]SISTEMI OPERATIVI E LABORATORIO

16 settembre 2009

c)
Descrivete brevemente i problemi che si possono presentare se non viene correttamente risolto il problema della sezione critica

Si vedano i ludici della sezione 6.2

d)
Che cosa significa che un sistema operativo ha un *kernel con diritto di prelazione*?

Si vedano i ludici della sezione 6.2

e)
Un sistema operativo usa un algoritmo di scheduling Round Robin e lascia che i processi utente si sincronizzino fra loro usando un meccanismo di sincronizzazione via hardware (ad esempio implementato attraverso l'istruzione macchina Test&Set) basato su busy waiting. Cosa succede quando un processo cerca di entrare in una sezione critica già occupata?

Si vedano i ludici della sezione 6.4

SISTEMI OPERATIVI E LABORATORIO

16 dicembre 2009

ESERCIZIO 1 (7 punti)

a) Riportate lo pseudocodice che descrive l'implementazione delle operazioni di Wait e Signal. Dite che cosa fanno le system call usate nel codice.

Si vedano i ludici della sezione 6.6.2

b) Riportate in pseudocodice un semplice esempio di due processi che si sincronizzano fra loro, e che, *a seconda dell'ordine con cui usano la cpu*, possono o meno entrare in una situazione di deadlock. Riportate anche il valore di inizializzazione del o dei semafori usati

Si vedano i ludici della sezione 6.8.1

c) Riportare il diagramma di stato della vita di un processo.

Si vedano i lucidi alla sezione 3.1.2

d) rispetto al diagramma di stato della vita di un processo, e considerando un moderno sistema time sharing con memoria paginata che implementa la memoria virtuale, descrivete brevemente le diverse ragioni per cui un processo può transire nella condizione di wait.

1. quando deve compiere una operazione di I/O
2. quando esegue una operazione di wait su un semaforo
3. quando il codice eseguito dal processo genera un page fault

SISTEMI OPERATIVI E LABORATORIO

7 gennaio 2010

ESERCIZIO 1 (7 punti)

a) Si consideri il problema dei lettori e scrittori visto a lezione, dove i codici del generico scrittore e del generico lettore sono riportati qui di seguito.

Inserite le operazioni di wait e signal mancanti necessarie per il funzionamento del sistema secondo la soluzione vista a lezione, indicando anche il semaforo mancante ed il suo valore di inizializzazione.

semafori e variabili condivise necessarie con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore ????????;
int numlettori = 0;

"scrittore"
{

Esegui la scrittura del file

}

"lettore"
{
wait(mutex);

numlettori++;

if numlettori == 1

signal(mutex);

... leggi il file ...

wait(mutex);

numlettori--;

if numlettori == 0

signal(mutex);
}

semafori e variabili condivise necessarie con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore scrivi = 1;
int numlettori = 0;

"scrittore"
{
wait(scrivi);
Esegui la scrittura del file
signal(scrivi);
}

"lettore"
{
wait(mutex);

numlettori++;
if numlettori == 1 **wait(scrivi);**

signal(mutex);
... leggi il file ...
wait(mutex);

numlettori--;
if numlettori == 0 **signal(scrivi);**

signal(mutex);
}

b) Elencate le tre condizioni che deve rispettare una corretta soluzione al problema della sezione critica, e descrivete i problemi che si possono presentare quando queste condizioni non vengono garantite.

Lucidi 6.2

c) Descrivete brevemente due algoritmi di scheduling preemptive e due algoritmi di scheduling non preemptive. Per ciascun algoritmo dite se soffre o no del problema della starvation

Si vedano le descrizioni degli algoritmi FCFS, SJF preemptive e non-preemptive, e Round Robin

SISTEMI OPERATIVI E LABORATORIO
21 luglio 2009

ESERCIZIO 1 (9 punti)

Quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante)

Processo	T. di arrivo	Burst
P1	0	13
P2	2	9
P3	2	7
P4	12	4

a) Calcolare il turnaround medio e il waiting time medio per i processi nel caso dell'algoritmo di scheduling SJF preemtpive (shortest remaining time first). RIPORTANDO IL DIAGRAMMA DI GANTT USATO PER IL CALCOLO.

Turnaround medio:

P1 = P2 = P3 = P4 =

Waiting time medio:

P1 = P2 = P3 = P4 =

b) SJF premptive può causare problemi di starvation? E SJF *non* preemptive?

……

Si. In entrambi gli algoritmi, potrebbero arrivare in coda di ready sempre nuovi processi con un CPU time inferiore a processi già in coda di ready e in attesa di essere schedulati.

c) Descrivete brevemente la differenza tra un algoritmo di scheduling preemptive e un algoritmo non-preemptive

……

In un algoritmo preemptive, quando in coda di ready entra un processo con una priorità migliore di quella del processo correntemente in CPU, quest'ultimo deve lasciare il processore a favore del processo appena arrivato. In un algoritmo non-preemptive il processo che ha la CPU la rilascia solo volontariamente.

d) Descrivete brevemente un algoritmo di scheduling a code multiple con retroazione

……..

Usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere spostato da una coda all'altra in base a come si è comportato l'ultima volta che gli è stata assegnata la CPU.

ESERCIZIO 1 (5 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(done)	**wait**(done)	**wait**(mutex)
wait(done)	**wait**(mutex)	<C>
wait(mutex)	<B>	**signal**(mutex)
<A>	**signal**(mutex)	**signal**(done)
signal(mutex)	**signal**(done)	

a) 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. Quale o quali delle sequenze qui sotto riportate può essere la porzione iniziale di sequenze prodotte dall'esecuzione concorrente di P_A, P_B e P_C? (marcate la o le sequenze che scegliete con una croce nello spazio apposito)

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

……

	C	B	B	A?
done = 0	1	1	1	

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

b) Durante l'esecuzione dei tre processi P_A, P_B e P_C, è possibile dire qual è *il valore più piccolo* che può assumere la variabile semaforica di "done"? E' possibile dire qual è *il valore massimo* che può assumere la variabile semaforica di "done"? (motivate la vostra risposta, assumendo che P_A, P_B e P_C siano i soli processi ad usare "done")

……..

Il valore più piccolo è -2: si verifica se P_A e P_B eseguono la wait sul semaforo uno di seguito all'altro.
Il valore massimo è indefinito (in realtà, in linea di principio può essere pari al più grande valore intero rappresentabile nel sistema su cui girano i processi): non si può infatti stabilire a priori quante volte P_C può ciclare prima che gli altri due processi riescano ad entrare in CPU.