Sistemi Operativi A

6 Settembre 2007

Esercizio 1

In un sistema, sono pronti per essere eseguiti i 4 processi P1, P2, P3 e P4. Tali processi hanno, rispettivamente, priorità 2,1,3,2 (1 più alta, 3 più bassa) e tempi di esecuzione 3,7,1,6. Nessuno dei quattro processi esegue operazioni di I/O durante la propria esecuzione.
Indicare con:

- CPU il processo in esecuzione
- Pn i processi pronti a priorità n (nel caso di scheduler Round Robin a priorità)
- T i processi terminati

Mostrare come vengono eseguiti tali processi e calcolare il tempo medio di completamento per processo (*turnaround* medio), nel caso in cui l'algoritmo di scheduling sia:

1. Round Robin senza priorità (quanto = 1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				

   Tournaround medio =

2. Round Robin a priorità statica (quanto = 1). Nota: se allo scadere del quanto non ci sono altri processi con priorita' uguale o maggiore, viene ri-schedulato lo stesso processo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				

   Tournaround medio =

3. Shortest Process Next (SPN)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				

   Tournaround medio =

Esercizio 2

Un sistema dispone di 5 unità nastro. Si presentano 4 processi che effettuano, nell'ordine, le seguenti richieste (i processi rilasciano i nastri solo dopo aver allocato il numero massimo di nastri dichiarato inizialmente)

Processo	unità massime	unità richieste
P1	3	1
P2	4	2
P3	3	1
P4	2	1

1. Mostrare una sequenza di allocazione che porta in uno stato di stallo (deadlock), spiegando approfonditamente;
2. Illustrare come la situazione sopra descritta viene gestita tramite l'algoritmo del banchiere.

Esercizio 3

Perchè è importante che l'attesa di un processo, ad esempio all'ingresso di una sezione critica, sia limitata? Illustrare un esempio di attesa indefinita (starvation).

Esercizio 4

Si consideri il problema dei filosofi a cena con 3 filosofi

```
Process Filosofo_0 {
   while(1) {
      <pensa>

      <mangia>

   }
}
```

```
Process Filosofo_1 {
   while(1) {
      <pensa>

      <mangia>

   }
}
```

```
Process Filosofo_2 {
   while(1) {
      <pensa>

      <mangia>

   }
}
```

Aggingere le necessarie sincronizzazioni utilizzando un array di tre semafori `fork[3]` ed evitando situazioni di stallo (deadlock). Spiegare dettagliatamente la soluzione proposta.

Esercizio 5

Si consideri un processo che fa rifermento a 5 pagine virtuali nel seguente ordine:

$$1, 2, 3, 4, 2, 4, 5, 4, 5, 1, 5, 1$$

Si consideri una memoria fisica (inizialmente vuota) di 3 frame e si mostri il funzionamento degli algoritmi seguenti:

1. sostituzione ottima,

   Spiegare brevemente:

	1	2	3	4	2	4	5	4	5	1	5	1
Frame 1												
Frame 2												
Frame 3												
Page Fault												

2. LRU (Least Recently Used),

   Spiegare brevemente:

	1	2	3	4	2	4	5	4	5	1	5	1
Frame 1												
Frame 2												
Frame 3												
Page Fault												

3. dell'orogologio (clock o seconda chance), indicando puntatore e reference bit.

   Spiegare brevemente:

	1	2	3	4	2	4	5	4	5	1	5	1
Frame 1												
Frame 2												
Frame 3												
Page Fault												

Esercizio 6

Che cos'è lo stallo (deadlock)? Illustrare una tecnica di *prevenzione* dello stallo.

Esercizio 7

Implementare le operazioni P e V (ovvero wait e signal) dei semafori contatori tramite un monitor. Assumere che la coda sulla condition sia gestita con politica FIFO.

```
monitor semaforo {
    int valore;
    condition attendi;

    P() {                              V() {                  init(i) {
                                                                   valore=i;
                                                               }

    }                                  }
}
```

Discutere approfonditamente la soluzione proposta.