiamo bisogno di massimizzare lo spazio di memorizzazione disponibile e la velocità di accesso ai dati, mentre l'affidabilità non è un requisito fondamentale.
Usiamo il RAID 1 se abbiamo bisogno della massima affidabilità e velocità di accesso ai dati.
Usiamo il RAID 5 se vogliamo un ragionevole compromesso tra affidabilità, velocità di accesso ai dati, e spazio di memorizzazione disponibile.

e) dove è memorizzato il pathname assoluto di un file?

Da nessuna parte.

f) In quale caso l'accesso in lettura ad un file memorizzato su un sistema RAID non è più veloce che se il file fosse memorizzato su un normale hard disk?

Quando il file è memorizzato su uno più blocchi appartenenti a strip contenuti sullo stesso disco del RAID (e il RAID usato non è di tipo 1, poiché in questo caso, se il file è memorizzato su almeno due strip, si può sfruttare il disco di mirroring)