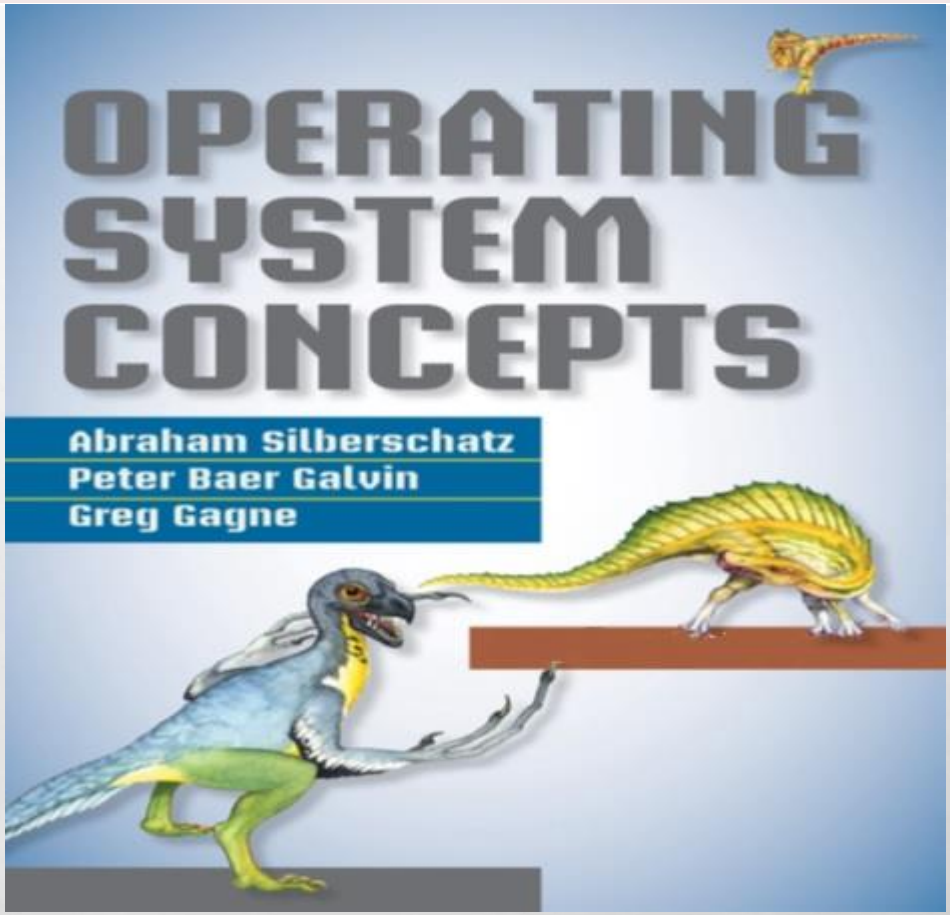
Mustansiriayah University
Collage of Education
Computers Science Department

Chapter Six
Deadlock
Part 3



Fourth Class

Dr. Hesham Adnan ALABBASI

2019-2020

6.5. Deadlock Avoidance

- Deadlock avoidance requires that the system has some additional a **prior** information available

يتطلب تجنب الـ Deadlock أن يتوفر لدى النظام بعض المعلومات الإضافية (المسبقة) المتاحة

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

يتطلب النموذج الأبسط والأكثر فائدة أن تعلن كل process عن العدد الأقصى لكل نوع قد تحتاجه من الموارد

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

تفحص خوارزمية تجنب الـ Deadlock ديناميكيًا حالة تخصيص الموارد لضمان عدم وجود حالة شرط circular-wait

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

يتم تعريف حالة تخصيص الموارد من خلال عدد الموارد المتاحة والمخصصة ، والحد الأقصى المطلوب من الـ processes

6.5.1. Safe State

A state is **safe** if **the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock**. More formally, a system is in a safe state **only if there exists a safe sequence**.

تكون الحالة آمنة إذا كان بإمكان النظام تخصيص الموارد لكل process (حتى الحد الأقصى) بترتيب أو تسلسل محدد بحيث يبقى النظام يستطيع تجنب الـ Deadlock . بشكل أكثر رسمية ، يكون النظام في حالة آمنة فقط إذا كان هناك تسلسل آمن.

- The system is in a safe state if there exist a sequence of processes <*P1*, *P2*, ..., *Pn*> of all the processes in the systems such that for each *Pi*, the resource requests that *Pi*, can still request, can be satisfied by the currently available resources plus the resources held by all *Pj*, with *j* < *i*.

يكون النظام في حالة آمنة إذا كان هناك تسلسل للprocesses < P1، P2، ...، Pn> لجميع الـ processes في الأنظمة وكما في التالي:
لكل الـ process Pi يستطيع ان يبقى يطلب موارد والتي يمكن تلبيتها من الموارد المتوفرة حالية اضافة الى الموارد التي يحتفظ او يمسك بها الـ process Pj على ان يكون i> j .

6.5.1. Safe State Cont.

That is:

- If P_i resource needs are not immediately available, then P_i can wait until all P_j have finished

اذا لم تكن الموارد التي تحتاجها Pi متوفرة حالياً, فيجب عليها الانتظار (Pi) الى ان تنهي كل الـ Pj .

- When P_j is finished, P_i can obtain needed resources, execute, return allocated resources, and terminate

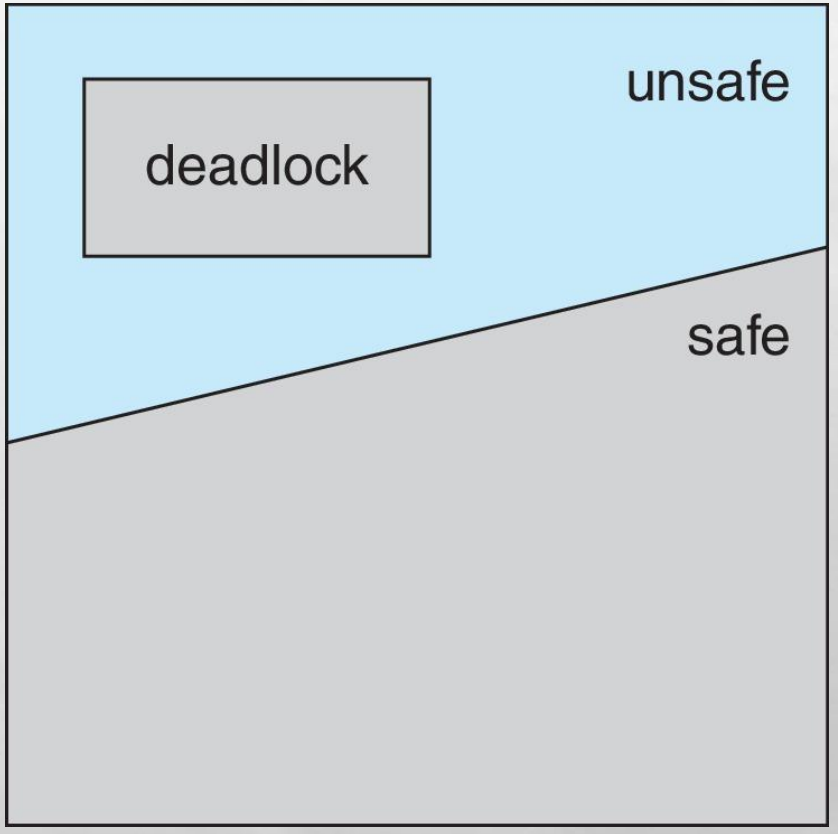
عندما تنتهي الـ Pj, الـ Pi تستطيع الحصول على الموارد التي تحتاجها وتنفذ عملها ومن ثم تطلق او تحرر جميع الموارد المحجوزة لها وتنتهي.

- When P_i terminates, P_{i+1} can obtain its needed resources, and so on

عندما تنتهي الـ Pj, الـ Pi+1 تستطيع الحصول على الموارد التي تحتاجها ويستمر التنفيذ بهذه الصيغة

Basic Facts

- If a system is in a safe state ➔ no deadlock
- If a system is in unsafe state ➔ possibly of deadlock
- Not all unsafe states are deadlocks.



- To illustrate, Consider a system with **12 magnetic tape drives** and **3 processes** (P_0, P_1, *and* P_2)
  - Process P_0 requires **10 tape drives**, (require= يطلب او يحتاج)
  - Process P_1 may need as many as **4 tape drives**, and
  - Process P_2 may need up to **9 tape drives**.
- Suppose that, at time t_0:
  - Process P_0 is holding 5 tape drives, (holding= يحتفظ أو يمسك)
  - Process P_1 is holding 2 tape drives, and
  - Process P_2 is holding 2 tape drives.

(Thus, there are **3 free tape drives**.)

Available resources = total resources in the system − total holding resources

= 12 – 9 = 3

Current Needs= Maximum Needs- Allocation.

	Maximum Needs (requires)	Current Needs	Allocation (holding)
P_0	10	5	5
P_1	4	2	2
P_2	9	7	2

	Maximum Needs (requires)	Current Needs	Allocation (holding)
P_0	10	5	5
P_1	4	2	2
P_2	9	7	2

- At time t_0, the system is in a **safe state**.

The sequence < P_1, P_0, p_2> satisfies the safety condition.

- Process P_1 can immediately be allocated all its tape drives (because the system still have 3 available tape drives) and then return them (the system will then have **(3+2) = 5** available tape drives;

- Then process P_0 can get all its tape drives and return them (the system will then have **(5+5) =10** available tape drives);

- And finally process P_2 can get all its tape drives and return them (the system will then have all **12** tape drives available).

- **A system can go from a safe state to an unsafe state.**
- Suppose that, at time t_1, process P_2 requests and is allocated **one more** tape drive.

	Maximum Needs (requires)	Current Needs	Allocation (holding)
P_0	10	5	5
P_1	4	2	2
P_2	9	6	3

New requests

The system is no longer in a safe state.

At this point, only process P_1, can be allocated all its tape drives. When it returns them, the system will have only 4 available tape drives. Since process P_0, is allocated 5 tape drives, but has a maximum of 10, it may request 5 more tape drives. Since they are unavailable, process P_0 must wait. Similarly, process P_2 may request an additional 6 tape drives and have to wait, resulting in a deadlock.

عند هذه النقطة ، فقط الـ process P1 يمكن ان يخصص لها ما تحتاجه من المورد tape drives وعندما يعيدها ، سيكون لدى النظام 4 من tape drives متاحة فقط. الـ process P0 كان مخصص له 5 من tape drives ولكنه يحتاج كحد أقصى 10 من tape drives فانه سيطلب 5 من tape drives و نظرًا لعدم توفرها ، يجب أن تنتظر وبالمثل ، قد تطلب الـ process P2 , 6 من tape drives وايضاً هي غير متوفرة فيجب عليه الانتظار، مما يؤدي إلى حدوث الـ Deadlock .

AVOIDANCE ALGORITHMS

- خوارزميات تجنب حدوث الـ DEADLOCK تتبع التالي:

1- اذا كان النظام يحتوي على مثيل (حالة) واحد من نوع المورد فيتم استخدام خوارزمية

RESOURCE-ALLOCATION GRAPH

2- اذا كان النظام يحتوي على مثيلات (حالت) متعددة لنوع المورد فيتم استخدام خوارزمية

USE THE BANKER'S ALGORITHM

- SINGLE INSTANCE OF A RESOURCE TYPE
  - USE A RESOURCE-ALLOCATION GRAPH
- MULTIPLE INSTANCES OF A RESOURCE TYPE
  - USE THE BANKER'S ALGORITHM

6.5.2. Resource-Allocation Graph Algorithm

If we have a resource-allocation system with only one instance of each resource type, we can use a variant of the resource-allocation graph for deadlock avoidance. In addition to the request and assignment edges already described, we introduce a new type of edge, called a **claim edge**

- إذا كان لدينا نظام تخصيص الموارد مع مثيل واحد فقط لكل نوع من أنواع الموارد ، فيمكننا resource-allocation graph لتجنب حالة الـ Deadlock.إضافة إلى الـ assignment edges التي تم شرحها سابقاً فيتم استخدام نوع جديد من الـ edge يسمى **claim edge**.

- **Claim edge** ***Pi*** → ***Rj*** indicates that process ***Pi***, may request resource ***Rj***, at some time in the future. This edge resembles a request edge in direction, but is represented in the graph by a **dashed line**.

Claim edge ***Pi*** → ***Rj*** تشير إلى أن الـ process Pi ، قد تطلب المورد Rj، في وقت ما في المستقبل. تشبه هذه الـ edge الجديدة الـ request edge في الاتجاه ، ولكن يتم تمثيلها في الرسم البياني بخط متقطع.

6.5.2. Resource-Allocation Graph Algorithm Cont.

1- When process ***Pi*** requests resource ***Rj***, the claim edge ***Pi*** → ***Rj*** is converted to a request edge.

1- عندما يطلب الـ Pi مصدر Ri فيتم تحويل الـ claim edge الى request edge

2- Similarly, when a resource ***Rj*** is released by ***Pi***, the assignment edge ***Rj*** → ***Pi*** is reconverted to a claim edge ***Pi*** → ***Rj***.

- وعندما يطلق أو يحرر المصدر Ri من قبل الـ Pi فيتم اعادة تحويل assignment edge الى claim edge

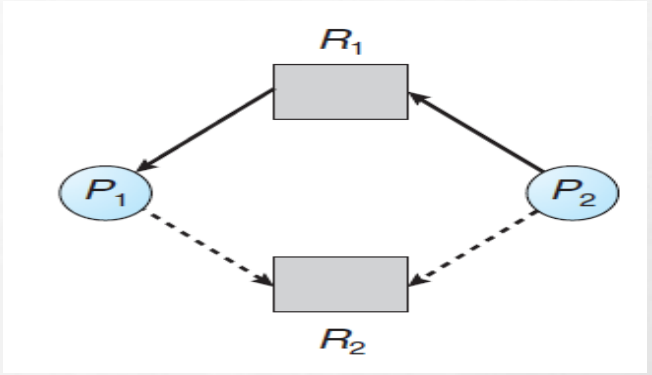
Suppose that process ***Pi*** requests resource ***Rj***. The request can be granted only if converting the request edge ***Pi*** → ***Rj*** to an assignment edge ***Rj*** → ***Pi*** does not result in the formation of a cycle in the resource-allocation graph.

If no cycle exists, then the allocation of the resource will leave the system in a safe state. If a cycle is found, then the allocation will put the system in an unsafe state.

افترض أن الـ process Pi تطلب المورد Ri . يمكن منح الطلب فقط إذا لم ينتج cycle عند تحويل الـ request edge ***Pi*** → ***Rj*** إلى assignment edge ***Rj*** → ***Pi*** في الرسم البياني.
في حالة عدم وجود cycle ، فإن تخصيص المورد سيترك النظام في حالة safe. إذا تم العثور على cycle ، فإن التخصيص سيضع النظام في حالة unsafe state.

EXAMPLE: To illustrate this algorithm, we consider the resource-allocation graph of Figure 6.5.

Figure 6.5 Resource-allocation graph for deadlock avoidance



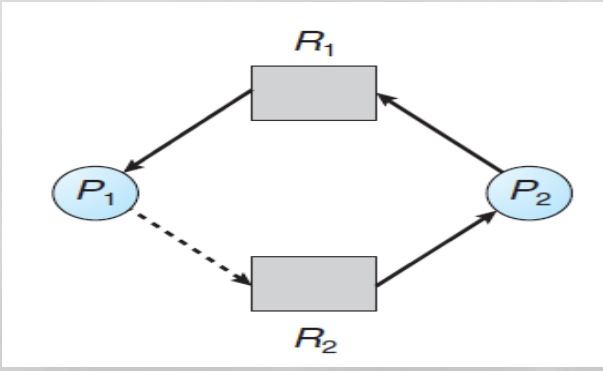
Suppose that P_2 requests R_2. Although R_2 is currently free, we cannot allocate it to P_2, since this action will create a cycle in the graph (Figure 6.6). A cycle indicates that the system is in an unsafe state.

If P_1 requests R_2, and P_2 requests R_1, then a deadlock will occur

افترض أن الـP2 تطلب مورد R2 وعلى الرغم من أن الـR2 متاح حالياً(غير محجوز) لا يمكننا تخصيصه إلى الـ P2، حيث سيؤدي هذا الإجراء إلى إنشاء cycle في الرسم البياني (الشكل 6.6) . تشير ال cycle إلى أن النظام في حالة unsafe.

إذا طلبت الـ P1 مورد R2 وp2 تطلب مورد R1،فسيحدث الـ deadlock

Figure 6.6 An unsafe state in a resource-allocation graph



End of Part 3