Process Synchronization- Peterson's Solution Semaphores

By
Dr. Upasana Pandey
Department of Computer Science & Engineering
IMS Engineering College (College Code:143)

Using turn variable two process solution for critical section

Boolean int turn=0;

P0	P1

while (1)
{
while (turn !=0);
Critical Section
turn=1;
Remainder section;
}
	while (1)
	{
	while (turn !=1);
	Critical Section
	turn=0;
	Remainder section;
	}

Outcome:
Mutual Exclusion is satisfied but Progress criteria is not satisfied. Therefore it is not consistent solution.

Using flag variable two process solution for critical section

```
Boolean flag[2];
flag [0]=false;
flag [1]=false;
```

P0	P1

1. while (1)
2. {
3. flag[0]=true;
4. while (flag[1]);
5. Critical Section
6. flag[0]=false;
}
	1. while (1)
	2. {
	3. flag[1]=true;
	4. while (flag[0]);
	5. Critical Section
	6. flag[1]=false;
	}

Outcome:
Mutual Exclusion and Progress criteria, both are satisfied. But if P0 executed till line 3 and context switch occurs for P1, P1 executed till line 3 and check the condition which is false then cannot enter into critical section. Same is happening with P0 also. In this situation no process can enter into critical section. Deadlock occurred.

Peterson’s Solution for Critical Section

Boolean int turn=0;
Boolean flag[2];
flag [0]=false;
flag [1]=false;

P0	P1

1. while (1)
2. {
3. flag[0]=true;
4. turn=1;
5. while (turn==1 && flag[1]==true);
6. Critical Section
7. flag[0]=false;
}
	1. while (1)
	2. {
	3. flag[1]=true;
	4. turn=0;
	5. while (turn==0 && flag[0]==true);
	6. Critical Section
	7. flag[1]=false;
	}

Outcome:

1. Mutual exclusion is satisfied.
2. Progress is satisfied.
3. Bounded Waiting is satisfied.
4. Restricted to two process only.

Semaphores

- It is a synchronization tool.
- It gives solution for n processes.
- A semaphore is an integer variable that apart from initialization, is accessed only through two standard atomic operations.
- Two standard operation: wait(), signal().
- Less complicated.
- Counting semaphore-integer value can range over an unrestricted domain.
- Binary semaphore-integer value can range only between 0 and 1.

Semaphores for Critical Section Problem

P1….Pn, int s=1;	**Wait(s)**	**Signal(s)**

do
{
Wait(s);
Critical Section;
Signal(s);
Remainder section;
}
While (true);
	Wait(s)
	{
	While(s<=0);
	s=s-1;
	}
		Signal (s)
		{
		s=s+1;
		}

1. Support mutual exclusion.
2. Support progress.
3. Does not support bounded waiting.

Semaphores for deciding order of execution

Suppose we have three process ; P1 (calculating Area), P2 (calculating Radius), P3 (calculating Cost for Area). In this case we need order of execution; P2->P1->P3. Initialize s1=0, s2=0		

P1
 	
Wait (s1);
Code of process P1;
Signal (s2);
	P2
	
	Code of P2;
	Signal(s1);
		P3
		
		Code of P3;

Semaphores for managing multiple instances of resources

Suppose we have total five instances of printer.
In this case we need to initialize
s=5;

```
do
{
Wait(s);
Critical Section;
Signal(s);
Remainder section;
}
While (true);
```

Thank you

For any query contact at upasana.pandey@imsec.ac.in