Compito di Sistemi Operativi, Proff. Mancini e Massaioli

30 gennaio 2003 Compito A

1. (punti: -1,4)
   Nella gestione della memoria, l'algoritmo di sostituzione delle pagine:

   (a) ha lo scopo di mantenere in memoria fisica le pagine meno accedute;
   (b) determina la prossima pagina da caricare da disco in memoria fisica;
   (c) determina la prossima pagina da scaricare dalla memoria fisica su disco ;
   (d) può sostituire pagine non appartenenti ai processi che causano *page fault*;
   (e) può sostituire solo pagine appartenenti ai processi che causano *page fault*;
   (f) include l'algoritmo LRU che può essere definito in termini di *tempi dei riferimenti passati*;
   (g) include l'algoritmo LRU che può essere definito in termini di *tempi dei riferimenti futuri*;
   (h) nessuna delle affermazioni precedenti è corretta.

2. (punti: -1,4)
   Un sistema operativo ha tra i suoi obiettivi:

   (a) rendere più agevole la fabbricazione di un computer;
   (b) rendere più agevole la gestione di un computer;
   (c) rendere più agevole l'uso di un computer;
   (d) massimizzare l'efficienza di utilizzo delle risorse del sistema da parte di processi e utenti;
   (e) impedire un uso inefficiente delle risorse del sistema da parte di processi e utenti;
   (f) consentire aggiornamenti *hardware* con minime sostituzioni di *software*;
   (g) consentire aggiornamenti *hardware* solo a persone autorizzate;
   (h) compilare i programmi degli utenti in modo da renderli più veloci;
   (i) tutte le affermazioni precedenti sono corrette.

3. (punti: -1,4)
   Il descrittore di processo (*process control block*) contiene:

   (a) l'identificatore del processo e quello del processo padre;
   (b) l'identificatore del processo ma non quello del processo padre;
   (c) l'identità dell'evento di cui il processo è in attesa;
   (d) una copia dello stato del processore al momento dell'ultimo prerilascio o sospensione;
   (e) le aree dati, codice e *stack* del processo;
   (f) la sola area *stack* del processo, utilizzando puntatori alle *page table* per il resto;
   (g) le *page table* che descrivono la memoria virtuale assegnata al processo;
   (h) le variabili condivise con altri processi nelle sezioni critiche;
   (i) nulla di quanto elencato sopra.

4. (punti: -1,4)
   La funzionalità del *disk scheduling* riordina le richieste di I/O su disco in arrivo dai processi:

   (a) in modo da distribuirle tra tutti i dischi del sistema e bilanciarne il carico di lavoro;

(b) al fine di ridurre l'usura delle testine;

(c) utilizzando eventualmente una politica FIFO;

(d) utilizzando eventualmente una politica a priorità;

(e) utilizzando eventualmente una politica *round-robin*;

(f) utilizzando eventualmente una politica *second chance*;

(g) utilizzando eventualmente una politica LIFO;

(h) privilegiando l'esecuzione dei processi I/O-*bound*;

(i) in modo da ridurre il tempo di *seek* medio;

(j) in modo da ridurre il ritardo rotazionale medio;

(k) in modo da ridurre il tempo di trasferimento medio dei dati;

(l) in modo da ridurre il tempo di accesso medio.

5. (punti: -1,4)
Un sistema operativo utilizza uno *scheduler* di processi basato su quanti di tempo e priorità intere non negative. Lo *scheduler* opera ripetendo all'infinito un ciclo composto dai quattro passi seguenti:
- viene chiamata una subroutine definita dall'amministratore del sistema, che aggiorna le priorità di tutti i processi;
- viene determinato l'insieme dei processi che hanno il massimo valore di priorità su tutti quelli in stato di pronto;
- viene assegnato un quanto di tempo ad un processo scelto a caso in tale insieme;
- al termine del quanto di tempo il processo viene prerilasciato.
All'atto della sottomissione di un processo, gli viene attribuita priorità 0, ed è possibile specificare la durata totale del processo in quanti di tempo.

Scrivendo l'opportuna subroutine di aggiornamento delle priorità, è possibile far si che a regime (ossia trascurando comportamenti differenti nei primi istanti di vita dei processi) lo *scheduler* si comporti secondo:

(a) una politica *round-robin*, grazie ad uno schema di priorità statiche (la subroutine non cambia le priorità);

(b) una politica *round-robin*, grazie ad uno schema di priorità dinamiche (la subroutine cambia opportunamente le priorità);

(c) una politica *round-robin* con quanti di tempo multipli interi del quanto di tempo dello *scheduler*;

(d) una politica *round-robin* con quanti di tempo sottomultipli interi del quanto di tempo dello *scheduler*;

(e) una politica *First-Come-First-Served*, grazie ad uno schema di priorità statiche (la subroutine non cambia le priorità);

(f) una politica *First-Come-First-Served*, grazie ad uno schema di priorità dinamiche (la subroutine cambia opportunamente le priorità);

(g) una politica *Shortest Process Next*, grazie ad uno schema di priorità statiche (la subroutine non cambia le priorità);

(h) una politica *Shortest Process Next*, grazie ad uno schema di priorità dinamiche (la subroutine cambia opportunamente le priorità);

(i) una politica *Shortest Remaining Time*, grazie ad uno schema di priorità statiche (la subroutine non cambia le priorità);

(j) una politica *Shortest Remaining Time*, grazie ad uno schema di priorità dinamiche (la subroutine cambia opportunamente le priorità).

6. (punti: 6)
Illustrare in al più 70 parole scopi e funzionamento del *buffering* dell'I/O.

7. (punti: 6)
Illustrare in al più 70 parole la nozione di stallo ed i metodi per prevenirlo.

8. (punti: 6)
Considerare un insieme di sei processi P_1, P_2, P_3, P_4, P_5, e P_6 con i seguenti tempi di arrivo e tempi di esecuzione in millisecondi:

Processo	Tempo di arrivo	Tempo di esecuzione
P_1	2	9
P_2	4	17
P_3	5	15
P_4	25	7
P_5	28	6
P_6	61	3

Assegnare questo insieme di processi ad un processore in base alla politica Round Robin considerando un quanto di tempo di 5 millisecondi.

Calcolare il valor medio del tempo di attesa ed il valor medio del tempo di turnaround dei processi.

9. Considerare la seguente stringa di riferimenti di un processo alla memoria in un sistema con memoria virtuale
S = 1 0 2 2 1 7 6 7 0 1 2 0 8 0 4 5 1 5 2 4

(a) (punti: 3)
Illustrare il comportamento dell'algoritmo LRU di sostituzione delle pagine per una memoria fisica di 5 blocchi. Calcolare il numero di page fault che si verificano.

(b) (punti: 3)
Illustrare il comportamento dell'algoritmo Second Chance di sostituzione delle pagine per una memoria fisica di 5 blocchi. Calcolare il numero di page fault che si verificano.

10. (punti: 16)
E' necessario sviluppare un'applicazione concorrente che necessita di sincronizzazioni basate su semafori. Purtroppo il sistema operativo che la dovrà ospitare non fornisce tale meccanismo, né supporta la condivisione di aree di memoria tra processi, ma esclusivamente forme di comunicazione basate su scambio di messaggi tra processi. Il sistema di comunicazione supporta comunicazioni bloccanti e non bloccanti, ed indirizzamento diretto o indiretto.
Scegliere le forme di comunicazione e di indirizzamento più appropriate (motivando brevemente la scelta) ed utilizzarle per implementare in pseudocodice le primitive init() P() e V() per il generico semaforo produttore-consumatore.

Risposte Scritto di Sistemi Operativi - Compito A - Data,

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

1. (punti: -1,4) **a** **b** **c** **d** **e** **f** **g** **h** **i** **j** **k** **l**
2. (punti: -1,4) **a** **b** **c** **d** **e** **f** **g** **h** **i** **j** **k** **l**
3. (punti: -1,4) **a** **b** **c** **d** **e** **f** **g** **h** **i** **j** **k** **l**
4. (punti: -1,4) **a** **b** **c** **d** **e** **f** **g** **h** **i** **j** **k** **l**
5. (punti: -1,4) **a** **b** **c** **d** **e** **f** **g** **h** **i** **j** **k** **l**
6. (punti: 6)

7. (punti: 6)

8. (punti: 6)

9. (a) (punti: 3)

F_0																				
F_1																				
F_2																				
F_3																				
F_4																				
F_5																				

(b) (punti: 3)

F_0																				
F_1																				
F_2																				
F_3																				
F_4																				
F_5																				