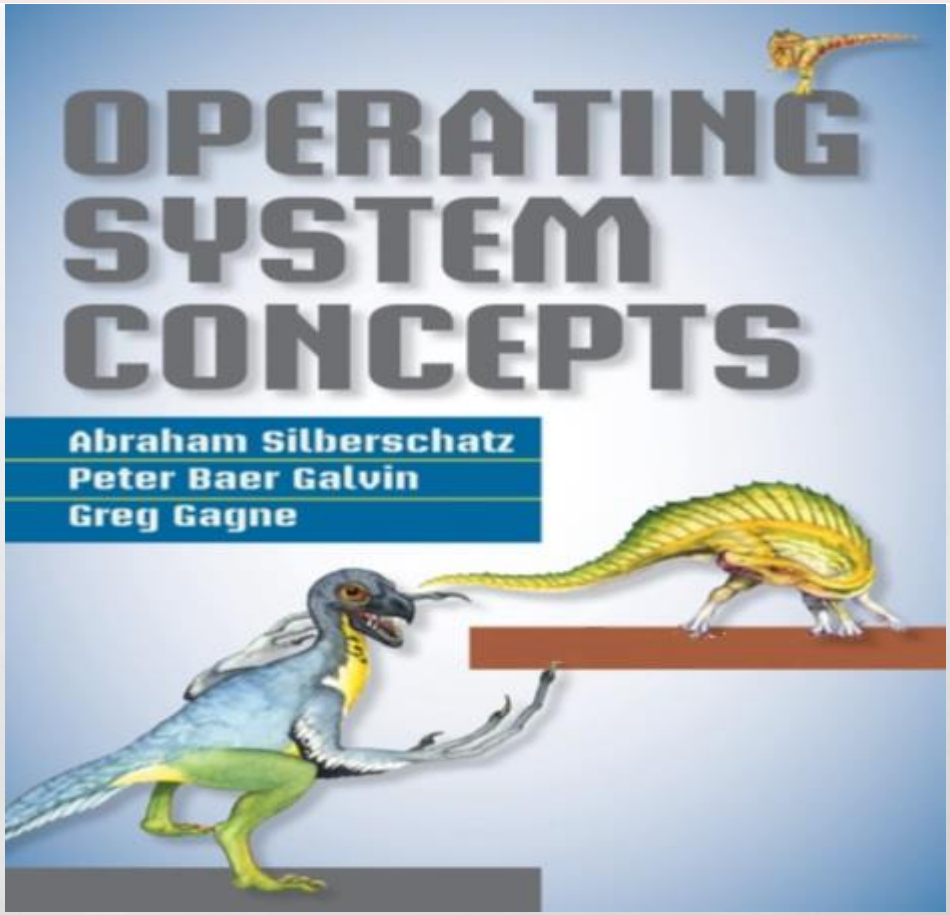
Mustansiriayah University
Collage of Education
Computers Science Department

Chapter Six
Deadlock
Part 5



Fourth Class

Dr. Hesham Adnan ALABBASI

2019-2020

6.6. Deadlock Detection (كشف)

- If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm, then a deadlock situation may occur.

- اذا كان النظام لا يوفر او يوظف خوارزميات منع أو تجنب الـ Deadlock فممكن ان يحدث الـ Deadlock .

- In this environment, the system may provide:

  - Detection algorithm: An algorithm that examines the state of the system to determine whether a deadlock has occurred

  - Recovery scheme: An algorithm to recover from the deadlock.

- في هذه البيئة فيمكن للنظام ان يوفر:

- - خوارزمية الكشف: والتي تقوم بفحص حالة النظام لتحديد فيما اذا تم حدوث الـ Deadlock .

- - طريقة للتخلص أو الاسترداد من الـ Deadlock .

6.6.1. Single Instance of Each Resource Type
الموارد التي تحتوي على حالة واحدة منها

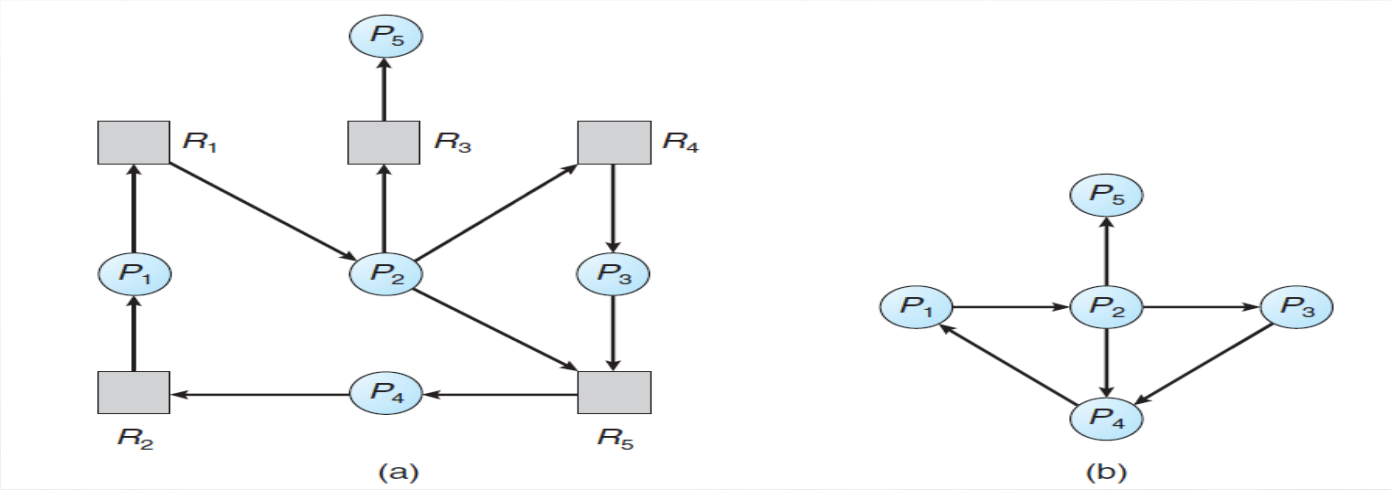
- If all resources have only a single instance, then we can define a deadlock detection algorithm that uses a variant of the resource-allocation graph, called a **wait-for graph**. We obtain this graph from the resource-allocation graph by removing the resource nodes and collapsing the appropriate edges.

إذا كانت جميع الموارد تحتوي على مثيل واحد فقط ، فيمكننا تحديد خوارزمية كشف حالة الـ Deadlock التي تستخدم متغيرًا من resource-allocation graph يسمى **wait-for graph.** نحصل عليه عن طريق إزالة الـ **nodes** وطي الـ **edges** .

- More precisely, an edge from ***Pi*** to ***Pj*** in a wait-for graph indicates that process ***Pi*** is waiting for process ***Pj*** to release a resource that ***Pi*** needs.
- بتعبير أدق ، تشير الـedge من Pi إلى Pj في الـ wait-for graph إلى أن الـ process Pi تنتظر الـ process Pj لإطلاق المورد الذي تحتاجه Pi.
- An edge ***Pi*** → ***Pj*** exists in a wait-for graph if and only if the corresponding resource allocation graph contains two edges ***Pi*** → ***Rq*** and ***Rq*** → ***Pj*** for some resource ***Rq*** .
- توجد Pi → Pj edge في الـ wait-for graph إذا وفقط إذا كان الـ resource allocation graph contains يحتوي على أثنين من الـ edges Pi → Rq و Rq → Pj لبعض الموارد Rq.

Single Instance of Each Resource Type Cont.

In Figure 6.8, we present a resource-allocation graph and the corresponding wait-for graph.



Resource-Allocation Graph Corresponding wait-for graph

- As before, a deadlock exists in the system if and only if the wait-for graph ***contains a cycle***. To detect deadlocks, the system needs to maintain the wait for graph and periodically invoke an algorithm that searches for a cycle in the graph.

- كما في السابق ، يوجد الـ Deadlock النظام إذا وفقط إذا كان الـ wait-for graph يحتوي على ***cycle*** . لاكتشاف الـ Deadlock ، يحتاج النظام إلى الاحتفاظ بالـ wait-for graph واستدعاء خوارزمية بشكل دوري تبحث عن الـ cycle في الرسم .

6.6.2. Several Instances of a Resource Type
الموارد التي تحتوي عدة حالات

In a system with several instances of resources types, A deadlock detection algorithm that is applicable to such a system is used. This algorithm employs several time-varying data structures that are similar to those used in the banker's algorithm.

تستخدم في النظام الذي يحتوي على عدة حالات من الموارد. هذه الخوارزمية وظف هياكل بيانات مشابهه الى خوارزمية الـ banker

- **Available***:* A vector of length ***m*** indicates the number of available resources of each type

متجه بالطول m يشير الى عدد الموارد المتوفرة من كل نوع

- **Allocation***:* An ***n*** **x** ***m*** matrix defines the number of resources of each type currently allocated to each process

مصفوفة ***n*** **x** ***m*** **تشير الى عدد الموارد المحجوزة من كل نوع لكل process**

- **Request***:* An ***n*** **x** ***m*** matrix indicates the current request of each process. If ***Request*** **[*i*][*j*] = *k***, then process P_i is requesting ***k*** more instances of resource type R_j.

مصفوفة ***n*** **x** ***m*** **تشير الى طلب كل process . اذا كان الطلب** ***Request*** **[*i*][*j*] = *k*,** هذا يعني ان الـ process Pi يطلب k من حالات المورد Rj .

Detection Algorithm

The detection algorithm described here simply investigates every possible allocation sequence for the processes that remain to be completed.

تبحث خوارزمية الكشف الموضحة هنا ببساطة في كل تسلسل تخصيص ممكن للprocesses التي لم تكتمل بعد.

1. Let ***Work*** and ***Finish*** be vectors of length *m* and *n,* respectively. Initialize

لنفرض لدينا المتجه Work بالطول m , والمتجه Finish بالطول n . قيمهم الاولية كم ادناه:

Work = available *and*

Finish[i] = false for i=0, 1, ..., *n – 1.*

If Allocation ≠ 0, then and ***Finish[i]= false***, otherwise ***Finish[i]= true.***

2. Find an index i such that both

   a. ***Finish[i] ==false***

   b. ***Need i ≤ Work***

   If no such i exists, go to step 4.

3. ***Work = Work + Allocation i***

Finish[i] = true

Go to step 2.

4. If ***Finish[i]*** = ***true*** some i, 0 ***≤ i < n*** then the system is in a deadlocked state. Moreover, if ***Finish[i] == false***, then process ***Pi*** is deadlocked

Example of Deadlock detection

A system with:
-5 processes P_0 through P_4;
-3 resource types (A, B, C).
-*A* (7 instances), *B* (2 instances), and *C* (6 instances)
Suppose that, at time $\mathbf{T_0}$, we have the following resource-allocation state:

Process	*Allocation*	Request	*Available*
	A B C	A B C	A B C
P0	0 1 0	0 0 0	0 0 0
P1	2 0 0	2 0 2	
P2	3 0 2	0 0 0	
P3	2 1 1	1 0 0	
P4	0 0 2	0 0 2	

ملاحظة : هنا في السؤال يعطى التسلسل $<P_0, P_2, P_3, P_1, P_4>$ **مثلاً ويطلب كشف هل ان النظام يكون في** حالة الـ Deadlock أو لا

Example of Deadlock detection Cont.

الحل:

عندما نلاحظ التسلسل $<P_0, P_2, P_3, P_1, P_4>$ فنبدأ ب Process P0 , نلاحظ ان الـ Request له (0 0 0) فيستطيع التنفيذ بالموارد المتوفرة لدية وبعد انتهاء التنفيذ يرجع الموارد التي كانت محجوزة لديه.

New available=Available + Allocation

0 1 0 = 0 0 0 + 0 1 0

نستمر بالتسلسل Process P2 , نلاحظ ان الـ Request له (0 0 0) فيستطيع التنفيذ بالموارد المتوفرة لديه وبعد انتهاء التنفيذ يرجع الموارد التي كانت محجوزة لديه.

New available=Available + Allocation

3 1 2 = 0 1 0 + 3 0 2

نستمر بالتسلسل Process P3 , نلاحظ ان الـ Request له (1 0 0) فيتم اعطاء الموارد التي يحتاجها ويبدا بالتنفيذ وبعد انتهاء التنفيذ يرجع الموارد التي كانت محجوزة لديه.

New available=Available + Allocation

5 2 3 = 3 1 2 + 2 1 1

نستمر بالتسلسل Process P1 , نلاحظ ان الـ Request له (2 0 2) فيتم اعطاء الموارد التي يحتاجها ويبدا بالتنفيذ وبعد انتهاء التنفيذ يرجع الموارد التي كانت محجوزة لديه.

New available=Available + Allocation

7 2 3 = 5 2 3+ 2 0 0

نستمر بالتسلسل Process P4 , نلاحظ ان الـ Request له (0 0 2) فيتم اعطاء الموارد التي يحتاجها ويبدا بالتنفيذ وبعد انتهاء التنفيذ يرجع الموارد التي كانت محجوزة لديه.

New available=Available + Allocation

7 2 5 = 7 2 3+ 0 0 2

Example of Deadlock detection Cont.

We claim that the system is not in a deadlocked state. Indeed, if we execute our algorithm, we will find that the sequence **<P_0, P_2, P_3, P_1, P_4>** results in ***Finish[i]*** = true for all *i*.

النظام هنا ليس في حالة الـ Deadlock .
اذا تم تنفيذ الخوارزمية نجد ان التسلسل **<P_0, P_2, P_3, P_1, P_4>** ينتج عنه ان الـ
Finish[i] = true لكل i

Example Cont.

Suppose now that process P_2 makes one additional request for an instance of type C (0 0 1). The Request matrix is modified as follows:

نفرض الان ان الـ process P_2 عمل طلب اضافي نوع واحد من المورد C بمعنى (0 0 1) فمصفوفة الـ Request سوف تحدث كما ادناه.

التسلسل او الترتيب هو نفس القديم $<P_0, P_2, P_3, P_1, P_4>$

هنا النظام سوف يدخل في حالة اللـ Deadlock, بالرغم من انه سوف ينفذ P1 وتصبح عدد الموارد الجديد(0 1 0) = New available , لكن الـ Request للـ process P2 هو (0 0 1) لايمكن توفير الموارد التي يحتاجها وبذلك يدخل النظام في حالة الـ Deadlock في الـ process P2 .

Process	Request
	A B C
P0	0 0 0
P1	2 0 2
P2	0 0 1
P3	1 0 0
P4	0 0 2

We claim that the system is now deadlocked. Although we can reclaim the resources held by process P_0, the number of available resources is not sufficient to fulfill the requests of the other processes.

Thus, a deadlock exists, consisting of processes P_1, P_2, P_3, and P_4.

6.7. Recovery from Deadlock

- **When a detection algorithm determines that a deadlock exists, several alternatives are available.**

- One possibility is to inform the operator that a deadlock has occurred and to let the operator deal with the deadlock manually.

- Another possibility is to let the system **recover** from the deadlock automatically.

عندما تحدد خوارزمية الكشف وجود حالة Deadlock، تتوفر عدة بدائل.

-أحد الاحتمالات هو إبلاغ المشغل بحدوث حالة Deadlock والسماح للمشغل بالتعامل مع حالة Deadlock يدوياً .

-هناك إمكانية أخرى تتمثل في السماح للنظام بالتعافي من حالة الـ Deadlock تلقائيًا

- There are two options for breaking a deadlock (methods).

1. One is simply to abort one or more processes to break the circular wait.
2. The other is to preempt some resources from one or more of the deadlocked processes.

- هناك خياران لكسر ال Deadlock

1. واحد هو ببساطة تجاوز(abort) process واحدة أو أكثر لكسر الـ circular wait.
2. والآخر هو أستقطاع (استباق) بعض الموارد من واحدة أو أكثر من الـ processes التي حدث فيها الـ Deadlock

6.7.1. Process termination

- To eliminate deadlocks by aborting a process, we use one of two methods. In both methods, the system reclaims all resources allocated to the terminated processes.
  1. Abort all deadlocked processes.
  2. Abort one process at a time until the deadlock cycle is eliminated.

لإزالة الـ Deadlock عن طريق تجاوز او اهمال process ، نستخدم إحدى الطريقتين .في كلتا الطريقتين ، يسترد النظام جميع الموارد المخصصة لل processes التي حدث فيها الـ Deadlock

1. تجاوز او اهمال جميع الـ processes التي حدث فيها الـ Deadlock .
2. تجاوز او اهمال process واحدة في كل مرة حتى يتم التخلص من دورة الـ Deadlock .

- Aborting a process may not be easy. If the process was in the midst of updating a file, terminating it will leave that file in an incorrect state. Similarly, if the process was in the midst of printing data on a printer, the system must reset the printer to a correct state before printing the next job.

- تجاوز او اهمال process قد لا تكون سهلة. إذا كانت الـ process في خضم تحديث ملف ، فإن إنهاءه سيترك هذا الملف في حالة غير صحيحة. وبالمثل ، إذا كانت الـ process في خضم بيانات الطباعة على الطابعة ، فيجب على النظام إعادة تعيين الطابعة إلى الحالة الصحيحة قبل طباعة المهمة التالية.

6.7.1. Process termination Cont.

- If the partial termination method is used, then we must determine which deadlocked process (or processes) should be terminated. This determination is a policy decision, similar to CPU-scheduling decisions.
- إذا تم استخدام طريقة الإنهاء الجزئي ، فيجب علينا تحديد أي process أو processes التي حدث فيها الت Deadlock يجب أن يتم إنهاؤها. هذا التحديد هو قرار سياسة ، مشابه لقرارات جدولة وحدة المعالجة المركزية

- Many factors may affect which process is chosen, including:
- هنالك عدة معايير تؤثر على اختيار الـ process لغرض انهاءه

1. What the priority of the process is ?
2. How long the process has computed and how much longer the process will compute before completing its designated task
3. How many and what types of resources the process has used (for example, whether the resources are simple to preempt)
4. How many more resources the process needs in order to complete
5. How many processes will need to be terminated
6. Whether the process is interactive or batch

End of Chapter Six