Chapter 4: Process Management

Operating system

TS. Do Tien Dung

dungdt@ptit.edu.vn

Posts and Telecommunications Institute of Technology

Faculty of Information Technology 1

Faculty of Information Technology 1
Posts and Telecommunication Institute of Technology

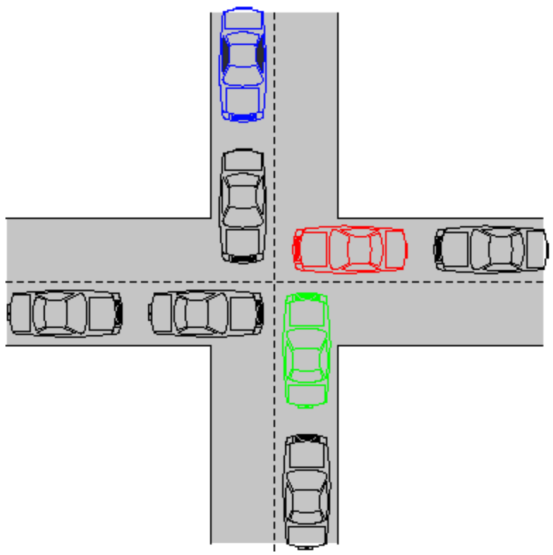
August 15, 2022

Content

Chapter 4 Content

1. Definitions
2. Thread
3. Scheduling
4. Synchronize the simultanous processes
5. Deadlock and starvation

Deadlockis situation in which a group of processes, whether competing for resources or cooperating, must stop indefinitely
Because the process must wait for an event that can only be generated by another process that is also in a waiting state



Example: Two processes P and Q execute simultaneously. Each process needs to use two resources X and Y simultaneously for a period of time:

- X and Y are only capable of serving 1 process at a time
- Each process needs to perform a Request operation to request resources
- Once done, the process releases resources using Release.

Tiến trình P	**Tiến trình Q**
…	…
Request X	Request Y
…	…
Request Y	Request X
…	…
Release X	Release Y
…	…
Release Y	Release X
…	…

Suppose that as a result of schedule, the execution order of the two processes is as follows:

...
P: Request X
Q: Requets Y
...
P: Request Y
Q: Request X
...

- The operation "P:Request Y"will cause P to wait until Q releases Y.
- Q must stop at operation "Q: Request X"to wait for resource X held by P.
- *As a result, the two processes fell into deadlock.*

Deadlock occurs when the following four conditions are simultaneously satisfied:

- **Mutual exclusion**: there is a dangerous resource, used by only 1 process at a time. Other processes requesting this resource will have to stop waiting for the resource to be released.
- **Hold and wait**: the process of holding a resource while waiting, for example waiting to be allocated another resource.
- **No preemption**: resources held by a process cannot be redistributed to another process unless the holding process voluntarily releases the resource
- **Circular wait**: there exists a process group P1, P2, . . . , Pn such that P1 waits for a resource held by P2, P2 waits for a resource held by P3, . . . , Pn waits for a resource held by P1 .

Deadlock solutions:

- **Deadlock prevention**: ensuring that one of the four deadlock conditions is never satisfied
- **deadlock avoidance**: allowing some deadlock condition to be satisfied but ensuring that a deadlock point is not reached
- P**Detection and resolution** (deadlock detection): allows deadlock to occur, detects deadlock, and restores the system to a deadlock-free state.

To prevent deadlock, you must ensure that at least one of four conditions does not occur:

- Mutual exclusion
- Hold and wait
- No preemption
- Circular wait

Mutual exclusion:

- Mutual exclusion can be avoided with resources that allow multiple processes to use them at the same time, such as files in read mode.
- However, in reality, there are always resources that cannot be shared at the same time.
- *The condition of mutual exclusion cannot be prevented.*

Hold and wait: There are two ways to prevent this condition:

- Method 1:
  - Requires the process to receive all necessary resources before continuing
  - If it does not receive enough, the process is blocked to wait until it can receive enough resources
- Method 2
  - A process may only request a resource if it does not hold another resource
  - Before requesting additional resources, the process must release the allocated resource and request it again (if necessary) with the new resource.

No preemption:

- Method 1:
  - When a process requests a resource but cannot because it has been allocated, the HDH will reclaim all the resources it is holding.
  - The process can only continue after retrieving the old resource along with the newly requested resource
- Method 2:
  - When the process requests resources, if available, they will be allocated immediately
  - If a resource is held by another process and this process is waiting for more resources, it will be reclaimed to give to the requesting process.
  - If the above two conditions are not met, the process requesting the resource must wait

Circular wait:

- Determine the order for resource types and only allow the process to request resources such that the resource the process requests later is in greater order than the resource it requests first
- Suppose in the system there are n types of resources denoted R1, R2, ..., Rn
- Suppose these types of resources are sorted in ascending order by index
- If the process has requested some resource of the form Ri then the process is only allowed to request the resource of the form Rj if $j > i$
- If a process needs multiple resources of the same type, the process must request all of those resources at the same time

Deadlock prevention solutions focus on using rules or constraints to prevent deadlock conditions
Disadvantages: makes resource use less efficient, reduces process performance.
Deadlock prevention:

- Use rules or constraints when allocating resources to prevent deadlock conditions
- Uses resources inefficiently, reducing process performance

Avoid deadlock:

- Allow the first three conditions to occur and only ensure that deadlock is never reached
- Each process's resource request will be considered and decided depending on the specific situation
- The OS asks the process to provide information about resource usage (maximum amount of resources the process needs to use).

Lender algorithm: *Lender algorithm because of the analogy between deciding to lend money and allocating resources in a computer.*

- A good lender is one who lends a lot
- However, when lending in excess of the actual amount of money, there will be a risk because each borrower cannot borrow the necessary amount to serve the business and therefore, both the bank and the lender will fall into a deadlock situation. occlusion.

- The OS asks the process for information about its resource usage and uses this information when allocating
- When the process wants to initialize, it announces the type of resource and the maximum number of resources each type will require
- If the number of requests does not exceed the system capacity, the process will be initiated
- The state is determined by the current resource usage status in the system:
  - Maximum number of resources requested by the process:
    - In matrix form M[n][m]: n is the number of processes, m: number of resources
    - M[i][j]: maximum number of resources of type j that process i requests

- The state is determined by the current resource usage status in the system:
  - Number of remaining resources:
    - As vector A[m]
    - A[j] is the amount of resource of type j remaining and available for allocation
  - Amount of resources allocated to each process:
    - As matrix D[n][m]
    - D[i][j] is the amount of resource of type j allocated to process i
  - Amount of resources still needed
    - As matrix C[n][m]
    - C[i][j]=M[i][j]-D[i][j] is the amount of resource of type j that process i still needs to allocate

Lender algorithm:
Safe state: state from which there is at least one allocation solution such that deadlock does not occur.

- How to avoid deadlock:
  - When a process requests a resource, the system assumes the resource has been granted
  - Update the status and determine if the status is safe?
    - If safe, resources will be granted for real
    - Otherwise, the process is blocked and waits until it can be safely allocated

Lender algorithm:

Algorithm to determine the safe state

1. Khai báo mảng W kích thước m và mảng F kích thước n. (F[i] chứa tình trạng tiến trình thứ i đã kết thúc hay chưa, W chứa lượng tài nguyên còn lại)

   Khởi tạo W=A và F[i]=false (i=0,…,n-1)

2. Tìm i sao cho:

   F[i] = false và C[i][j] ≤ W[j] với mọi j=0,…,m-1

   Nếu không có i như vậy thì chuyển sang bước 4

3. a) W = W + D[i]

   b) F[i] = true

   c) Quay lại bước 2

4. If F[i] = true với mọi i =0,…,n-1 thì trạng thái an toàn

   Else trạng thái không an toàn

Lender algorithm: *The system has 3 types of resources X, Y, Z with initial quantity X=10, Y=5, Z=7*

- 4 processes P1, P2, P3, P4 have maximum resource requirements given in the table

		X	Y	Z
	P1	7	5	3
	P2	3	2	2
M	P3	9	0	2
	P4	2	2	2

Yêu cầu tối đa

- Consider the following state of the system:
  - is a safe state
  - It is possible to find an allocation method that does not lead to a deadlock, for example: P2, P4, P1, P3

Lender algorithm: *The system has 3 types of resources X, Y, Z with initial quantity X=10, Y=5, Z=7*

This state is a safe state because it is possible to find a way to allocate that does not lead to deadlock, For example, in order P2, P4, P3, P1.

D

	X	Y	Z
P1	0	1	0
P2	2	0	0
P3	3	0	2
P4	2	1	1

Đã cấp

A

X	Y	Z
3	3	4

Còn lại

C=M-D

	X	Y	Z
P1	7	4	3
P2	1	2	2
P3	6	0	0
P4	0	1	1

Còn cần cấp

Lender algorithm: *The system has 3 types of resources X, Y, Z with initial quantity X=10, Y=5, Z=7*
P1 requests allocation of 3 resources of type Y, i.e. request = (0,3,0). If the request is satisfied, the system moves to the state:

X	Y	Z
3	0	4

Còn lại

	X	Y	Z
P1	7	1	3
P2	1	2	2
P3	6	0	0
P4	0	1	1

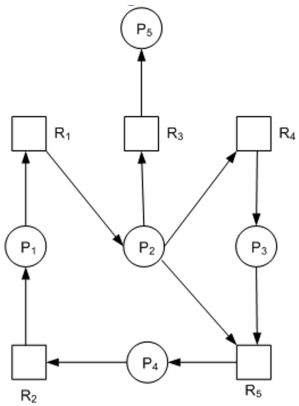
Còn cần cấp

Trạng thái không an toàn nên yêu cầu (0,3,0) bị từ chối.

- The system does not take measures to prevent/avoid possible deadlock
- The system periodically checks to detect whether there is a deadlock or not?
- If so, the system will process to restore the deadlock-free state

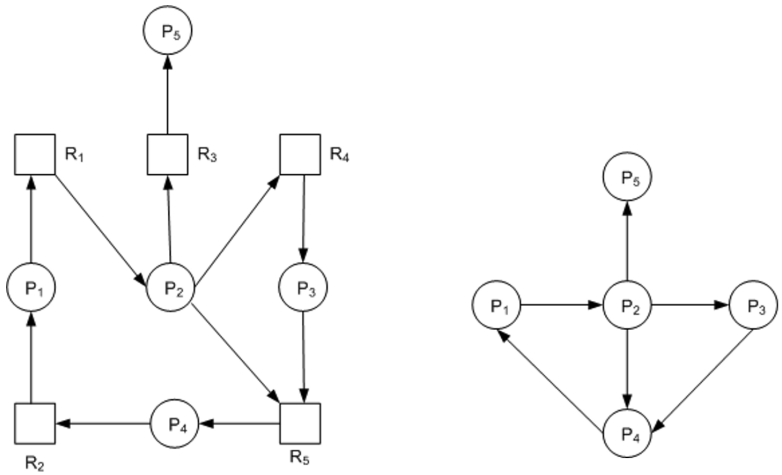
Detect deadlock:

- In case each type of resource has only one resource => use a graph to represent the waiting relationship between processes
- Build a resource allocation graph:
  - Nodes are processes and resources
  - A resource is connected to a process by a directed arc if the resource is allocated to that process
  - A process is connected to a resource by a directed arc if the process is being allocated that resource



Waiting graph:

- Constructed from the resource allocation graph by removing nodes corresponding to resources and entering arcs passing through the removed node
- Allows detection of a circular waiting process condition which is a sufficient condition for deadlock generation
- Use cycle detection algorithm on directed graph to detect deadlock on waiting graph.



Deadlock detection time:

- A deadlock can only occur after a process requests a resource and is not satisfied
- Run the deadlock detection algorithm every time a resource allocation request is not satisfied
- Allows deadlock detection as soon as it occurs
- Running frequently reduces system performance
- Reduce the frequency of running the deadlock detection algorithm:
  - After each cycle from a few tens of minutes to a few hours
  - When there are some signs such as CPU usage dropping below a certain threshold

- Handling deadlock:
  - Terminate all deadlocked processes
  - Terminate each deadlocked process in turn until the deadlock is gone:
    - HDH must rerun the deadlock detection algorithm after terminating a process
    - HDH can choose the order to end the process based on certain criteria
  - Restore the process to the point in time before deadlock then let the processes start again from this point:
    - Requires the HDH to store state so it can perform rollback and restore to previous checkpoints
    - When restarted, the processes may fall into deadlock again
  - Repeatedly reclaim resources from deadlocked processes until the deadlock ends

Summary

Chapter 4 Process Management

- Synchronize concurrent processes
- Deadlock and hunger

Summary

- Presenting major exercises
- Correcting exam questions
- Questions

Questions and exercises

1. **Question 1**: Consider the system's resource allocation status as follows: P2 requests allocation of 1 resource Y, 2 resources Z. Use the lender algorithm to determine whether P2's request is met or not?

	X	Y	Z
P1	0	1	0
P2	2	0	0
P3	3	0	2
P4	2	1	1

Đã cấp

X	Y	Z
3	3	4

Còn lại

	X	Y	Z
P1	7	4	3
P2	1	2	2
P3	6	0	0
P4	0	1	1

Còn cần cấp

2. **Question 2**: Consider the system's resource allocation status as follows: P1 requests allocation of 1 resource X, 1 resource Y, 2 resources Z. Use the lender algorithm to determine whether P1's request is met or not? What is status?

	X	Y	Z
P1	0	1	0
P2	2	0	0
P3	3	0	2
P4	2	1	1

Đã cấp

X	Y	Z
3	3	4

Còn lại

	X	Y	Z
P1	7	4	3
P2	1	2	2
P3	6	0	0
P4	0	1	1

Còn cần cấp