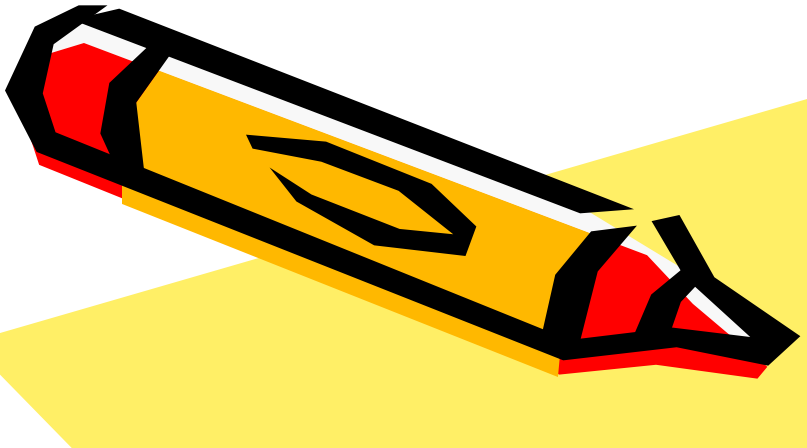
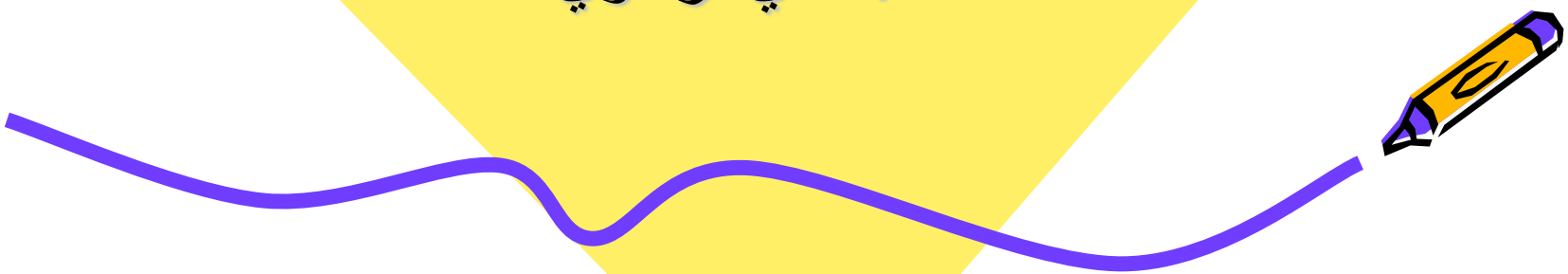


الحلقة المغلقة Deadlock
(أو الإستعصاء)

د. فادي تركاوي



مراجعة للمحاضرة السابقة

الخلاصة:

بالطبع و على نفس المبدأ بالإضافة لأن السيمافور يؤمن الإستبعاد المتبادل و التزامن للمصادرفي الحاسب فهو أيضا له نفس المبدأ للتحكم بالآلات المبرمجة لـــ :

- لآلات الصودا (المشروب الغازي)
- ألات النقود ATM
- ألات غسل الصحون Dish washer
- .....

Coke machine

```
/* number of full slots (Cokes) in machine */
semaphore fullSlot = 0;
/* number of empty slots in machine */
semaphore emptySlot = 100;
/* only one person accesses machine at a time */
semaphore mutex = 1;

DeliveryPerson()
{
    emptySlot.P( );        /* empty slot avail? */
    mutex.P( );            /* exclusive access */
    put 1 Coke in machine
    mutex.V( );
    fullSlot.V( );         /* another full slot! */
}

ThirstyPerson()
{
    fullSlot.P( );         /* full slot (Coke)? */
    mutex.P( );            /* exclusive access */
    get 1 Coke from machine
    mutex.V( );
    emptySlot.V( );        /* another empty slot! */
}
```

Deadlock

تُعتبر فلسفة بناء نظم التشغيل من أجمل العلوم وأعقدها بنفس الوقت.. وعموما أنظمة التشغيل يواجهها مشاكل كثيرة جداً وقد لايستطيع أحد إحصاءها. يحاول كاتبوا نظام التشغيل تزويد النظام بحلول مسبقة للمشاكل المتوقع حدوثها. لكن بسبب كثرة وعشوائية العوامل التي يتعرض لها نظام التشغيل.يصعب التعرف بنوعية هذه المشاكل.

تعريف

هو منع العمليات من استخدام موارد النظام بشكل دائم بسبب تنافسها مع العمليات الأخرى على هذه الموارد أثناء عملية الاتصال بينهم.

P_0	P_1
wait(A);	wait(B);
wait(B);	wait(A);

Deadlock الإستعصاء

- أفترض وجود Context Switch بعد طلب ال Printer من العملية A
- مع فرض وجود طابعة و قرص واحد فقط

Process A	Process B
printer–>wait( );	disk–>wait( );
disk->wait( );	printer->wait( );
print file	*print file*
printer->signal( );	disk->signal( );
disk->signal( );	printer->signal( );

تعريف

- Deadlock is defined as the permanent blocking of a set of processes that compete for system resources.
  - الإستعصاء هو حجز دائم لمجموعة من العمليات التي تتنافس لأجل مصادر النظام

هناك نوعان من الموارد:

1- يمكن احتكارها (Preemptable) **مثل** الذاكرة .

2- لا يمكن احتكارها (Non-Preemtable) **مثل** الطابعة.

الموارد التي يمكن احتكارها:

هي الموارد التي تعين لعملية ما إلى أن تنتهي من تنفيذ مهمتها ثم تعين إلى مهمة أخرى بدون أي تصادم.

الموارد التي لا يمكن احتكارها:

هي الموارد التي لا يمكن أن تؤخذ من عملية وتعين إلى أخرى بدون أن تسبب تصادم.

"عند ذكرنا لمثال الطابعة لا نقصد أن تعين إلى عمليه أخرى خلال تنفيذ مهامها, بل نظام التشغيل يعامل الطابعة كمورد يمكن احتلاله –المشغل يرتب المهام التي يجب على الطابعة تنفيذها

يحدث في الموارد التي لا يمكن احتلالها كما سنرى أن الجمود

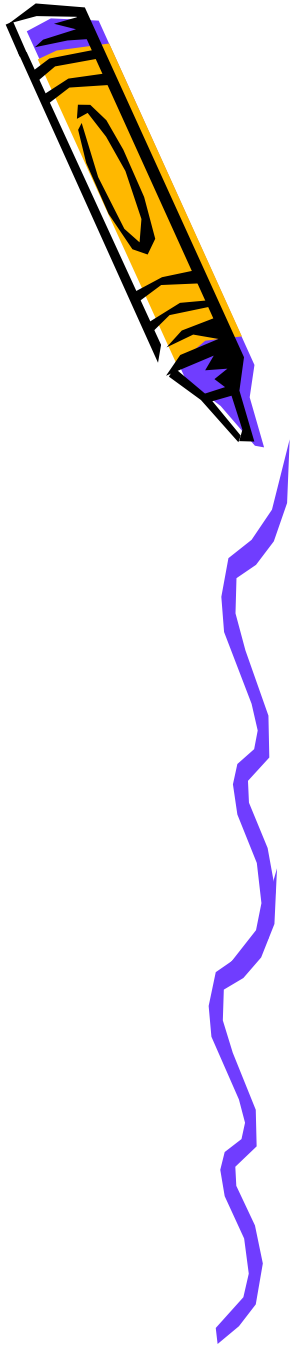
شروط حدوث الإستعصاء
Deadlock Condition

- يحدث الإستعصاء من بسبب ثلاثة أشياء :
- Mutual Exclusionيجب أن تنتظر هنا العملية التي تنتظر مصدر ما حتى تنتهي الأخرى (التي تملك المصدر). و هذا لا يمكن الإستغناء عنه في المصادر الغير القابلة للمشاركة كالطابعة مثلا أم في المصارد القابلة للمشاركة كالذاكرة فيمكن أن يستغنى عن الإقصاء المتبادل.
- Non Preemption هنا لا يتم إقصاء المصدر بشكل قسري و لكن بشكل إختياري و هذا يعني أنه لا يمكن لنظام التشغيل أو لعملية ان تبعد عملية أخرى عن المصدر الحالي و لتجنب ذلك نسمح بعملية ال Preemption و هذا يكون في المصادر التي يمكن أن تخزن حالتها كالذواكر
- Hold and Waitوهذا يعني أن العملية تستطيع أن تمسك بمصدر و تبقى في حالة الإنتظار للحصول على مصدر آخر ممسوك من قبل معالجات آخرى

معالجة الإستعصاء

Dealing with Deadlock

- Deadlock Preventioin منع الإستعصاء من الحدوث و ذلك من خلال منع ثلاث أشياء سيتم ذكرها لاحقا
- Deadlock Detectionكشف الإستعصاء و معالجته
- Deadlock Avoidance و يتم هنا تحقيق الطلبات من المصادر للعمليات بحيث يتم تجنب الإستعصاء

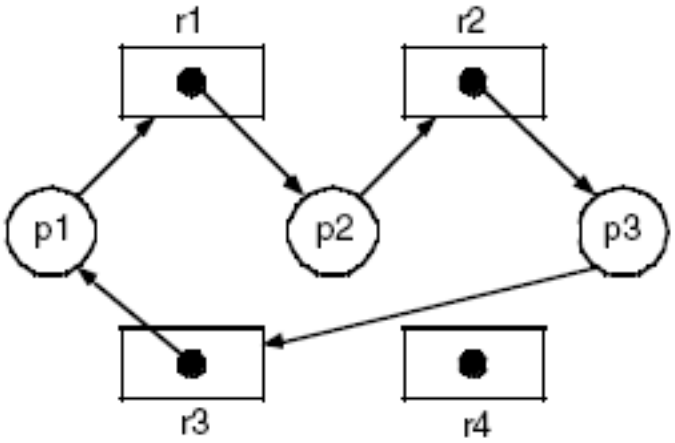


Deadlock Preventioin

منع الإستعصاء

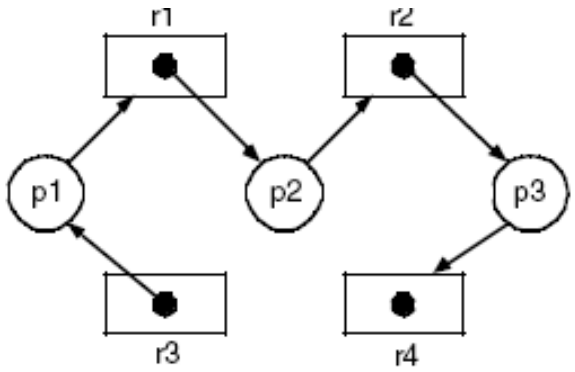
تخصيص المصادر
Resource Allocation

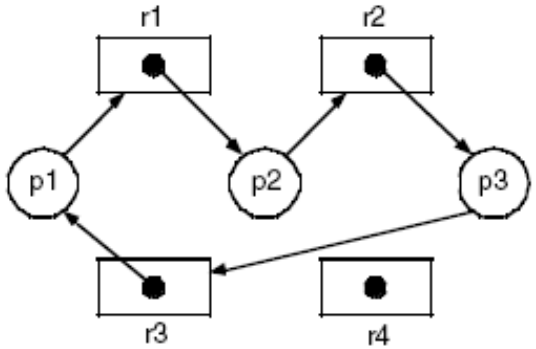
- عملية الإستقصاء يتم إستخلاص موديلها من خلال الغرافات الموجهة Directed Graph و هنا يتم تمثيل
  - المصادر كمربعات و تكرار المصدر يمثل بنقط ضمن ذلك المربع
  - الدوائر تمثل العمليات
  - الطلب يمثل بسهم من العملية بإتجاه المصدر
  - وفي حال تم تخصيص المصدر للعملية فإن ذلك يمثل بسهم من المصدر للعملية



تمثيل المصادر من خلال غراف وحيد المصدر

- في حال عدم إحتواء الغراف على حلقة فهذا يعني عدم وجود إستعصاء و بهذا يتم منع الإستعصاء
- في حال وجود حلقة فإن ذلك شرط ضروري و كافي لحصول الإستعصاء





Deadlock Detection

Deadlock Detection

Existing Resources
(4 2 3 1)

Available Resources
(2 1 0 0)

Current Allocation

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

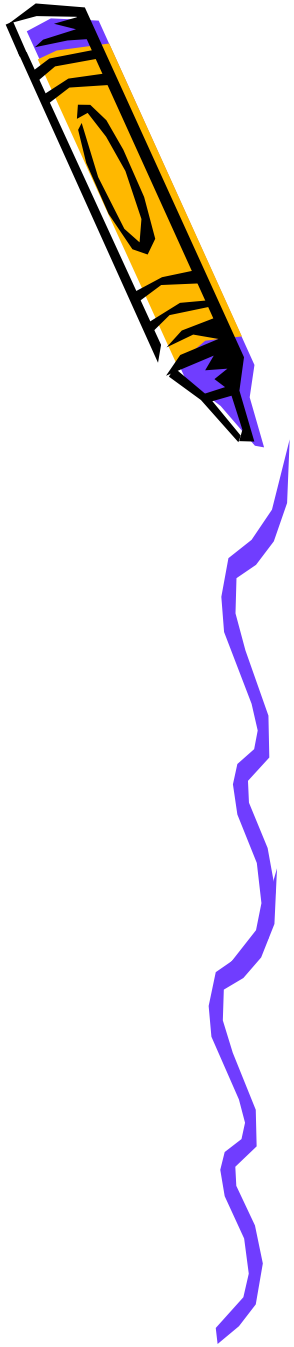
Request

$$\begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

resources = (tape drive plotter printer CDROM)

- Whose request can be fulfilled?
  - Process 1 — no — no CDROM available
  - Process 2 — no — no printer available
  - Process 3 — yes — give it the requested resources, and after it completes and releases those resources, A = (2 2 2 0)
  - Process 1 still can't run (no CDROM), but process 2 can run, giving A = (4 2 2 1)
  - Process 1 can run, giving A = (4 2 3 1)

Deadlock Detection خوارزمية



- Operation:
  - Every process is initially unmarked
  - As algorithm progresses, processes will be marked, which indicates they are able to complete, and thus are not deadlocked
  - When algorithm terminates, any unmarked processes are deadlocked
- Algorithm:
  1. Look for an unmarked process Pi for which the i-th row of the **Request** matrix is less than or equal to the **Available** vector
  2. If such a process is found, add the i-th row of the **Current** matrix to the **Available** vector, mark the process, and go back to step 1
  3. If no such process exists, the algorithm terminates

Deadlock Avoidance

Deadlock Avoidance Safe and Unsafe State

تجنب الإستعصاء الحالة الآمنة و الغير آمنة

	Has	Max
A	3	9
B	2	4
C	2	7

free: 3
(a)

	Has	Max
A	3	9
B	4	4
C	2	7

free: 1
(b)

	Has	Max
A	3	9
B	0	—
C	2	7

free: 5
(c)

	Has	Max
A	3	9
B	0	—
C	7	7

free: 0
(d)

	Has	Max
A	3	9
B	0	—
C	0	—

free: 7
(e)

	Has	Max
A	9	9
B	0	—
C	0	—

free: 1
(f)

	Has	Max
A	0	—
B	0	—
C	0	—

free: 10
(g)

- الحالة تكون آمنة في حال توفر تتابع من الحصول على المصادر الذي يحقق إنهاء عملية واحدة على الأقل

B runs, asks for 2 more resources, 1 free -
B finishes, releases its resources, 5 free -
C runs, asks for 5 more resources, 0 free -
C finishes, releases its resources, 7 free -
A runs, gets 6 more, everyone done... -

Deadlock Avoidance

	Has	Max
A	3	9
B	2	4
C	2	7

free: 3
(a)

	Has	Max
A	4	9
B	2	4
C	2	7

free: 2
(b)

	Has	Max
A	4	9
B	4	4
C	2	7

free: 0
(c)

	Has	Max
A	4	9
B	0	—
C	2	7

free: 4
(d)

- Suppose we start in state (a), and reach state (b) by giving A another resource
  - B runs, asks for 2 more resources, 0 free
    - B finishes, releases its resources, 4 free
  - C can't run — might want 5 resources
    - Same for A
- State (b) is unsafe, meaning that from there, deadlock **may** eventually occur
  - State (b) is not a deadlocked state — the system can still run for a bit
  - Deadlock may not occur — A might release one of its resources before asking for more, which allows C to complete

خوارزمية البانكر للمصدر الآحادي

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

free: 10
(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

free: 2
(b)

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

free: 1
(c)

- موظف البنك (البانكر) يعطي قروض للزبائن A, B, C, D
- موظف البنك يعلم بشكل مسبق بأنه ليس من المحتمل بأن كل الزبائن يطلبون كامل القرض في نفس الوقت
- في نقطة ما في الحالة b البانكر يجب أن يعطي Cو بعدها B و بعدها D
- في حال أعطى البانكر قرض إضافي ل B ففي هذه الحالة سوف ندخل الإستعصاء

خوارزمية بانكر للمصدر الأحادي

- خوارزمية طلب المصدر:
- لتقرير فيما إذا أن الحالة التالية آمنة، يقوم البانكر بفحص فيماإذا كان هناك مصادر كافية لتأمين بعض الزبائن (هذا يعني بأن الزبون سيقوم بآخذ القرض بشكل كامل و بعد ذلك يقوم بإعادته). و هنا نفرض بأن المصادر أو القروض سيتم إستردادها و بعدها يتم النظر فيماإذا كانت المصادر كافية لتأمين زبون آخر