SISTEMI OPERATIVI E LABORATORIO
Indirizzo Sistemi e Reti – 1 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 in una qualsiasi delle due parti dello scritto, dovete saltare il secondo scritto della sessione di luglio.
3. Si ricorda che a partire da 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 lettori e scrittori visto a lezione, dove i codici del generico scrittore e del generico lettore sono riportati qui di seguito.

Inserite le operazioni di wait e signal mancanti necessarie per il funzionamento del sistema secondo la soluzione vista a lezione, indicando anche il semaforo mancante ed il suo valore di inizializzazione.

semafori e variabili condivise necessarie con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore scrivi = 1;
int numlettori = 0;

"scrittore"
{
wait(scrivi);
Esegui la scrittura del file
signal(scrivi)
}

"lettore"
{
wait(mutex);

numlettori++;

if numlettori == 1 **wait(scrivi);**

signal(mutex);

... leggi il file ...

wait(mutex);

numlettori--;

if numlettori == 0 **signal(scrivi);**

signal(mutex);

b) La soluzione del problema dei lettori e scrittori vista a lezione garantisce l'assenza di starvation? (motivate la vostra risposta)

No, infatti un qualsiasi processo scrittore potrebbe dover attendere all'infinito senza riuscire a ad entrare in sezione critica. Al contrario i processi lettori sono liberi da starvation (in altre parole, non è garantita l'attesa limitata

c) La soluzione del problema dei lettori e scrittori vista a lezione garantisce la condizione del progresso? (motivate la vostra risposta)

Si, in quanto esiste sempre almeno un processo che riesce ad entrare in sezione critica. In altre parole la soluzione garantisce l'assenza di deadlock.

d) Descrivete brevemente due algoritmi di scheduling preemptive e due algoritmi di scheduling non preemptive. Per ciascun algoritmo dite se soffre o no del problema della starvation

Si vedano le descrizioni degli algoritmi FCFS, SJF preemptive e non-preemptive, e Round Robin

ESERCIZIO 2 (5 punti)

a) Descrivete brevemente i tre tipi di binding degli indirizzi, indicando per ciascuno gli eventuali svantaggi.

Si vedano i ludici della sezione 8.1.2

b) Perché con la paginazione della memoria si ottiene una forma automatica di protezione dello spazio di indirizzamento di ciascun processo?

Si vedano i ludici della sezione 8.4.1

c) A parità di tutte le altre condizioni, i programmi girano più velocemente su un sistema che adotta una paginazione semplice della memoria o su un sistema che adotta una paginazione gerarchica?

Girano più velocemente su un sistema con paginazione semplice, perché la traduzione degli indirizzi richiede mediamente meno tempo

d) Due sistemi A e B sono completamente identici, ma A adotta una paginazione semplice della memoria e B adotta una paginazione a più livelli. Si supponga che nei due sistemi vi sia una quantità arbitrariamente grande *ma finita* di RAM, e che non sia implementata la memoria virtuale. In un qualsiasi istante, in quale dei due sistemi vi sarà con maggiore probabilità un più alto numero di processi contemporaneamente attivi? (motivate la vostra risposta)

Nel sistema B. Infatti, l'assenza di memoria virtuale e una quantità finita di RAM fa si che, nel sistema A, un processo con una page table molto grande potrebbe non poter girare se non si trova

un numero di frame *adiacenti* sufficiente per ospitare l'intera page table. Nel sistema B questa situazione occorre con meno probabilità, perché anche la page table (interna) può essere paginata.

e) Quali sono i vantaggi dell'uso delle librerie dinamiche?

Si vedano i ludici della sezione 8.1.2

ESERCIZIO 3 (4 punti)

Un hard disk ha la capienza di 2^{38} byte, ed è formattato in blocchi da 512 byte.

a) Quanti accessi al disco sono necessari per leggere l'ultimo blocco di un file A della dimensione di 4096 byte, assumendo che sia già in RAM il numero del primo blocco del file stesso e che venga adottata una allocazione concatenata dello spazio su disco? (motivate la vostra risposta)

9. Ogni blocco infatti memorizza 508 byte di dati più 4 byte di puntatore al blocco successivo (infatti, $2^{38}/2^9 = 2^{29}$), per cui sono necessari 9 blocchi per memorizzare l'intero file.

b) Qual è lo spreco di memoria dovuto alla frammentazione interna nella memorizzazione di A (motivate la risposta)?

L'hard disk è suddiviso in $2^{38}/2^9 = 2^{29}$ blocchi, sono necessari 4 byte per memorizzare un puntatore al blocco successivo, e ogni blocco contiene 508 byte di dati. Il nono blocco memorizzerà quindi 32 byte del file, e la frammentazione interna corrisponde a 512 – 32 = 480 byte (476 se si considerano non sprecati i 4 byte del nono blocco che contengono il puntatore, non utilizzato, al blocco successivo)

c) Se si adottasse una allocazione indicizzata dello spazio su disco, quanti accessi al disco sarebbero necessari per leggere l'ultimo byte di un file B grande 100k byte (specificate quali assunzioni fate nel rispondere a questa domanda e motivate la vostra risposta)?

Poiché sono necessari 4 byte per scrivere il numero di un blocco, in un blocco indice possono essere memorizzati 128 puntatori a blocco, e con un blocco indice possiamo indirizzare in tutto $2^7 * 2^9 = 2^{16} =$ 64k byte. Un solo blocco indice non è quindi sufficiente a memorizzare B. Assumendo una allocazione indicizzata concatenata (ma si ottengono gli stessi risultati con una allocazione indicizzata gerarchica) usando un secondo blocco indice è possibile memorizzare l'intero file. Se il numero del primo blocco indice è già in RAM, per leggere l'ultimo carattere del file sono necessari 3 accessi al disco: lettura del primo blocco indice, lettura del secondo blocco indice, lettura dell'ultimo blocco del file.