SISTEMI OPERATIVI E LABORATORIO
(Esonero e Scritto - Indirizzo Sistemi e Reti)
1 luglio 2008

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

Ricordate che non potete usare calcolatrici o materiale didattico. Siate sintetici nelle vostre risposte, anche quando è richiesto di motivarle, sono sufficienti poche righe per rispondere correttamente. (Si ricorda che gli studenti degli anni precedenti devono sostenere l'intero scritto).

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

(il punteggio conseguito farà media con quello ottenuto nella parte di laboratorio. E' comunque necessario prendere almeno 18 punti per considerare passata la parte di teoria o la parte di laboratorio.)

ESERCIZIO 1 (9 punti)

In un sistema con memoria virtuale le pagine hanno una dimensione di FFF byte, la RAM è fatta di 7FFF frame, e lo spazio di indirizzamento logico massimo è di FFFF pagine.

a) Qual è la lunghezza in bit di un indirizzo logico? 24 bit ($2^{16} \cdot 2^{12} = 2^{28}$)

b) Qual è la lunghezza in bit di un indirizzo fisico? 27 bit ($2^{15} \cdot 2^{12} = 2^{27}$)

c) Riportate lo schema che descrive l'architettura di paginazione (senza TLB)

vedere lo schema dei lucidi della sezione 8.4.1 (figura 8.7)

d) perché si adotta la paginazione della memoria?

Vedere lucidi della sezione 8.4.1

e) Perché viene di solito implementata anche la memoria virtuale?

Vedere lucidi della sezione 9.1

f) Che tipo di codice (o, equivalentemente, che tipo di binding degli indirizzi) usa un sistema che implementa paginazione e memoria virtuale, e perche?

Codice dinamicamente rilocabile. In questo modo gli indirizzi logici vengono convertiti in indirizzi fisici durante l'esecuzione dei programmi, e il codice può essere facilmente spezzato in pagine non adiacenti e queste possono essere spostate da un frame all'altro (in caso di page fault).

ESERCIZIO 2 (9 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) Elencate e descrivete sinteticamente un algoritmo di scheduling non pre-emptive e uno pre-emptive. Di ciascun algoritmo dite se può generare starvation.

Non pre-emptive:
First come first served. I processi usano la cpu nell'ordine in cui sono entrati in coda di ready. Non soffre di starvation

Shortest Job first non pre-emptive: il processo con il burst di cpu più corto tra quelli in coda di ready usa la cpu. Soffre di starvation.

Pre-emptive:
round robin: I processi usano la cpu per un quanto di tempo predefinito, poi tornano in coda di ready e saranno rischedulati solo dopo che tutti gli altri processi in coda hanno ricevuto un quanto di tempo. Non soffre di starvation.

Shortest Job first pre-emptive: il processo con il burst di cpu più corto tra quelli in coda di ready o in esecuzione usa la cpu. Soffre di starvation.

c) Riportate il diagramma di stato della vita di un processo, ed elencate tre diversi tipi di code in cui si può trovare un processo durante la sua vita.

Diagramma di stato: si vedano i lucidi della sezione 3.1.2.

1) coda ready to run
2) coda di attesa per l'uso di un determinato dispositivo
3) coda dei processi sospesi su un semaforo

ESERCIZIO 3 (9 punti)

Un hard disk ha la capienza di 2^{34} byte, ed è formattato in blocchi da 1024 byte.

a) Quanti accessi al disco sono necessari per leggere l'ultimo blocco di un file A della dimensione di 400 Kbyte, assumendo che siano già in RAM tutti gli attributi del file stesso e che venga adottata una allocazione indicizzata dello spazio su disco? (motivate la vostra risposta, esplicitando le assunzioni che fate.)

3. Infatti, ci vogliono 24 bit (3 byte) per memorizzare in numero di un blocco, per cui un blocco indice puo' contenere 1024/3 = 341 puntatori a blocco. Un solo blocco indice non e' quindi sufficiente a indirizzare tutti i blocchi di A. Se si assume una allocazione indicizzata a schema concatenato, con un secondo blocco indice possiamo indirizzare tutti i blocchi de file. Poiché è già in RAM il numero del primo blocco indice, è necessario leggere il primo blocco indice per recuperare il valore del secondo blocco indice, leggere il secondo blocco indice per sapere il numero dell'ultimo blocco del file, leggere questo blocco, per un totale di 3 letture in RAM (stesso risultato si ottiene assumendo una allocazione indicizzata a due livelli).

b) Si consideri un file A all'interno di un file system. Si supponga per semplicità che nel file system non siano presenti link fisici o simbolici. Quanti pathname assoluti ha il file A? Quanti pathname relativi ha il file A?

1; tanti quante sono le directory del file system che contiene A.

c) Che cosa succede quando un processo "apre" (esegue la system call "open" su) un file?

Una nuova entry viene allocata nella open file table del sistema, e gli attributi del file vengono copiati in RAM, eventualmente insieme alla prima porzione del file. Al processo che apre il file viene restituito il puntatore alla entry della open file table che contiene gli attributi del file (successivi accessi al file potranno essere fatti usando quel puntatore).

b) Descrivete brevemente il metodo di allocazione dello spazio su disco usato nei sistemi Unix (è sufficiente usare un opportuno disegno esplicativo).

Si vedano i lucidi della sezione 11.4.3