Sistemi Operativi
16 maggio 2012
Compito B

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Un processo in background dovrebbe funzionare in modo da non degradare significativamente il servizio fornito agli altri processi. Quale delle seguenti alternative si suggerisce per implementarlo?

   (a) Assegnare la minima priorità al processo in background.

   (b) Attribuire un quanto minore al processo in background rispetto agli altri processi.

   Risposta: (3 punti) L'alternativa corretta è la seconda, ovvero, attribuire al processo in background un quanto di tempo minore; assegnargli la minima priorità infatti potrebbe provocare una starvation del processo in background.

2. Si considerino due processi concorrenti P_i e P_j che eseguono la propria sezione critica in alternanza stretta. Il codice di P_i è il seguente:

```
while (TRUE) {
    while (turn ≠ i) no-op;
       sez.  critica
    turn := j;
       sez.  non critica
};
```

   dove `turn` è una variabile inizializzata a i e il codice di P_j è analogo a quello di P_i.

   Si scriva un codice alternativo per i processi P_i e P_j, che utilizza i semafori anziché la variabile `turn` per garantire l'alternanza stretta.

   Risposta: (4 punti) Utilizziamo due semafori binari sem_i e sem_j, tali che

$$sem_i = \begin{cases} 1 \text{ (è il turno di } P_i) \\ 0 \text{ (non è il turno di } P_i) \end{cases}$$

$$sem_j = \begin{cases} 1 \text{ (è il turno di } P_j) \\ 0 \text{ (non è il turno di } P_j) \end{cases}$$

   I due semafori vengono inizializzati come segue:
   $sem_i := 1$
   $sem_j := 0$
   dopodichè i due processi eseguono il codice seguente:
   P_i:

```
while (TRUE) {
    down(sem_i);
    sez.  critica
    up(sem_j);
    sez.  non critica
};
```

   P_j:

```
while (TRUE) {
    down(sem_j);
    sez.  critica
    up(sem_i);
    sez.  non critica
};
```

3. Le seguenti richieste sarebbero accettate nello stato attuale dall'algoritmo del banchiere?

Richieste massime			*Risorse allocate*			*Totali allocate*		*Totali esistenti*	
	R_1	R_2		R_1	R_2	R_1	R_2	R_1	R_2
P_1	2	5	P_1	1	3	3	4	4	5
P_2	3	2	P_2	2	1				

   (a) Il processo P_2 chiede $(1, 0)$.

   (b) Il processo P_2 chiede $(0, 1)$.

   (c) Il processo P_2 chiede $(1, 1)$.

(d) Il processo P_1 chiede $(1, 0)$.

(e) Il processo P_1 chiede $(0, 1)$.

Risposta: (5 punti) Per rispondere alle domande, calcoliamo la matrice R delle richieste (*Richieste massime* − *Risorse allocate*) ed il vettore A delle risorse disponibili (*Totali esistenti* − *Totali allocate*):

Richieste (R)

	R_1	R_2
P_1	1	2
P_2	1	1

$A = (1, 1)$

(a) Se venisse accettata la richiesta $(1, 0)$ del processo P_2, i valori di *Risorse allocate*, R e A sarebbero i seguenti:

Risorse allocate

	R_1	R_2
P_1	1	3
P_2	3	1

Richieste (R)

	R_1	R_2
P_1	1	2
P_2	0	1

$A = (0, 1)$

La richiesta verrebbe accettata, dato che esiste la sequenza di esecuzione sicura P_2, P_1.

(b) Se venisse accettata la richiesta $(0, 1)$ del processo P_2, i valori di *Risorse allocate*, R e A sarebbero i seguenti:

Risorse allocate

	R_1	R_2
P_1	1	3
P_2	2	2

Richieste (R)

	R_1	R_2
P_1	1	2
P_2	1	0

$A = (1, 0)$

La richiesta verrebbe accettata, dato che esiste la sequenza di esecuzione sicura P_2, P_1.

(c) Se venisse accettata la richiesta $(1, 1)$ del processo P_2, i valori di *Risorse allocate*, R e A sarebbero i seguenti:

Risorse allocate

	R_1	R_2
P_1	1	3
P_2	3	2

Richieste (R)

	R_1	R_2
P_1	1	2
P_2	0	0

$A = (0, 0)$

La richiesta verrebbe accettata, dato che esiste la sequenza di esecuzione sicura P_2 (ha già ottenuto tutte le risorse di cui necessita), P_1.

(d) Se venisse accettata la richiesta $(1, 0)$ del processo P_1, i valori di *Risorse allocate*, R e A sarebbero i seguenti:

Risorse allocate

	R_1	R_2
P_1	2	3
P_2	2	1

Richieste (R)

	R_1	R_2
P_1	0	2
P_2	1	1

$A = (0, 1)$

La richiesta verrebbe rifiutata, dato che non esiste nessuna riga di R minore od uguale al vettore A (lo stato di esecuzione non è sicuro).

(e) Se venisse accettata la richiesta $(0, 1)$ del processo P_1, i valori di *Risorse allocate*, R e A sarebbero i seguenti:

Risorse allocate

	R_1	R_2
P_1	1	4
P_2	2	1

Richieste (R)

	R_1	R_2
P_1	1	1
P_2	1	1

$A = (1, 0)$

La richiesta verrebbe rifiutata, dato che non esiste nessuna riga di R minore od uguale al vettore A (lo stato di esecuzione non è sicuro).

4. Una macchina ha uno spazio degli indirizzi a 32 bit e una pagina di 8 KB. La tabella delle pagine è completamente nell'hardware, con una parola a 32 bit per voce. Quando parte un processo, la tabella delle pagine è copiata dalla memoria nell'hardware, una parola ogni 100ns. Quale parte del tempo di CPU è dedicato al caricamento delle tabelle delle pagine, se ciascun processo è eseguito per 100ms (compreso il tempo di caricamento della tabella delle pagine)?

   Risposta: (6 punti) Per rappresentare l'offset in una pagina di 8KB servono 13 bit ($2^{13} = 8192$); quindi rimangono 19 bit per il numero di pagina virtuale. Conseguentemente avremo $2^{19} = 524288$ voci nella page table: per caricare le pagine il sistema impiegherà quindi $524288 \cdot 100\ ns = 52428800\ ns$, ovvero, $52,4288\ ms$ (il $52,5\%$ del tempo dedicato al processo).

5. Si fornisca un esempio semplice di una sequenza di riferimenti a pagine dove la prima pagina selezionata per la sostituzione sia diversa a seconda che sia usato l'algoritmo di sostituzione Clock o LRU. Si assuma che il processo allochi 3 frame e la stringa di riferimenti contenga numeri di pagina dall'insieme 1,2, 3, 4.

   Risposta: (5 punti) Consideriamo la sequenza di riferimenti seguente:

   1 2 3 1 4

   In questo caso la prima pagina selezionata per la sostituzione da LRU sarà 2 (il secondo riferimento alla pagina 1, la riporta in cima alla pila, facendo in modo che in occasione del riferimento alla pagina 4 la pagina in fondo alla pila dei riferimenti sia 2). Invece Clock selezionerà la pagina 1 in quanto dopo i primi 4 riferimenti tutti i reference bit delle pagine 1, 2, 3 sono impostati a 1 e quindi Clock degenera in un FIFO, scegliendo la pagina da più tempo in memoria (1).

6. Si illustri la struttura di un inode (ovvero quali informazioni contiene).

   Risposta: (3 punti) Ogni inode in un sistema UNIX contiene le seguenti informazioni:

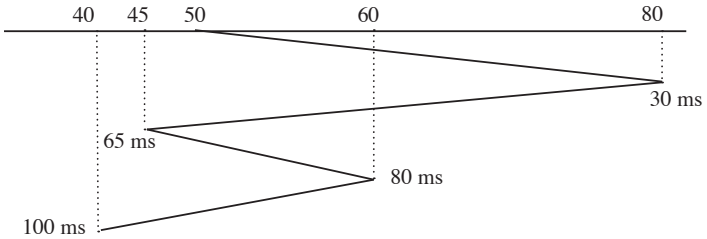
   - modo: bit di accesso, di tipo e speciali del file,
   - UID e GID del possessore,
   - dimensione del le in byte,
   - timestamp di ultimo accesso (atime), di ultima modica (mtime), di ultimo cambiamento dellinode (ctime),
   - numero di link hard che puntano all'inode,
   - blocchi diretti: puntatori ai primi 12 blocchi del file,
   - primo indiretto: indirizzo del blocco indice dei primi indiretti,
   - secondo indiretto: indirizzo del blocco indice dei secondi indiretti,

7. Si consideri un disco gestito con politica C-LOOK (Look *circolare*). Inizialmente la testina è posizionata sul cilindro 50 con direzione di servizio verso tracce con numero crescente; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 80, 45, 60, 40, rispettivamente agli istanti 0 ms, 10 ms, 60 ms, 70 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. (3 punti) Le richieste vengono servite nell'ordine 80, 45, 60, 40:



2. (2 punti) Il tempo di attesa medio per le quattro richieste in oggetto è $\frac{(30-0)+(65-10)+(80-60)+(100-70)}{4} = \frac{30+55+20+30}{4} = \frac{135}{4} = 33,75\ ms.$