CSE 4/521 Introduction to Operating Systems

Lecture 11 – Deadlocks

(System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention)

Summer 2018

Overview

Objective:

1. To develop a description of deadlocks, which prevent sets of concurrent processes from completing their tasks
2. To present a number of different methods for preventing or avoiding deadlocks in a computer system

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

Recap

- Real-time CPU Scheduling
  - Rate monotonic scheduling, earliest deadline first scheduling
- Operating-System Examples
  - Scheduling in Linux , scheduling in Windows, scheduling in Solaris

Questions

1. What are the characteristics of rate monotonic scheduling and earliest deadline first scheduling? (Easy)
2. How does Linux decide the next task to run? (Medium)
3. Draw the CPU queueing diagram for EDS scheme:

P1 = 50
P2 = 80
t1 = 25
t2 = 35
Deadline is to complete before next period

Overview

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

Overview

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

System Model

- System consists of resources
- Resource types $R_1, R_2, \ldots, R_m$

  CPU cycles, memory space, I/O devices
- Each resource type R_i has W_i instances.
- Each process utilizes a resource as follows:
  - request
  - use
  - release

Overview

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

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: there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0.

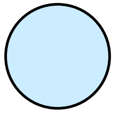
Deadlock Characterization : Resource-Allocation Graph

A set of vertices V and a set of edges E

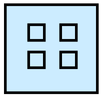
- V is partitioned into two types:
  - $P = \{P_1, P_2, \ldots, P_n\}$, the set consisting of all the processes in the system
  - $R = \{R_1, R_2, \ldots, R_m\}$, the set consisting of all resource types in the system
- request edge – directed edge $P_i \rightarrow R_j$
- assignment edge – directed edge $R_j \rightarrow P_i$

Deadlock Characterization : Resource-Allocation Graph

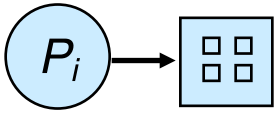
- Process



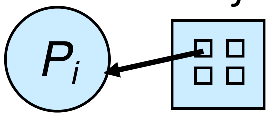
- Resource Type with 4 instances



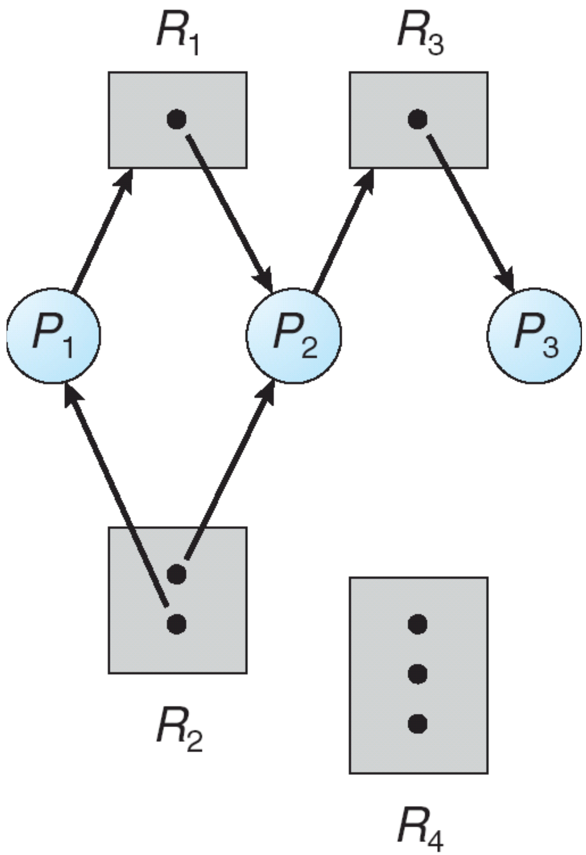
- P_i requests instance of R_j



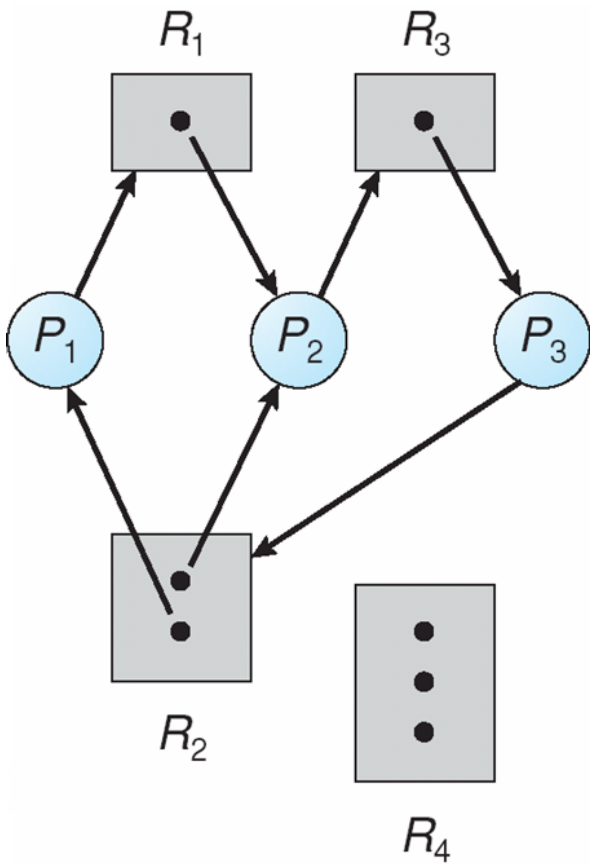
- P_i is holding an instance of R_j



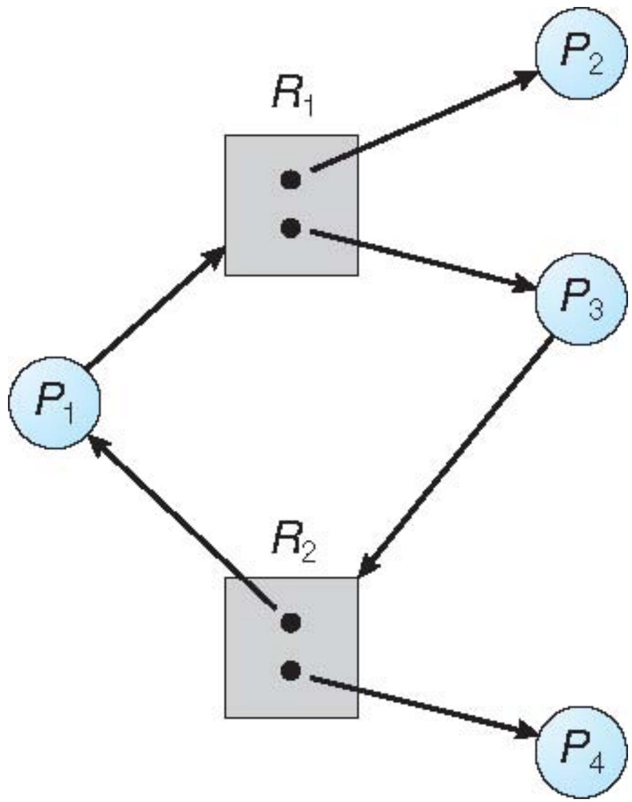
Deadlock Characterization : Resource-Allocation Graph Example



Deadlock Characterization : Resource-Allocation Graph with A Deadlock



Deadlock Characterization : Graph with a Cycle But No Deadlock



Deadlock Characterization : Basic Facts

- If graph contains no cycles $\Rightarrow$ no deadlock
- If graph contains a cycle $\Rightarrow$
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Overview

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

Method for Handling Deadlocks

- Ensure that the system will never enter a deadlock state:
  - Deadlock prevention
  - Deadlock avoidance
- Allow the system to enter a deadlock state and then recover
- Ignore the problem and pretend that deadlocks never occur in the system

Overview

- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention

Deadlock Prevention

Restrain the ways request can be made

- Mutual Exclusion – not required for sharable resources (e.g., read-only files); must hold for non-sharable 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 allocated to it.
  - Low resource utilization; starvation possible

Deadlock Prevention

- 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
- Circular Wait – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration

Deadlock Prevention : Deadlock Example

```
/* thread one runs in this function */
void *do_work_one(void *param)
{
   pthread_mutex_lock(&first_mutex);
   pthread_mutex_lock(&second_mutex);
   /** * Do some work */
   pthread_mutex_unlock(&second_mutex);
   pthread_mutex_unlock(&first_mutex);
   pthread_exit(0);
}
/* thread two runs in this function */
void *do_work_two(void *param)
{
   pthread_mutex_lock(&second_mutex);
   pthread_mutex_lock(&first_mutex);
   /** * Do some work */
   pthread_mutex_unlock(&first_mutex);
   pthread_mutex_unlock(&second_mutex);
   pthread_exit(0);
}
```

Deadlock Prevention : Deadlock Example with Lock Ordering

```
void transaction(Account from,
Account to, double amount)
{
   mutex lock1, lock2;
   lock1 = get_lock(from);
   lock2 = get_lock(to);
   acquire(lock1);
      acquire(lock2);
         withdraw(from, amount);
         deposit(to, amount);
      release(lock2);
   release(lock1);
}
```

Transactions 1 and 2 execute concurrently. Transaction 1 transfers $25 from account A to account B, and Transaction 2 transfers $50 from account B to account A

Credits for slides

Silberschatz, Galvin and Gagne