Synchoronization

ICS lec 8

Race Condition

- if the bo many variables accessing Some Variable at a time it will give Wrong result. its Called race Condition. (out come depends on accessing order of the variable)

- We need Synchronization to avoid this. accessing many processes a single Shared variabl at a single time.

- to design a protocol to archeve Synchoronization we have Critical Section problem.

Critical Section problem.

- its a code Segment.
- placed in Side every process.
- Common Variables, update, remove all Stored in here.

Entry section → Critical Section → exit Section → rmainder Section.

- if a process entered to Critical Section and accessing and Shared variable. no other process Can access it. untill done.

Solution must meet 3 reqs.

01) Mutual Exclusion
02) Progress
03) bounded Waiting.

(01) Mutual exclusion.

- only one process can be executed at a time in Critical Sections

(02) Progress.

- Processes which in remainder Section Cant take ony decision about entering processes. those who entering only Can take decision.

(03) Bounded waiting.

- have a no of time that a process Can enter to critical Section.

Solutions.

01) Interrupt-based Solution
02) Software Solution.
03) Peterson's Solution.

01 Interrupt based Solution.

Entry : disable interrupt

Exit : enable

-not a good Solution.

~~02 Software Solution.~~

03 Peterson's Solution.

have 2 shared Variables.

- turn
- boolean flag [2]

turn - Shows whos turn to enter Critical S.

flag - Shows process ready to Enter

flag [i] = P_i → i ready

flag [j] = P_j → j ready

- will not work for the modern Computers. give unexpected results.

-to work With modern Computers it have a Memory Barrier. it allow 2 processes to be in CS at a Same time.

Date No

Memory barriers.

- Memory model - Verify different Computer architecture ~~makes to~~ how they give memory to each.

2 type of memory models.

01 Strongly ordered
02 weakly ordered.

- memory barrier force changes that made to visible all other processors.

- on here from how to get hardware support to get Synchronization.

Synchronization hardware

3 types of hardware Supports.

01 hardware instructions
02 Atomic variables.

- Atomic variables - updates without any interruption. execute without

- Mutex Locks

  - Software based tool that used to solve the cs. by the os designers.

    - boolean variable indicate if lock available or not.

    - get the lock do the work release the lock

    - to get the lock have to whait. it called Spin lock.

- Semaphore

  also a Software better than Mutex Looks.

  - Only accessible Using 2 atomic variables

    2 type of semaphore.

    (01) Counting semaphore (Unlimited)

    (02) Binary Semaphore. (limited)

Binary - only share 1 semaphore will be shared among processors.

Counting - used in finite processors.

- if semaphore = 0 all resources are used. other process have to wait unit it becom (> 0) to execute.

- can solve synchronization problems.

- Should not have wait ( ) and signal ( ) at same time in semaphore.

Problems of Semaphore:

- incorrect use of operations.

Monitors

- high level method that used to process synchronization.
  - Only 1 process at a time can be active in monitor.

Classical problems of Synchronization.

- these used to test newly proposed Synchronization schemes.
  1. Bounded buffer problem
  2. Readers & writer problem
  3. Dining Philosophers problem.