Deadlocks Lecture :Chapter 6

Resources:

- resource: something a process uses
    - usually limited
- examples of computer resources
    - printers
    - semaphores/locks
    - memory
    - tables(in a database)
- Processes need access to resources in reasonable order
- two types of resources:
    - preemptable resources: can be take away from a process with no ill effects
        - can take away and nobody cares as long as its eventually put back
        - just causes a process to be blocked
    - nonpreemptable resources will cause the process to fail if taken away
- if process are in different parts of memory, then they don't share resources
- can't share disk drives, only one thing at a time
- CPU: handle contentions with process

Using resources:

- sequence of events required to use a resource
    - request the resource
    - use the resource
    - release the resource
- can't use the resource if request is denied
    - request process has options
        - block and wait for resource
        - continue (if possible) without it: may be able to use an alternate resource
        - process fails with error code
    - some of these may be able to prevent deadlock...
- If you take something important away your process could die
- ask for it, use it, give it away

When do deadlocks happen?

- suppose:
    - process 1 holds resource A snd requests resource B
    - process 2 holds B and requests A

    - both can be blocked, with neither able to receive
- Deadlocks occur:
    - process are granted exclusive access to devices or software constructs (resources)
    - each deadlocked process needs a resource held by another deadlocked process

What is a deadlock?

- formal definition: “a set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause”
- usually, the event is release of a currently held resource
- in deadlock, none of the processes can
    - run release resources
    - be awakened

Four conditions for deadlock:

- Mutual exclusion
    - each resource is assigned to at most one process
- Hold and wait:
    - a process holding resources can request more resources
- no preemption
    - previously granted resources cannot be forcibly taken away
- Circular wait
    - there must be a circular chain of 2 or more processes where each is waiting for a resource held by the next member of the chain

Resource allocation graphs

- resource allocation modeled by directed graphs
- example 1:
    - resource R assigned to process A
- example 2:
    - Process B is requesting/ waiting for resource S
- example 3:
    - Process C holds T, waiting for U
    - Process D holds U, waiting for T
    - C and D are in a deadlock!
- If there is a circular waits it does not mean there is a deadlock
    - you need all four conditions above to have a deadlock

Dealing with deadlocks

- how can the OS deal with deadlock?
    - ignore the problem altogether!
        - hopefully it'll never happen...
    - detect deadlock and recover from it
    - dynamically avoid deadlock
        - careful resource allocation
    - prevent deadlock
        - remove at least one of the four necessary conditions

The Ostrich Algorithm

- pretend there's no problem
- reasonable if
    - deadlocks occur very rarely
    - cost of prevention is high
- UNIX and Windows take this approach
    - resources (memory, CPU, disk space) are plentiful
    - deadlocks over such resources rarely occur
    - deadlocks typically …

Not getting into deadlocks

- Find ways to:
    - detect deadlock and revise it
    - stop it from happening in the first place