SISTEMI OPERATIVI E LABORATORIO
(Indirizzo Sistemi e Reti) – 19 febbraio 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. Gli **studenti in corso che decidono di sostenere solo la parte di teoria o solo la parte di laboratorio** devono consegnare i loro elaborati al termine della prima ora dello scritto.
3. Gli **studenti in corso che decidono di sostenere l'intero scritto (teoria + laboratorio)** ricordino che se prendono in una delle due parti un voto X insufficiente e:
   a. $X < 10$: non possono presentarsi al successivo scritto di febbraio/marzo
   b. $10 \leq X \leq 14$: devono risostenere l'intero scritto, anche se nell'altra parte hanno preso un voto maggiore o uguale a 18
   c. $14 < X$: possono riprovare la sola parte insufficiente nel secondo scritto di febbraio/marzo (ovviamente se nell'altra parte hanno raggiunto la sufficienza)
4. Gli **studenti degli anni precedenti** devono sostenere l'intero scritto.
5. Si ricorda che a partire dallo scritto di giugno non è più possibile sostenere separatamente l'esame delle due parti del corso. Chi non ha conseguito la sufficienza in *entrambe* le parti entro il secondo scritto della sessione di febbraio/marzo, da giugno deve comunque ridare 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.)

ESERCIZIO 1 (9 punti)

Si consideri un sistema in cui in una tabella delle pagine di un processo l'offset massimo (ossia l'indice più grande usabile nella tabella delle pagine) può essere 7FF. Un indirizzo fisico del sistema è scritto su 21 bit, e la RAM è suddivisa in 400 (esadecimale) frame.

(a) Quanto è grande lo spazio di indirizzamento logico del sistema (esplicitate i calcoli che fate)?

400(esadecimale) = 1024, per cui un numero di frame è scritto su 10 bit, e la dimensione di un frame, e quindi di una pagina, è di $2^{\wedge 11}$ byte. Poiché il numero più grande di una pagina è 7FF, ci possono essere al massimo $2^{\wedge 11}$ pagine, e lo spazio di indirizzamento logico è di $2^{\wedge 11} \times 2^{\wedge 11}$ byte (pari a circa 4 megabyte).

(b) Per ciascuna entry di una tabella delle pagine di questo sistema, è necessario memorizzare anche il bit di validità della pagina corrispondente? (motivate la vostra risposta)

Si. Infatti lo spazio di indirizzamento logico è più grande di quello fisico, e il sistema deve implementare anche la memoria virtuale.

(c) Per ciascuna entry di una tabella delle pagine di questo sistema, è necessario memorizzare anche il dirty bit della pagina corrispondente? (motivate la vostra risposta)

Si, nel caso in cui venga utilizzato un algoritmo di rimpiazzamento delle pagine che ne fa uso, come ad esempio l'algoritmo della seconda chance migliorato.

(d) Quale soluzione viene comunemente usata per limitare l'inefficienza dell'accesso in RAM tipica dei sistemi che adottano la paginazione della memoria?

Viene usato un TLB, una memoria associativa che limita il tempo necessario alla traduzione di un indirizzo logico in fisico.

(e) Che tipo di codice genera il compilatore di un moderno sistema operativo che implementa la memoria virtuale, e perché?

Codice dinamicamente rilocabile, in modo che il codice possa essere spostato, se necessario, da un punto all'altro della RAM senza dover ricalcolare gli indirizzi usati dalle istruzioni del codice stesso.

ESERCIZIO 2 (9 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(mutex)	**wait**(done)	**wait**(done)
<A>	**wait**(mutex)	**wait**(done)
signal(mutex)	<B>	**wait**(mutex)
signal(done)	**signal**(mutex)	<C>
	signal(done)	**signal**(mutex)

a1)
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. 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 le sequenze che scegliete con una croce nello spazio apposito)
A2)
In ciascuna delle sequenze restanti, che non possono essere prodotte dall'esecuzione concorrente dei tre processi, cerchiate la lettera che rappresenta l'ultima procedura che può effettivamente essere eseguita nell'esecuzione concorrente di P_A, P_B e P_C

1. [ ] A,B,B,C,A,A,C,B,A…
2. [X] A,A,B,C,A,A,C,A,A …
3. [X] A,B,A,A,C,A,B,B,C …
4. [ ] A,B,A,C,A,A,C,B,A …

b)
Riportate un semplice esempio in pseudo-codice (simile a quello usato per la prima parte di questa domanda) 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;

c)
All'interno di un sistema operativo, un certo processo P è correntemente in stato di "Running", e si sa che non dovrà più rilasciare la CPU volontariamente prima di aver terminato la propria esecuzione (in altre parole, non deve più eseguire operazioni di I/O, di sincronizzazione o di comunicazione con altri processi).
Quale/quali, tra gli algoritmi di scheduling **FCFS**, **SJF preemptive, SJF non-preemptive**, **round robin** garantisce/garantiscono che il processo P riuscirà a portare a termine la propria computazione? (motivate la vostra risposta, assumendo che SJF possa effettivamente essere implementato)

FCFS, SJF non-preemptive e round robin. Infatti, nel caso di SJF preemptive potrebbe sempre arrivare in coda di ready un processo che deve usare la CPU per un tempo minore di quanto rimane da eseguire a P.

d) in che modo potremmo assicurarci che P possa terminare la propria esecuzione indipendentemente da quale degli algoritmi di scheduling sopra indicati viene usato?

Usando un meccanismo di aging, cosicché la priorità di P aumenta quanto maggiore è il tempo che passa in coda di ready.

e) in quale caso P potrebbe uscire volontariamente dallo stato di "Ready to Run"?

mai.

ESERCIZIO 3 (9 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) 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

c) 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