CS & IT ENGINEERING

Operating System

Process Synchronization

Lecture - 04

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Mutual Exclusion

Topic Progress

Topic Bounded Waiting

Topic Two-Process Solution for Critical Section

Topics to be covered....

Topic Two-Process Solution for Critical Section

Topic Synchronization Hardware

Topic Test-And-Set()

Topic Swap()

Topic : Solution of Critical Section Problem

Mutual Exclusion:

If one process is executing the critical section, then other process is not allowed to enter into critical section.

Topic : Solution of Critical Section Problem

Progress:

If no any process is in critical section and any process wants to enter into critical section, then the process must be allowed.

Topic : Solution of Critical Section Problem

Bounded Waiting:

If a process p1 is executing in critical section and other process p2 is waiting for critical section, then the waiting time of p2 must be bounded. Which means p1 must not enter in to critical section again and again by keeping p2 in waiting for long.

Solution 1

```
Boolean lock=false;
```

P1

```
while(true)
{
    while(lock);
    lock=true;
        //CS
    lock=false;
    RS;
}
```

P2

```
while(true)
{
    while(lock);
    lock=true;
        //CS
    lock=false;
    RS;
}
```

X Mutual Exclusion

✓ Progress

X Bounded waiting

H.W:-

P1, P2 can starve ?

Progress:-

lock = false

Case 1 :-

only process p1 comes

lock = ~~f~~ T

p1 can enter into c.s.

Case 2 :-

only process p2 comes

lock = ~~f~~ T

p2 can enter into c.s.

Progress satisfied

Bounded waiting :-

Case :-

P1 is in CS and P2 is waiting for CS

lock = ~~False~~ ~~True~~ ~~False~~ True

P0, P1 both can suffer from starvation

✓ mutual Exclusion

X progress

✓ Bounded waiting

Solution 2

```
int turn=0;   // shared variable
```

P0

```
while(true)
{
    while(turn!=0);
    CS
    turn=1;
    RS;
}
```

P0 enters into C.S. when turn = 0

P1

```
while(true)
{
    while(turn!=1);
    CS
    turn=0;
    RS;
}
```

P1 enters into CS when turn = 1

Both processes run in strict alternation

Mutual Exclusion:-

turn = ~~0~~ 1

satisfied

At a time turn value can be either 0 or 1; hence only one of P0, P1 can enter into critical section.

Progress:-

Case 1:-

only P1 comes first and wants to enter into C.S.

turn = 0

P1 can not enter into C.S.

Case 2:-

only P0 comes first

turn = ~~0~~ 1 after C.S.

then P0 again cannot enter into C.S.

Progress not satisfied

Bounded waiting :-

processes will run in strict alternation hence one process can not enter into c.s. 2 times back to back.

Peterson's Solution :-

Flag[i] ⇒ indicates that process P_i wants to enter into c.s. or not.

turn ⇒ indicates priority

Process P0 Flag[0]

—"— P1 Flag[1]

Peterson's Solution

shared
```
Boolean Flag[2] = {false, false};
int turn;
```

P0

```
while(true) {
  Flag[0]=true;
  turn=1;
  while(Flag[1] && turn==1);
    CS
  Flag[0]=False;
    RS;
}
```

P1

```
while(true){
  Flag[1]=true;
  turn=0;
  while(Flag[0] && turn==0);
    CS
  Flag[1]=False;
    RS;
}
```

✓ M.E

✓ Progress

✓ Bounded waiting

H.W.

P0, P1 starve?

Mutual exclusion:-

at a time turn can be either 0 or 1; hence only one process Can enter into c.s. at a time.

mutual Exclusion satisfied

Case1:-

Flag[0] = ~~f~~ T

Flag[1] = ~~f~~ T

Turn = ~~1~~ 0

Progress:-

Case 1:-

only P0 Comes

P0 Can enter into C.S.

Case 2:-

only P1 comes

P1 Can enter into C.S.

Bounded waiting :-

P0 in CS, P1 waiting for CS.
P0 comes out and wants to enter again in CS

⇓

✓ Bounded waiting

flag[0] = ~~F~~ ~~T~~ ~~F~~ T

flag[1] = ~~F~~ T

turn = ~~1~~ ~~0~~ 1 ✓

2 mins Summary

Topic Mutual Exclusion

Topic Progress

Topic Bounded Waiting

Topic Two-Process Solution for Critical Section

Happy Learning

THANK - YOU