Chapter 7: Deadlocks

a group of processes requesting and using resources in a manner which causes computing grid lock, so that all of these processes are halted

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Detection
- Recovery from Deadlock
- Combined Approach to Deadlock Handling

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion:** only one process at a time can use a resource.
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes.
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular wait**

Methods for Handling Deadlocks

- Ensure that the system will *never* enter a deadlock state: prevention and avoidance

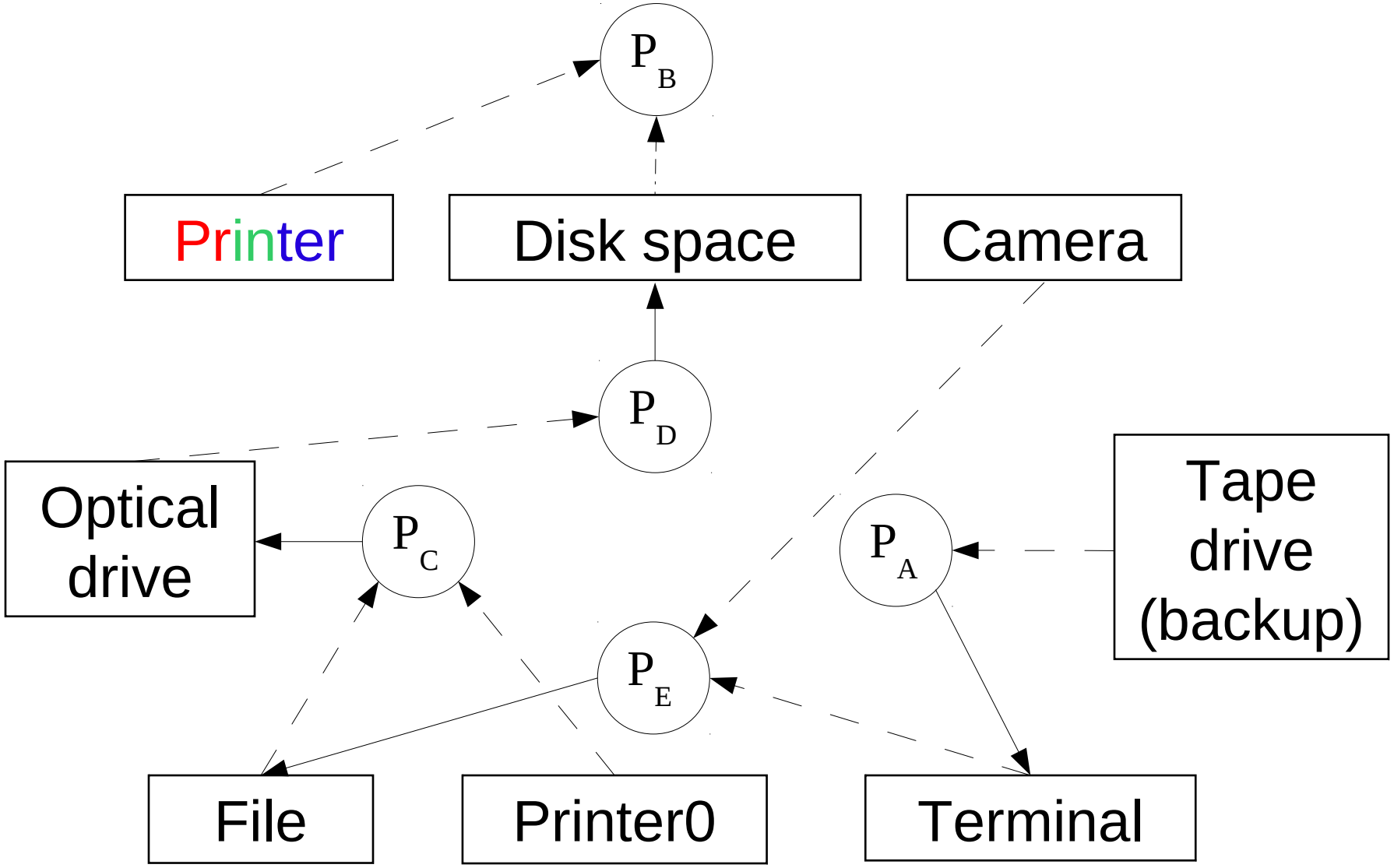
- Allow the system to enter a deadlock state and then recover.

- do nothing: ignore the problem and pretend that deadlocks never occur in the system; used by popular operating systems, including UNIX, Linux, Windows, Mac OsX

Deadlock Detection

- Allow system to enter deadlock state
- Detection algorithm: graph based algorithm
- Recovery scheme: resource preemption or process termination

Resource allocation graph



Deadlock Prevention

Restrain the ways request can be made.

- **Mutual Exclusion** – not required for sharable resources; must hold for nonsharable resources.

- **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources.
  - Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none.
  - Low resource utilization; possible starvation; unnecessary delays

Deadlock Prevention (Cont.)

- **No Preemption** –
  - If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released.
  - Preempted resources are added to the list of resources for which the process is waiting.
  - Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.
  - loss of work, long delays, starvation
- **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.
  - added burdens on programmers, non-intuitive, rigid and not user-friendly environment

Deadlock Avoidance

Requires that the system acquires additional *a priori* information on resource usage

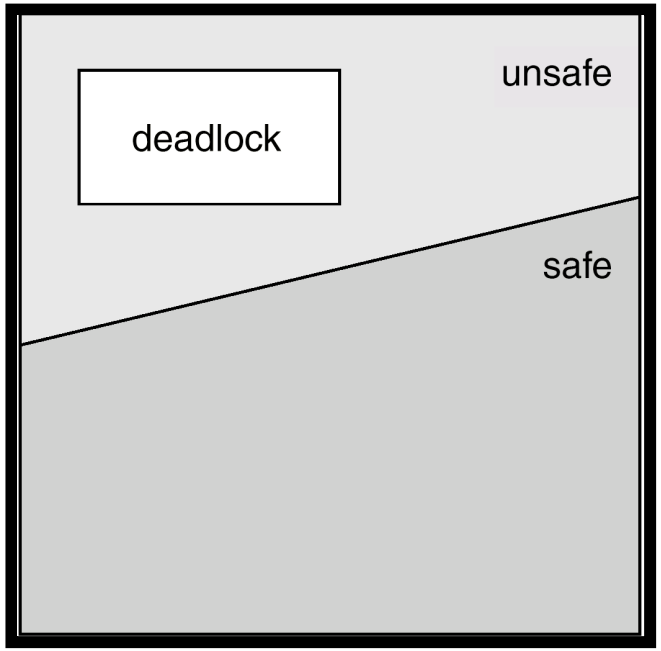
- Simplest and most useful model requires that each process declare the *maximum number* of resources of each type that it may need.

- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition.

- Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes.

Safe, Unsafe , Deadlock State

- When a process requests an available resource, system must decide if immediate allocation leaves the system in a *safe state*.
- System is in safe state if there exists a safe sequence of all processes – a sequence which guarantees that all processes will execution to completion
- If a system is in safe state ⇒ no deadlocks.
- If a system is in unsafe state ⇒ possibility of deadlock.
- Avoidance ⇒ ensure that a system will never enter an unsafe state.



Banker’s Algorithm

- Multiple instances of resources
- Each process must a priori claim maximum use.
- When a process requests a resource and a safe state does not exist, the process will block.
- When a process completes its execution, it must return all resources (to the pool) in a finite amount of time.

Example of Banker's Algorithm

- 5 processes
- 10 equivalent disk spaces
- 7 equivalent floppy drives
- 4 modems
- Snapshot at time T_0:

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	0	(2)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	1	(1)	2	1	(1)	2
P4	0	(4)	4	2	(1)	3	0	(3)	3
	pool 3			pool 2			pool 2		

Example

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	0	(2)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	1	(1)	2	1	(1)	2
P4	0	(4)	4	2	(1)	3	0	(3)	3
	pool 3			pool 2			pool 2		

- P1,P3,P4,P2,P0 is a safe state
- P3,P1,P0,P2,P4 is also a safe state
- Any other safe states?

Example: P4 requests a floppy drive!

Test: Is there a safe state, when a floppy is allocated to P4?

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	0	(2)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	1	(1)	2	1	(1)	2
P4	0	(4)	4	2	(1)	3	0	(3)	3
	pool 3			pool 2			pool 2		

Ex (cont): P1 requests a modem!

Test: Is there a safe state, when a modem is allocated to P1?

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	0	(2)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	1	(1)	2	1	(1)	2
P4	0	(4)	4	3	(0)	3	0	(3)	3
	pool 3			pool 1			pool 2		

Ex (cont): P3 requests a second floppy!

Test: Is there a safe state, when a floppy is allocated to P3?

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	1	(1)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	1	(1)	2	1	(1)	2
P4	0	(4)	4	3	(0)	3	0	(3)	3
	pool 3			pool 1			pool 1		

Ex (cont): P1 requests a second modem!

Test: Can a safe state be found, when a modem is allocated to P1?

	disk spaces			floppy drives			modems		
	alloc	need	max	alloc	need	max	alloc	need	max
P0	1	(6)	7	0	(3)	3	1	(3)	4
P1	2	(2)	4	0	(2)	2	1	(1)	2
P2	3	(6)	9	2	(0)	2	0	(0)	0
P3	1	(0)	1	2	(0)	2	1	(1)	2
P4	0	(4)	4	3	(0)	3	0	(3)	3
	pool 3			pool 0			pool 1		

banker's algorithm

- heavy resource consumption/overhead – $O(mn^2)$ time complexity
- difficult to determine maximum resource use in advance – rigid environment
- resources cannot fail
- number of processes (rows) cannot

Combined Approach to Deadlock Handling

- Combine the three basic approaches
  - prevention
  - avoidance
  - detection

  allowing the use of the optimal approach for each of resources in the system.

- Partition resources into hierarchically ordered classes.

- Use most appropriate technique for