جامعة المنارة

كلية: .....الهندسة.....

قسم: ..... الهندسة المعلوماتية.....

اسم المقرر: ..... نظم تشغيل 2.....

رقم الجلسة (...7...)

عنوان الجلسة

الجمـــــــــود

م.عمار مصطفى

الفصل الدراسي العام الدراسي 2025/ 2024

جدول المحتويات

Contents

العنوان	رقم الصفحة
التمثيل البياني لحالة الجمود	
خوارزمية المصرفي	

جامعة المنارة

الغاية من الجلسة : التعريف بحالة الجمود و طرق كشف حالة الجمود

1. ما هو الانسداد (Deadlock) في نظم التشغيل؟

الإجابة :الانسداد هو حالة يصبح فيها مجموعة من العمليات في نظام التشغيل غيرقادرة على التقدم لأن كل عملية تحتفظ بمورد وتنتظر موردًا آخر يحتفظ به عملية أخرى، مما يؤدي إلى توقف دائم.

2. ما هي الشروط الأربعة اللازمة لحدوث الانسداد؟

الإجابة :

- الاحتفاظ الحصري :(Mutual Exclusion) يجب أن تكون الموارد محتفظ بها بشكل حصري.
- الاحتفاظ والانتظار :(Hold and Wait) العملية تحتفظ بمورد واحد على الأقل وتنتظر موردًا آخر.
- عدم الاستباقية :(No Preemption) لا يمكن سحب الموارد من العملية بالقوة.
- الانتظار الدوري :(Circular Wait) تشكل العمليات سلسلة دورية حيث تنتظر كل عملية موردًا يحتفظ به العملية التالية.

التمرين 1: كشف الجمود في رسم بسيط

المعطى:

في نظام يحتوي على المسالك التالية:

- T1 ⟶ R1
- R1 ⟶ T2
- T2 ⟶ R2
- R2 ⟶ T1

المطلوب:

1. مثل الرسم البياني لتخصيص الموارد.
2. هل توجد حلقة؟
3. هل هذه الحلقة تؤدي إلى جمود؟ ولماذا؟

الحل:

1. رسم حلقي T1 ⟶ R1 ⟶ T2 ⟶ R2 ⟶ T1 :
2. نعم، توجد حلقة.
3. نعم، لأن كل مورد له **مثيل واحد فقط**، وجميع المسالك تنتظر بعضها البعض.

التمرين 2: حلقة بدون جمود

المعطى:

- R1 يحتوي على مثيلين.
- T1 ⟶ R1 (طلب)
- T2 ⟶ R1 (طلب)
- R1 ⟶ T1(تخصيص مثيل)
- R1 ⟶ T2(تخصيص مثيل)

المطلوب:

1. هل يوجد جمود رغم وجود حافة دائرية بين T1 و T2عبر R1 ؟

الحل:

- لا يوجد جمود، لأن **كل من T1 و T2حصل على مثيل من R1** ، ولا أحد ينتظر الآخر.

التمرين 3: بناء حلقة مركبة

المعطى:

- T1 ⟶ R1
- R1 ⟶ T2
- T2 ⟶ R2
- R2 ⟶ T3
- T3 ⟶ R3
- R3 ⟶ T1

المطلوب:

1. هل توجد حلقة؟

2. إذا كانت كل الموارد تحتوي على مثيل واحد فقط، فهل النظام في حالة جمود؟
3. ماذا يحدث إذا كان R3 يحتوي على 2 مثيل؟

الحل المتوقع:

1. نعم، توجد حلقةT1 ⟶ R3 ⟶ T3 ⟶ R2 ⟶ T2 ⟶ R1 ⟶ T1 :
2. نعم، جمود تام.

إذا كان R3 يحتوي على مثيلين، يمكن أن يحصل T3 على واحد، وإذا لم يكن ينتظر الآخر، **لا يوجد جمود**

التمرين 5: حلقة بدون جمود

المعطى:

- T1يحتفظ بمورد R1 (1 مثيل)، ويطلبR2
- T2يحتفظ بـ R2 (2 مثيل) ويطلبR1
- R2 يحتوي على 2 مثيل — 1 مخصص و1 متاح

المطلوب:

- هل الطلبات المتبادلة بين T1 و T2تؤدي إلى جمود؟
- ماذا يحدث إذا تم منح المثيل الثاني من R2 إلىT3 ؟

الحل:

- لا جمود طالما بقي مثيل متاح من.R2
- إذا تم منحه لT3 ، فإن T2 سيتوقف وينتظرR2 ، و T1ينتظر R2 أيضًا ⟶ **محتمل جمود**.

خوارزمية المصرفي :

تُستخدم لتجنب الانسداد من خلال التحقق مما إذا كان تخصيص الموارد سيؤدي إلى حالة آمنة، أي أن هناك تسلسلًا يمكن من خلاله تنفيذ جميع العمليات دون حدوث انسداد (جمود) . الخطوات الأساسية هي:

1. **قارن الاحتياجات (Need) لكل عملية بالموارد المتاحة.(Available)**
2. **إذا كانت الاحتياجات أقل من أو تساوي الموارد المتاحة، خصص الموارد مؤقتًا، وأعد الموارد المخصصة (Allocation) إلى الموارد المتاحة بعد "إنهاء" العملية.**

كرر العملية حتى يتم إنهاء جميع العمليات أو عدم إمكانية إيجاد عملية قابلة للتنفيذ

المسألة

لدينا نظام تشغيل يحتوي على 3 أنواع من الموارد A (10 وحدات)، B (5 وحدات)، C (7 وحدات). يوجد 4 عمليات (P0، P1، P2، P3) مع تخصيص الموارد واحتياجاتها كما في الجدول التالي:

العملية	التخصيص (ABC)	الاحتياج (ABC)	القصوى (ABC)	الموارد المتوفرة
P0	010	743	753	332
P1	200	122	322	
P2	302	030	332	
P3	200	211	422	

الأسئلة:

1. **هل النظام في حالة آمنة؟ إذا كان كذلك، ما هو التسلسل الآمن؟**
2. **إذا طلبت العملية P1 موردًا إضافيًا (1, 0, 2)، هل يمكن قبول الطلب بأمان؟**

.هل النظام في حالة آمنة؟ إذا كان كذلك، ما هو التسلسل الآمن؟

الخطوة 1: إعداد البيانات

- **الموارد الكلية.(10, 5, 7) :**
- **الموارد المتاحة.(3, 3, 2) :**
- **الاحتياج = (Need) القصوى - (Max) التخصيص(Allocation) ، وهو معطى في الجدول.**
- **ننشئ متغيرًا لتتبع العمليات المُنفذة (Finish) ونعيّنه False لكل عملية في البداية.**
- **الموارد المتاحة مؤقتًا.(3, 3, 2) = (Work)**

الخطوة 2: تطبيق خوارزمية المصرفي نبحث عن عملية يكون احتياجها (Need) أقل من أو يساوي الموارد المتاحة(Work) ، ثم ننفذها ونحدث الموارد المتاحة.

- التكرار الأول :
  - P0: Need = (7, 4, 3) > (3, 3, 2)، لا يمكن تنفيذ P0.
  - P1: Need = (1, 2, 2) ≤ (3, 3, 2)، يمكن تنفيذ P1.
    - نفذ P1: Finish[P1] = True.
    - Work = (3, 3, 2) + Allocation[P1] = (3, 3, 2) + (2, 0, 0) = (5, 3, 2).
- التكرار الثاني :
  - الموارد المتاحة الآن (5, 3, 2) :
  - P0: Need = (7, 4, 3) > (5, 3, 2)، لا يمكن تنفيذ P0.
  - P2: Need = (0, 3, 0) ≤ (5, 3, 2)، يمكن تنفيذ P2.
    - نفذ P2: Finish[P2] = True.
    - Work = (5, 3, 2) + Allocation[P2] = (5, 3, 2) + (3, 0, 2) = (8, 3, 4).
- التكرار الثالث :
  - الموارد المتاحة الآن (8, 3, 4) :
  - P0: Need = (7, 4, 3) > (8, 3, 4)لأن 4 (B) 3 <، لا يمكن تنفيذ P0.
  - P3: Need = (2, 1, 1) ≤ (8, 3, 4)، يمكن تنفيذ P3.
    - نفذ P3: Finish[P3] = True.
    - Work = (8, 3, 4) + Allocation[P3] = (8, 3, 4) + (2, 1, 1) = (10, 4, 5).
- التكرار الرابع :
  - الموارد المتاحة الآن (10, 4, 5) :
  - P0: Need = (7, 4, 3) ≤ (10, 4, 5)، يمكن تنفيذ P0.
    - نفذ P0: Finish[P0] = True.

- Work = (10, 4, 5) + Allocation[P0] = (10, 4, 5) + (0, 1, 0) = (10, 5, 5).

- جميع العمليات (P0, P1, P2, P3) تم تنفيذها Finish = True(للجميع.)

النتيجة:

- النظام في حالة آمنة.
- التسلسل الآمن.P0 ⟶ P3 ⟶ P2 ⟶ P1 :

ي: إذا طلبت العملية P1 موردًا إضافيًا (1, 0, 2)، هل يمكن قبول الطلب بأمان؟

الخطوة 1: التحقق من إمكانية التخصيص

- الطلب.(2, 0, 1) :
- الموارد المتاحة.(3, 3, 2) :
- المقارنة: (1, 0, 2) ≤ (3, 3, 2)، الطلب ممكن لأن 1 ≤ 3، 0 ≤ 3، 2 ≤ 2.
- التحقق من أن الطلب لا يتجاوز الاحتياج: الطلب(2, 2, 1) = [P1]Need ≤ (2, 0, 1) ، وهو صحيح.

الخطوة 2: تخصيص الطلب مؤقتًا

- تحديث تخصيص.(3, 0, 2) = (1, 0, 2) + (2, 0, 0) = P1: Allocation[P1]
- تحديث احتياج.(0, 2, 0) = (1, 0, 2) - (1, 2, 2) = P1: Need[P1]
- تحديث الموارد المتاحة.(2, 3, 0) = (1, 0, 2) - (3, 3, 2) = Available :

الجدول الجديد:

العملية	التخصيص (ABC)	الاحتياج (ABC)	الموارد المتوفرة
P0	010	743	230
P1	302	020	
P2	302	030	
P3	200	211	

الخطوة 3: التحقق من الحالة الآمنة

- Work = (2, 3, 0).
- Finish = [False, False, False, False].

- التكرار الأول :
  - P0: Need = (7, 4, 3) > (2, 3, 0)، لا يمكن تنفيذ P0.
  - P1: Need = (0, 2, 0) ≤ (2, 3, 0)، يمكن تنفيذ P1.
    - نفذ P1: Finish[P1] = True.
    - Work = (2, 3, 0) + Allocation[P1] = (2, 3, 0) + (3, 0, 2) = (5, 3, 2).
- التكرار الثاني :
  - الموارد المتاحة (5, 3, 2) :
  - P0: Need = (7, 4, 3) > (5, 3, 2)، لا يمكن تنفيذ P0.
  - P2: Need = (0, 3, 0) ≤ (5, 3, 2)، يمكن تنفيذ P2.
    - نفذ P2: Finish[P2] = True.
    - Work = (5, 3, 2) + Allocation[P2] = (5, 3, 2) + (3, 0, 2) = (8, 3, 4).
- التكرار الثالث :
  - الموارد المتاحة (8, 3, 4) :
  - P0: Need = (7, 4, 3) > (8, 3, 4) لأن 4 (B) 3 <، لا يمكن تنفيذ P0.
  - P3: Need = (2, 1, 1) ≤ (8, 3, 4)، يمكن تنفيذ P3.
    - نفذ P3: Finish[P3] = True.
    - Work = (8, 3, 4) + Allocation[P3] = (8, 3, 4) + (2, 1, 1) = (10, 4, 5).
- التكرار الرابع :
  - الموارد المتاحة (10, 4, 5) :
  - P0: Need = (7, 4, 3) ≤ (10, 4, 5)، يمكن تنفيذ P0.
    - نفذ P0: Finish[P0] = True.
    - Work = (10, 4, 5) + Allocation[P0] = (10, 4, 5) + (0, 1, 0) = (10, 5, 5).
- جميع العمليات تم تنفيذها.

النتيجة:

- **النظام في حالة آمنة بعد تخصيص الطلب.**
- **التسلسل الآمن.P0 ⟶ P3 ⟶ P2 ⟶ P1 :**
- **الإجابة :يمكن قبول طلب P1 للموارد (1, 0, 2) بأمان.**

جامعة المنارة