Deadlocks

Deadlock

- permanent blocking of set of processes that either compete for system resources or communicate with each other.

- Deadlock needs at least 2 processes and 2 resources.

There are four conditions that needs to be satisfied for a deadlock to happen.

1. Mutual Exclusion
2. Hold & wait
3. No preemption
4. Circular Wait

} necessary conditions but **not** sufficient.

- Joint Resource Diagram
  - illustrates progress of processes competing for resources
  - If execution path enters a fatal region in a JRD, deadlock can not be avoided.

Resource

↳ Reusable Resources

- A resource that can only be used by one process at a time safely.

↳ Consumable resources

- A resource that can be created (produced) and destroyed (consumed).

Ex: messages, signals.

Common Strategies to deal with deadlocks.

Deadlock prevention

- Disallows one of the three conditions necessary for deadlocks or prevent circular waiting by employing some type of policy.

Deadlock Avoidance

- Do not grant resource requests if the granting the resource would lead to ~~t~~ a deadlock

- less restrictive than deadlock prevention.

Deadlock Detection

- Grant resource requests when possible, but periodically check for deadlocks and take action to recover.

Deadlock Avoidance

~~- Banker's Algorithm~~

- Process Initiation Denial

  Do not start a process if the demand of the process will lead to a deadlock

- Resource Allocation Denial (Banker's Algorithm)

  Do not grant resource allocation if the resource allocation will lead to a deadlock.

Deadlock Detection

- Deadlock Detection Alogorithm.

Dining Philosopher's problem

- n philosophers
- n forks
- philosophers think or eat
- How can they eat without starving or running into a deadlock?

  - Semaphore solution
  - Monitor solution.

Resource Allocation Graph (RAG)

- Node for each process
- Node for each instance of resource
- Edge from process to resource instance, if the process requests the resource.
- Edge from resource to process, if resource is allocated to the process
- ~~Circle~~ cycle indicates a ~~the~~ deadlock.

Deadlock prevention

- No mutual Exclusion
  - Hard to remove ME as it will lead to inconsistant results.

- No Hold & wait
  - can ask process to request all resources at the beginning.

- ~~No~~ preemption
  - can ~~deny~~ ask process to release resources if resource request was denied.
  - If a process request a resource that is held by another process, we can ask the process to release the resource, if that particular resource has lower priority.

Producer / consumer problem

- Shared buffer
- multiple producers
- One consumer.

1. Producer should not ~~enter~~ add items to the buffer, if the buffer is full.

2. Consumer should not consume if the buffer is empty

3. Producers or consumers must not modify the buffer at the same time.

   - Binary Semaphore solution

   - Counting semaphore solution.

Readers / Writers problem

- Shared buffer.
- Any # of Readers should be able to read at the same time.
- Only one writer should write at a time.
- If writer is writing no reader should read.

  - Reader priority sol.
  - Writer priority sol.