EE 360P Vijay Garg Spring 2003

Deadlocks

1 Introduction

A process is assumed to make calls to the system for using resources in the following sequence: *request*, **use**, and *release*.

2 Conditions for Deadlock

Mutual Exclusion The resources cannot be accessed in shared mode. For some resources, one can break this condition by allowing multiple processes to access it. For example, an object may be accessed in read mode by multiple processes. Not possible in general.

Hold and wait A process is allowed to wait while holding resources. This condition can be avoided by requiring processes to get all resources at one time. This may lead to slowing down the system where a process may have to wait for a long time to get all the resources, even though some of them may not be necessary to start the operation. This leads to under-utilization of resources.

The other strategy to prevent this condition is by requiring processes to request resources only when they do not hold any resource. If a process needs additional resources, it is required to release existing resources. This strategy is not possible for many operations which require simultaneous use of resources.

No Preemption Resources cannot be taken away from processes that are waiting. In some cases, this condition can be prevented by taking back the resources by aborting or rolling back a process.

Circular Wait A cycle of processes exist such that each process waits for a resource held by the next process in the cycle. One can prevent this condition by linearly ordering all resources and requiring processes to request resources in increasing order.

3 Dynamic Deadlock Prevention

4 Resource Allocation Graph

A Resource Allocation Graph is a directed graph with vertices as processes and resources. There are two types of edges - request edges and assignment edges. A request edge is from a process P_i to R_j if P_i has requested a resource of type R_j. An assignment edge is from a resource R_j to P_i if a resource of type R_j has been assigned to process P_i.

First assume that there is exactly one instance of each resource type. In this case, the system is in a deadlock state iff there is a directed cycle in the resource allocation graph.

When the multiplicity of resources can be more than one, then presence of a cycle is only a necessary condition but not sufficient for deadlock.

5 Banker's algorithm

Assume that there are n processes competing for m resource types. The following data structures are used:

- $Total[i]$: Total number of resources in the system
- $Avail[i]$: Total number of resources not assigned to any process
- $Need[i, j]$: Need of process i for resources of type j
- $Alloc[i, j]$: allocation of process i for resources of type j

Safe State: A state is safe if there exists a sequence of operations in which all of the processes can run to completion.

To determine if the state is safe, one can use the following procedure.

```
while not all processes finished do
  find a process that can be completed;
  if none then the system is in unsafe state;
  return resources allocated to that process;
end while;
```

To determine if a request should be satisfied the system simply checks if the resulting state of the system is safe or not.