Deadlock strategies

Recorded lecture for 2/20/26

From before the exam: Deadlock

- Situation in which group of threads/processes all block forever
- Typically, each holds a resource that others are blocking on

http://minutillo.com/steve/weblog/2003/1/21/deadlock/, where it is attributed to "Chuck @ China" (http://chake.chinatefl.com/)

Deadlock avoidance: Eliminate possibility of deadlock

Banker's Algorithm: For each resource allocation, check if it makes system unsafe

Strategy 2: Detect and recover

- Periodically run a deadlock detection algorithm and kill an involved process if one is found

Strategy 3: Deadlock prevention

- Deny one of the conditions needed for deadlock:
  - Mutual exclusion: Resources assigned to <= 1 process at a time
  - Hold and wait: Process can hold resources while waiting for more
  - No preemption: System can’t take resources back
  - Circular wait: There can be a circular chain of processes waiting for each other’s resources

Which of the conditions is prevented by the scheme below?

Two-phase locking: Acquire all locks at once; release and backoff (wait and try again) if you can't. Then perform critical section and release everything

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Two-phase locking: Acquire all locks at once; release and backoff (wait and try again) if you can't. Then perform critical section and release everything

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Assign a number to every resource and require that processes request resources in order

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Assign a number to every resource and require that processes request resources in order

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Spooling: Instead of blocking for a resource such as a printer, write the operation to a buffer which the OS handles when possible

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Spooling: Instead of blocking for a resource such as a printer, write the operation to a buffer which the OS handles when possible

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Require processes to request all needed resources when they ask to start and only start processes whose needs can be met (more appropriate for hardware resources than locks...)

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Which of the conditions is prevented by the scheme below?

Require processes to request all needed resources when they ask to start and only start processes whose needs can be met (more appropriate for hardware resources than locks...)

A. Mutual exclusion (Resources assigned to <= 1 process at a time)
B. Hold and wait (Process can hold resources while waiting)
C. No preemption (System can't take resources back)
D. Circular waiting (Can be circular chain of processes waiting)
E. None of the above; deadlock can still occur

Strategy 4: The “Ostrich Algorithm”: Ignore the possibility of deadlock and reboot if it occurs

Strategy 4: The “Ostrich Algorithm”: Ignore the possibility of deadlock and reboot if it occurs

Which of the following is true about this strategy?

A. It’s cheap and easy to implement
B. How common could deadlocks be anyway? Enjoy life and don’t be a worrywart!
C. Ostriches are cute so it must be a good strategy
D. It is far too irresponsible to use on production systems
E. Not exactly one of the above

Strategy 4: The "Ostrich Algorithm": Ignore the possibility of deadlock and reboot if it occurs

Which of the following is true about this strategy?

A. It's cheap and easy to implement
B. How common could deadlocks be anyway? Enjoy life and don't be a worrywart!
C. Ostriches are cute so it must be a good strategy
D. It is far too irresponsible to use on production systems
E. Not exactly one of the above (A and B)

Which strategy is the most interesting?

A. Deadlock avoidance (Banker's algorithm)
B. Detect and recover
C. Deadlock prevention by removing mutual exclusion or hold and wait
D. Deadlock prevention by removing no preemption or circular waiting
E. Do nothing (Ostrich algorithm)