COS3 lektion 7

Nis Sarup

24. oktober 2010

7 Deadlocks

- Deadlock: When two processes are waiting for each other and neither can do anything before the other.

7.1 System Model

- System Model: Each resource must be REQUESTED before USE and RELEASED again.
- Deadlock if two processes holds a resource each and will not RELEASE it until it has gotten access to the one the other process holds.

7.2 Deadlock Characterization

- Deadlock: Processes never finish executing and system resources are tied up, preventing other jobs from starting.
- Necessary conditions:
  - Mutual exclusion:At least one resource must be held in a non-shareable mode.
  - Hold and Wait: A process must be holding at least one resource and be waiting to acquire additional resources currently held by other processes.
  - No preemption: Resources can only be released by the process holding it.
  - Circular wait: P1 waits for P2 which waits for P3 which waits for p4...

7.3 Methods for Handling Deadlocks

- A protocol to prevent and avoid deadlocks.
- Enable the system to detect and recover from deadlocks.
- Ignore and pretend deadlocks never happens.

7.4 Deadlock Prevention

- Ensure that at least one fo the four conditions for a deadlock cannot hold.

7.5 Deadlock Avoidance

- The system can identify which REQUESTs for resources would result in a deadlock and makes the requesting process wait until the REQUEST would no longer result in a deadlock.

7.6 Deadlock Detection

- Detection of deadlocks requires two thing:
  - A deadlock detecting algorithm.
  - An algorithm to recover from the deadlock.
- It is necessary to evaluate how often to run the detecting algorithm.
- Once every resource REQUESt would be too often.
- When CPU utilization drops below 40% is more appropriate.

7.7 Recovery from Deadlocks

- Process Termination:
  - Abort all deadlocked processes.
    - Expensive
    - Much will need to be recomputed.
  - Abort one deadlocked process until deadlock resolves.
    - Deadlock detecting algorithm must run after each termination.
    - System must decide which process to terminate from some parameters: priority, runtime, etc.
- Resource preempting has issues:
  - Which resource to preempt? Which parameters to consider.
  - Should the resource be rolled back?
  - How to prevent starvation (The process is always going to be preempted on the same resource).