Université Eshahid Hamma Lakhdar El-Oued
Faculté des sciences exactes
Département d'Informatique

Module : Système Exploitation II
Niveau : 3ème Année Informatique
Le 22/01/2026 – Durée : 1h30

Contrôle Semestriel

Exercice 1

On a trois processus : P1, P2, P3 et trois types de ressources **:**

- R1 : 1 exemplaire
- R2 : 2 exemplaires
- R3 : 1 exemplaire

À un instant *t*, l'état du système est le suivant :

- P1 **détient** (possède) R1 et **demande** R2
- P2 **détient** R2 et **demande** R3
- P3 **détient** R3 et **demande** R1

Questions :

1. Le graphe contient-il un cycle ? Le préciser.
2. Peut-on conclure qu'il y a un interblocage ?
3. Donner une séquence d'exécution possible qui évite l'interblocage (si possible).
4. Donner une modification (politique ou ordre) qui empêche ce type d'interblocage.

Exercice 2

On a trois processus P1, P2, P3.
Chaque processus exécute des actions dans cet ordre local :

- P1 : A1 ; A2 ; A3
- P2 : B1 ; B2 ; B3
- P3 : C1 ; C2 ; C3

On impose les **équations (contraintes de précédence)** suivantes :

1. A1 < B2 (B2 ne peut commencer qu'après A1)
2. B1 < C2
3. C1 < A2
4. A2 < B3
5. B2 < C3
6. C2 < A3

Travail demandé :

Modéliser ces contraintes à l'aide de sémaphores et écrire le pseudo-code complet des trois processus.

Exercice 3

Un pont ne permet de faire passer **qu'une voiture à la fois** et fonctionne ainsi :

- Les voitures arrivent des deux côtés : **Côté A** et **Côté B**.
- On autorise un **flux alterné** :
  - 3 voitures maximum du côté A, puis 3 du côté B, etc.
  - Une seule voiture peut être sur le pont à un instant donné.

1. Modéliser chaque voiture comme un processus.
2. Proposer des sémaphores pour :
   - garantir l'exclusion sur le pont
   - garantir l'alternance A/B

يوجد جسر ضيّق يربط بين جهتين متقابلتين تُسمّى:

- الجهة A
- الجهة B

نظراً لضيق الجسر، فإنه لا يسمح بمرور إلا سيارة واحدة فقط في أي لحظة زمنية.

تصل السيارات إلى الجسر بشكل عشوائي من الجهتين A وB ، ويجب تنظيم حركة المرور وفق القواعد التالية:

1. لا يجوز وجود أكثر من سيارة واحدة على الجسر في نفس الوقت (إقصاء متبادل).
2. يتم المرور بطريقة تناوبية منظّمة على شكل دفعات:
   - تمر ثلاث سيارات كحد أقصى متتالية من الجهةA ،
   - ثم تمر ثلاث سيارات كحد أقصى متتالية من الجهةB ،
   - ثم يعود الدور إلى الجهةA ، وهكذا بشكل دوري.
3. إذا لم تكن هناك سيارات منتظرة في إحدى الجهتين، يمكن للجهة الأخرى الاستمرار في المرور دون انتظار الدور.

<u>المطلوب :</u>

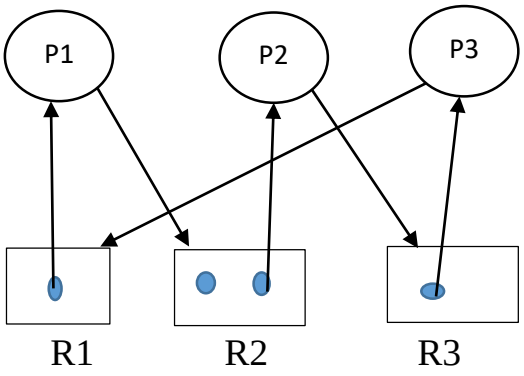
1. نمذجة أو تمثيل كل سيارة على أنها عملية معالجة (Processus) مستقلة.
2. اقتراح مجموعة من السيمفورات (Semaphores) للقيام بالمزامنة اللازمة من أجل:

- ضمان أن تمر سيارة واحدة فقط على الجسر في كل مرة.
- ضمان التناوب بين الجهتين A و B بمجموعات من ثلاث سيارات كحد أقصى.

Corrigé-Type

Exercice 1 (07.00 Pts)

Dessin du graphe (Graphe d'allocation de ressources) (02.00 pts)



1) Cycle ? (01.00 pt)

Oui. Le cycle est :

P1 -> R2 -> P2 -> R3 -> P3 -> R1 -> P1`

2) Cycle = Interblocage ? (01.00 pt)

Pas toujours.

* Si toutes les ressources du cycle ont une seule exemplaire, cycle ⇒ Interblocage certain

* Ici, R2 a 2 exemplaires : une occupée et autre libre.

☞ Donc : cycle présent mais interblocage pas certain (pas obligatoire), car une autre instance de `R2` peut casser l'attente.

3) Séquence qui évite l'interblocage (2.00 pts)

Comme une de `R2` est libre, on peut allouer l'autre à `P1` :

1. Allouer `instance de R2 libre à `P1` → `P1` possède `R1 + une de R2`

2. `P1` termine sa section critique et libère `R1` et une de `R2`

3. Maintenant `P3` peut obtenir `R1` (il le demandait), donc il progresse et libère `R3`

4. `P2` obtient `R3`, progresse, puis libère l'autre instance de de `R2`

➞Tous les processus sont exécutés et toutes les ressources sont libérées, pas d'interblocage.

➔La séquence est : P1->P3->P2

4) Une modification pour empêcher ce scénario (prévention) (01.00 pt)

Deux options classiques (tu peux donner l'une des deux) :

Option A — Ordre global sur les ressources (prévention)

Imposer un ordre unique : `R1 < R2 < R3`

Tout processus doit demander les ressources dans cet ordre.

Exemple : `P3` ne doit pas demander `R1` après `R3` si ça viole l'ordre.

Option B — "Tout ou rien" (éviter la condition de détention et d'attente)

Un processus ne garde pas une ressource s'il n'obtient pas l'autre :

* si `P2` a `R2` mais n'a pas `R3`, il libère `R2` et réessaie plus tard.

Exercice 2 (05.50 pts)

Idée

Chaque contrainte X < Y se traduit par un sémaphore SXY initialisé à 0 :

- Celui qui exécute **X** fait V(SXY) à la fin de X.
- Celui qui exécute **Y** fait P(SXY) juste avant Y.

1) Déclaration (01.00 pt)

S_A1_B2, S_B1_C2, S_C1_A2, S_A2_B3, S_B2_C3, S_C2_A3 : Semaphore (=0);

2) Pseudo-code

Processus P1 **(01.50 pts)**

```
{
        A1;
        V(S_A1_B2);

        P(S_C1_A2);
        A2;
        V(S_A2_B3);

        P(S_C2_A3);
        A3;
}
```

Processus P2 **(01.50 pts)**

```
{
      B1;
```

```
        V(S_B1_C2);

        P(S_A1_B2);
        B2;
        V(S_B2_C3);

        P(S_A2_B3);
        B3;
}
```

Processus P3 **(01.50 pts)**

```
{
        C1;
        V(S_C1_A2);

        P(S_B1_C2);
        C2;
        V(S_C2_A3);

        P(S_B2_C3);
        C3;
}
```

Exercice 3 (07.50 Pts)

Déclaration des Sémaphores / variables (01.50 pts)

pont : Semaphore (= 1) // Semaphore contrôle le pont
mutex = 1 ; // protège les variables partagées. attA/ attB /cpt
goA : Semaphore(= 0) ; //bloquer une voiture du côté A tant qu'elle n'a pas le droit de passer,
goB : Semaphore (= 0) ;//bloquer une voiture du côté B tant qu'elle n'a pas le droit de passer,
attA : entier (=0) ; //nombres voitures en attente côté A
attB : entier (=0) ; //nombres voitures en attente côté B
tour = 'A' (ou 'B') // détermine le rôle A ou B
cpt = 0 ; // nombre de voitures déjà passées dans le lot courant (max 3)

Processus Voiture côté A ( ) (03.00 pts)

```
{
P(mutex) ;
 attA++;
 while (tour != 'A') or (cpt == 3 and attB > 0){
    V(mutex);
    P(goA);
 }
 P(mutex);
 attA-- ;
 cpt++ ;
V(mutex) ;
```

```
P(pont) ;
  < traverser() le pont>
V(pont) ;

P(mutex) ;
  if (cpt == 3 and attB > 0) or (attA == 0 and attB > 0)
        { tour = 'B'; cpt = 0; V(goB) ;}
  else
        {V(goA) ;}
V(mutex) ;
}
```

Processus Voiture côté B () (03.00 pts)

```
{
P(mutex)
  attB++;
  while (tour != 'B') or (cpt == 3 and attA > 0)
     { V(mutex);
        P(goB);
     }
 P(mutex);
 attB-- ;
 cpt++ ;
V(mutex) ;

P(pont) ;
  < traverser() le pont>
V(pont) ;

P(mutex) ;
  if (cpt == 3 and attA > 0) or (attB == 0 and attA > 0)
     { tour = 'A'; cpt = 0; V(goA) ; }
  else
     { V(goB); }
V(mutex);
}
```