SISTEMI OPERATIVI – CORSO A
14 giugno 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)

a) Si consideri il problema dei produttori e consumatori, con buffer limitato ad *m* elementi, dove i codici del generico produttore e del generico consumatore sono i seguenti:

semafori necessari con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore full = 0;
semaphore empty = m;

"consumatore"	*"produttore"*
repeat	**repeat**
	<produci dato>
wait(full)	*wait(empty)*
wait(mutex)	*wait(mutex)*
<preleva dato dal buffer>	
	<inserisci dato nel buffer>
signal(mutex)	*signal(mutex)*
signal(empty)	*signal(full)*
<consuma dato>	
forever	**forever**

Inserite le opportune operazioni di wait e signal necessarie per il corretto funzionamento del sistema, indicando anche i semafori necessari ed il loro valore di inizializzazione.

b) Riportate lo pseudocodice che descrive l'implementazione dell'operazione di Signal. In che modo l'uso combinato di Wait e Signal evita lo spreco di tempo di CPU che è invece tipico dei meccanismi di sincronizzazione basati su busy waiting?

Si vedano i lucidi della sezione 6.5.2

c) Come è possibile implementare le sezioni critiche contenute nel codice della Signal? (spiegate perché la soluzione che avete indicato è accettabile)

1. Usando il busy waiting (perché, essendo le sezione critiche da implementare molto corte, l'uso del busy waiting produce uno spreco molto limitato di tempo di CPU)

2. Usando la disabilitazione degli interrupt (perché questa verrebbe comunque fatta sotto il controllo di codice del Sistema Operativo, e non di codice utente)

d) Riportate un semplice esempio in pseudo-codice di due processi concorrenti che usano uno o più semafori per sincronizzarsi e che, *a seconda dell'ordine relativo in cui vengono eseguite le istruzioni dei due processi, può sia funzionare correttamente che portare in una situazione di deadlock*. Indicate anche come devono essere inizializzati i semafori che usate.

P1	P2
wait(mutex1)	wait(mutex2)
wait(mutex2)	wait(mutex1)
sez. critica	sez. critica
signal(mutex2)	signal(mutex1)
signal(mutex1)	signal(mutex2)

semaphore mutex1 = 1; semaphore mutex2 = 1;

ESERCIZIO 2 (5 punti)

Un sistema con memoria paginata usa un TLB con un hit-ratio del 90%, e un tempo di accesso di 10 nanosecondi. Un accesso in RAM richiede invece 0,09 microsecondi.

a) Qual è, in nanosecondi, il tempo medio di accesso in RAM (esplicitate i calcoli che fate)?

Tmedio = 0,90 * (90+10) + 0,10 * (2*90 + 10) = 90 + 19 = 109 nanosecondi

b) Il sistema viene ora dotato di memoria virtuale, usando come algoritmo di rimpiazzamento quello della *seconda chance migliorato*.

E' possibile che un processo del sistema si veda aumentare il numero di frame che usa e contemporaneamente aumenta anche il numero di page fault generati dal processo?

Si, perché nel caso peggiore l'algoritmo si comporta come FIFO, che soffre dell'anomalia di Belady.

c) Indicate tutte le informazioni esplicitamente contenute in ciascuna entry di una page table di un sistema che usa l'algoritmo della seconda chance migliorato.

Numero di un frame, bit di validità, bit di riferimento, dirty bit.

d) E' corretto dire che in un sistema che implementa la memoria virtuale, i processi partono più velocemente (in media)?

Si, perché un processo può partire anche se non tutto il suo codice e i dati sono stati caricati in RAM.

e) Descrivete brevemente come avviene la prevenzione del thrashing nel sistema Solaris.

Ad intervalli regolari Solaris verifica se il numero di frame liberi è sceso sotto una soglia predefinita lostfree. Se si, si attiva il processo pageout che opera in due fasi. Nella prima fase azzera i bit di

riferimento di tutte le pagine in RAM. Nella seconda fase riscandisce tutte le pagine, e quelle con bit di riferimento ancora a zero possono essere rimosse.

ESERCIZIO 3 (4 punti)

Un hard disk ha la capacità di 32 gigabyte, è formattato in blocchi da 200 (esadecimale) byte, e usa una qualche forma di allocazione indicizzata per memorizzare i file su disco. Sull'hard disk è memorizzato un file A grande 150 Kbyte. Nel rispondere alle domande sottostanti, specificate sempre le assunzioni che fate.

a) Quante operazioni di I/O su disco sono necessarie per portare in RAM l'ultimo blocco del file A, assumendo che inizialmente sia presente in RAM solo la copia del file directory che "contiene" il file?

   L'hard disk contiene $2^{35}/2^9 = 2^{26}$ blocchi, e sono quindi necessari 4 byte per scrivere in numero di un blocco. Un blocco indice può quindi contenere al massimo 512/4 = 128 numeri di blocco. La risposta dipende poi dal tipo di allocazione indicizzata assunta:

   1) Allocazione indicizzata a schema concatenato: sono necessari 3 blocchi indice per tenere traccia di tutti i blocchi del file. Se assumiamo che sia già in RAM il numero del primo blocco indice, sono necessarie 4 operazioni di I/O: lettura dei tre blocchi indice più lettura del blocco del file.
   2) Allocazione indicizzata a più livelli. È sufficiente usare uno schema a due livelli. Se assumiamo che sia già in RAM il numero del blocco indice esterno, sono necessarie 3 operazioni di I/O: lettura del blocco indice esterno, lettura di un blocco indice interno, lettura del blocco del file.
   3) Allocazione indicizzata Unix: Assumendo già in RAM il numero dell'index-node, sono necessarie 4 operazioni di I/O: le tre del punto 2) precedute dalla lettura dell'index-node.

b) Nel caso di accesso ai dati di file molto piccoli, è più efficiente l'implementazione scelta in Windows con NTFS o quella scelta da Unix con gli index-node? (motivate la vostra risposta)

   NTFS. Infatti per file molto piccoli, l'elemento che contiene gli attributi del file può contenere anche i dati del file stesso.

c) In Unix l'accesso agli attributi di un file è più efficiente se si passa attraverso uno dei suoi link fisici o se si passa attraverso uno dei suoi link simbolici (ovviamente assumendo che vi sia almeno un link simbolico a quel file)? Motivate la vostra risposta.

   Passando attraverso uno dei link fisici, in quanto è necessario accedere ad un solo index node anziché due.

d) In un sistema Unix viene dato il comando "rm A". In quale caso tutti i dati e gli attributi del file A sono effettivamente rimossi dal file system?

   Quando il link counter di A vale 1 (E' naturalmente necessario che l'utente che esegue il comando abbia ì permessi per farlo)