SISTEMI OPERATIVI E LABORATORIO
Indirizzo Sistemi e Reti – 13 luglio 2010

Cognome:_____Nome:_____
Matricola:_____

1. Ricordate che non potete usare calcolatrici o materiale didattico, e che se consegnate annullate automaticamente qualsiasi voto conseguito in una prova precedente.
2. Ricordate che se prendete un voto inferiore a 10 dovete saltare lo scritto della sessione di settembre.
3. Si ricorda che in questo appello è necessario sostenere l'intera prova scritta (teoria più laboratorio)

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) Come è possibile semplificare il codice del *consumatore* se si sa che c'è un solo processo *consumatore* che gira?

È sufficiente rimuovere le wait e le signal sul semaforo *mutex* di mutua esclusione.

c) Un processo A deve eseguire una certa procedura *myprocA()* **prima** che un altro processo B abbia eseguito la proceura *myprocB()*. Riportate una semplice soluzione a questo problema che faccia uso di un opportuno semaforo di sincronizzazione. Indicate anche il valore di inizializzazione del semaforo.

Semaphore sync = 0;

proc A:
myprocA();
signal(sync);

proc B:

wait(sync);
myprocB();

d) Perché una soluzione al problema della sezione critica basata sul busy waiting non è particolarmente adatta (anche se funzionante) in un sistema operativo time-sharing?

Perché i processi che chiedono di entrare in una sezione critica occupata consumano inutilmente tempo di CPU

e) In quale caso una soluzione al problema della sezione critica basata su busy waiting può comunque essere utile?

Per implementare le operazioni di wait e signal, che sono molto corte.

ESERCIZIO 2 (5 punti)

In un sistema la memoria fisica è divisa in 2^{22} frame, un indirizzo logico è scritto su 32 bit, e all'interno di una pagina, l'offset massimo è 3FF.

a) Quanti frame occupa la la page table più grande del sistema? (motivate numericamente la vostra risposta)

Un frame/pagina è grande 2^{10} = 1024 byte, e quindi la page table più grande può avere $2^{(32-10)} = 2^{22}$ entry. Nel sistema vi sono 2^{22} frame, per cui sono necessari tre byte per scrivere il numero di un frame, e quindi la page table più grande occupa $(2^{22} \cdot 3) / 2^{10}$ frame = $2^{12} \cdot 3$ frame = 12K frame

b) Il sistema può adottare una paginazione semplice (ad un solo livello)? (motivate la vostra risposta)

No perché la page table più grande non può essere memorizzata in un unico frame, e neanche in un numero ragionevolmente piccolo di frame adiacenti.

c) Quale dimensione minima dovrebbero avere le pagine di questo sistema per essere certi di non dover ricorrere ad una paginazione a più livelli? (motivate la vostra risposta, e per questa domanda assumete di usare 4 byte per scrivere il numero di un frame all'interno di una page table)

Poniamo 32 = 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, moltiplicato per la dimensione di una entry deve poter essere contenuto in una pagina/frame, ossia: $2^m \cdot 2^2 \leq 2^n$
Da cui: $m + 2 \leq n$. Poiché $m = 32 - n$; risolvendo il semplice sistema si ha $n = 17$, ossia le pagine devono almeno essere grandi 2^{17} = 128 Kbyte.

d) Questo sistema può non adottare un algoritmo di rimpiazzamento delle pagine?

Si, in quanto lo spazio di indirizzamento logico non è maggiore di quello fisico, per cui non è necessario implementare la memoria virtuale.

ESERCIZIO 3 (4 punti)

a) In quale caso l'uso dell'allocazione indicizzata dello spazio su disco risulta particolarmente svantaggiosa in termini di spazio su disco sprecato?

Nel caso di file piccoli, poiché quasi tutto il blocco indice viene sprecato.

b) L'uso di *cluster di blocchi adiacenti* (un po' come visto nel caso dell'allocazione concatenata) acuisce o rende meno grave il problema di cui si parla al punto a)?

Lo acuisce, in quanto il sistema lavora in sostanza con blocchi più grandi, e quindi lo spreco di spazio per memorizzare i file piccoli è proporzionalmente maggiore.

c) Utilizzando opportuni disegni descrivete le diverse forme di allocazione indicizzata viste a lezione: indicizzata *a schema concatenato,* indicizzata *a più livelli, variante adottata nei sistemi unix*.

Si vedano i lucidi della sezione 11.4.3

d) Se in un sistema unix i blocchi sono da 1024 byte e il numero di un blocco è scritto su 4 byte, qual è la dimensione massima di un file memorizzabile nel sistema (è sufficiente riportare l'espressione da usare per il calcolo)

Un blocco da 1024 byte può contenere 1024/4 = 256 puntatori a blocco. Si ha quindi:
$(10 * 1024) + (256 * 1024) + (256^2 * 1024) + (256^3 * 1024)$ byte