Sistemi Operativi A

27 Giugno 2006

Esercizio 1

Considerare una variante del problema del produttore e consumatore in cui sia presente un solo consumatore che utilizzi tutti i dati prodotti da n produttori. Il produttore i-esimo produce il dato $x[i]$. Aggiungere le necessarie sincronizzazioni al codice del produttore i-esimo e del consumatore utilizzando i semafori. Spiegare il funzionamento della soluzione indicata.

```
produttore(i) {                      consumatore {
  while(true) {                      while(true) {


    <produci x[i]>                       <consuma x[1] ... x[n]>


} }                                  }}
```

Spiegazione:

Esercizio 2

Si consideri il seguente processo

```
process CopiaFile(i,j){
    fs.acquisisci(i); fs.acquisisci(j);
    print("copio da ",i," a ",j);
    fs.rilascia(i); fs.rilascia(j);
}

monitor fs {
    boolean lock[10]=false;
    condition attendi;

    void acquisici(int i) {          void rilascia(int i) {
         while (lock[i])                  lock[i]=false;
             attendi.wait;                attendi.notifyall;
         lock[i]=true;               }
    }
```

1. Considerare l'esecuzione concorrente di due istanze di CopiaFile(0,1). Quali sono i possibili output?

   Spiegare:

2. Considerare l'esecuzione concorrente di CopiaFile(1,0) e CopiaFile(0,1). Quali sono i possibili output in questo caso?

   Spiegare:

Esercizio 3

In un sistema time-sharing con scheduling Round Robin a priorità sono presenti 3 processi P1, P2, P3 nei seguenti stati: P1 è in esecuzione; P2 è in attesa su operazione di I/O, P3 è pronto. Ipotizzare che P2 abbia priorità "alta" mentre P1 e P3 abbiano priorità "bassa".

1. Descrivere, tramite un opportuno schema, come cambia lo stato del sistema se, partendo da questa situazione, si verificano, nell'ordine, i seguenti eventi:
   (a) completamento dell'operazione di I/O per P2
   (b) quanto di tempo
   (c) quanto di tempo
   (d) il processo in esecuzione chiede una operazione di I/O
   (e) quanto di tempo
   (f) il processo in esecuzione termina

2. Come cambierebbe lo stato del sistema nel caso di sceduling a code multiple in cui i processi a bassa priorità siano gestiti con scheduling batch FCFS? (ignorare eventuali eventi "quanto di tempo" durante l'esecuzione di processi batch)

Esercizio 4

Viene progettata una nuova applicazione multithreaded per la prenotazione dei posti sui treni in cui i vari programmi "client", eseguiti sui terminali automatici nelle varie stazioni dei treni, comunicano con un programma centralizzato "server" che si occupa di eseguire la prenotazione vera a propria. Ad ogni nuova richiesta da parte di un client, il server crea un nuovo thread per gestire la prenotazione. Immaginare, per semplicità, che i dati delle prenotazioni siano tutti conservati in una memoria centrale condivisa da tutti i thread sulla macchina server.
L'applicazione è stata accuratamente testata con tre client prima di essere messa in funzione ma dopo poche settimane di utilizzo a livello nazionale si verificano alcuni casi di viaggiatori che reclamano la prenotazione dello stesso posto sul medesimo treno. Cosa può essere accaduto? Mostrare un frammento di codice che può aver causato il problema sopra menzionato e spiegare come modificarlo in modo da renderlo funzionante.

Esercizio 5

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

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

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

1. FIFO (First In First Out),

   Spiegare:

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

2. LRU (Least Recently Used),

   Spiegare:

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

3. dell'orogologio (clock).

   Spiegare:

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

Commentare brevemente i risultati ottenuti:

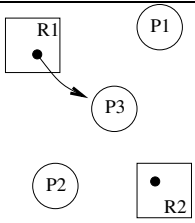
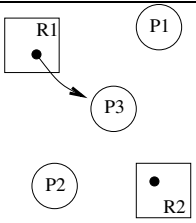
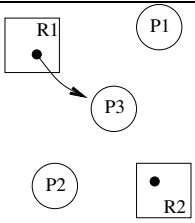
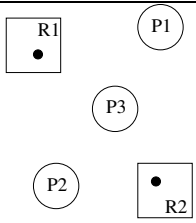
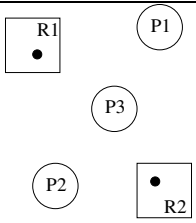
Esercizio 6

Una macchina server di un dipartimento di astronomia contiene immagini bitmap ad altissima definizione, ognuna di svariati Mbyte, e viene utilizzata da diverse macchine client le quali richiedono il "download" e la visualizzazione di tali immagini. Per dare il prima possibile una visione globale dell'immagine attualmente scaricata, si modifica la procedura di scaricamento in modo tale che i pixel dell'immagine vengano inviati dal server al client in ordine casuale invece che con una scansione per righe. L'effetto visivo è buono ma le prestazioni del server decadono in modo vistoso, soprattutto quando ci sono diverse richieste da parte dei client. In che tipo di problema ci si è quasi sicuramente imbattuti? Spiegare.

Esercizio 7

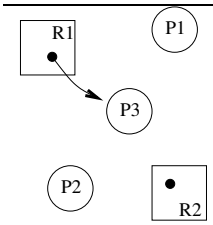
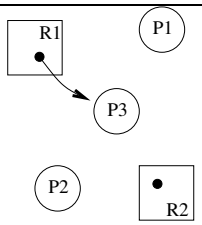
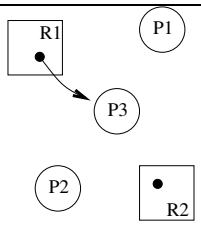
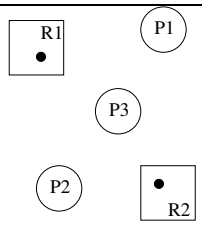
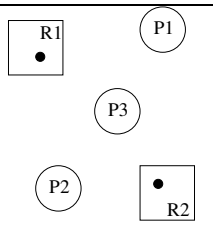
Si consideri un sistema con una risorsa di tipo R1 e una risorsa di tipo R2. Sono presenti tre processi: P1, P2 e P3 e il processo P3 ha già assegnata la risorsa di tipo R1. Completare con le opportune frecce il grafo di assegnazione dopo ogni richiesta nei tre casi sotto menzionati:

1. Nessuna gestione del deadlock da parte del sistema operativo (gestione FIFO delle richieste)

				
P1 chiede R1	P2 chiede R2	P2 chiede R1	P3 rilascia R1	P1 chiede R2
Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:

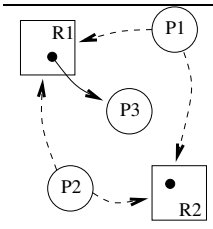
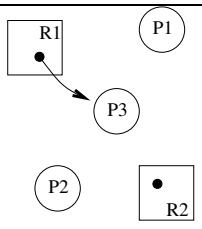
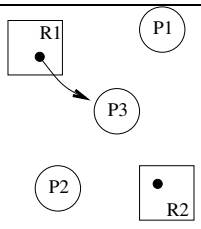
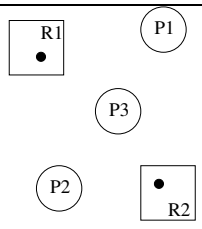
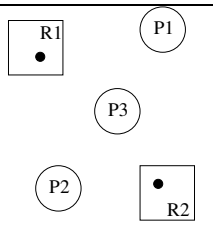
Si verifica una situazione di deadlock? Perchè?

2. Gestione gerarchica delle risorse: R1 < R2

				
P1 chiede R1	P2 chiede R2	P2 chiede R1	P3 rilascia R1	P1 chiede R2
Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:

Si verifica una situazione di deadlock? Perchè?

3. Algoritmo con grafo di assegnazione e archi di reclamo

				
P1 chiede R1	P2 chiede R2	P2 chiede R1	P3 rilascia R1	P1 chiede R2
Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:	Spiegazione:

Si verifica una situazione di deadlock? Perchè?