TD2 : Inter-blocage, Concurrence et Synchronisation de processus par sémaphore

Exercice 1 :

Écrivons les algorithmes de P1 ,P2,P3 de façon a prévenir le non-interblocages

Processus P1 : (Producteur dans T1)

Processus P1

```
Début
   a0: <Produire un message>
   P(semVideT1)  // Attendre qu'il y ait de la place vide dans T1
   P(mutexT1)    // Accéder exclusivement à T1
   <Déposer le message dans T1>
   V(mutexT1)    // Libérer l'accès à T1
   V(semPleinT1)  // Indiquer qu'il y a un message dans T1
   Aller à a0    // Répéter le processus
Fin
```

Processus P2 : (Traitement dans T1, puis dépôt dans T2)

Processus P2

```
Début
   a1:
   P(semPleinT1)  // Attendre qu'il y ait un message dans T1
   P(mutexT1)    // Accéder exclusivement à T1
   <Prélève un message de T1>
   V(mutexT1)    // Libérer l'accès à T1
   V(semVideT1)  // Indiquer qu'il y a une place vide de plus dans T1
   <Traiter le message>
   P(semVideT2)  // Attendre qu'il y ait de la place vide dans T2
   P(mutexT2)    // Accéder exclusivement à T2
   <Déposer le résultat dans T2>
   V(mutexT2)    // Libérer l'accès à T2
   V(semPleinT2)  // Indiquer qu'il y a un résultat dans T2
   Aller à a1    // Répéter le processus
Fin
```

Processus P3 : (Consommation dans T)

Processus P3

```
Début
   a2:
   P(semPleinT2)  // Attendre qu'il y ait un message dans T2
   P(mutexT2)    // Accéder exclusivement à T2
   <Prélève un message de T2>
   V(mutexT2)    // Libérer l'accès à T2
   V(semVideT2)  // Indiquer qu'il y a une place vide de plus dans T2
   <Consommer le message>
   Aller à a2    // Répéter le processus
Fin
```

Scénario avec un tampon unique (T) :

Processus P1 : (Producteur dans T)

```
        Processus P1
Début
   a0: <Produire un message>
   P(semVideT)  // Attendre qu'il y ait de la place vide dans T
   P(mutexT)    // Accéder exclusivement à T
   <Déposer le message dans T>
   V(mutexT)    // Libérer l'accès à T
   V(semPleinT)  // Indiquer qu'il y a un message dans T
   Aller à a0    // Répéter le processus
Fin
```

Processus P2 : (Traitement dans T)

```
                Processus P2

Début

   a1:

   P(semPleinT)  // Attendre qu'il y ait un message dans T

   P(mutexT)    // Accéder exclusivement à T

   <Prélève un message de T>

   V(mutexT)    // Libérer l'accès à T

   V(semVideT)  // Indiquer qu'il y a une place vide de plus dans T

   <Traiter le message>

   Aller à a1    // Répéter le processus

Fin
```

Processus P3 : (Consommation dans T)

```
       Processus P3
Début
    a2:
    P(semPleinT)   // Attendre qu'il y ait un message dans T
    P(mutexT)     // Accéder exclusivement à T
    <Prélève un message de T>
    V(mutexT)     // Libérer l'accès à T
    V(semVideT)  // Indiquer qu'il y a une place vide de plus dans T
    <Consommer le message>
    Aller à a2     // Répéter le processus
Fin
```

Exercice 2

SemA est utilisé pour gérer l'accès à la section critique entre les processus **A**, **B**, **C**, et **F**.

SemB est utilisé pour synchroniser les processus **B**, **C**, **D**, et **E**.

Début

Initialiser SemA à 1
Initialiser SemB à 0 pour gerer la sysnchronisation

Processus A :
// attendre P(SemA)
//execution SectionCritiqueA()
//liberer V(SemA)

Processus B :
P(SemA)
SectionCritiqueB()
V(SemA)
//signaler V(SemB)

Processus C :
P(SemA)
SectionCritiqueC()
V(SemA)
V(SemB)

Processus D :
P(SemB)
SectionCritiqueD()
V(SemB)

Processus E :
P(SemB)
SectionCritiqueE()
V(SemB)

Processus F :
P(SemA)
SectionCritiqueF()
V(SemA)

Fin

Exercice 3

les algorithmes pour les processus Pi et Ps qui remplissent le tampon et impriment les objets :

1. **Processus Pi** (i=1…N) :
    - Tant que vrai :
        1. Attendre qu'une case du tampon soit libre (opération **P**).
        2. Déposer un message dans la case libre du tampon.
        3. Si c'est la dernière case du tampon, activer le processus **Ps** (opération **V**).
2. **Processus Ps** :
    - Attendre que le processus **Pi** (qui remplit la dernière case du tampon) l'active.
    - Imprimer tous les objets déposés dans le tampon.
    - Autoriser à nouveau les processus **Pi** à accéder au tampon.

N_m=nombre de message

Mutex_depo, Mutex_imp;

Mutex_depo ←-1; //initualisation du semaphore de depot de message

Mutex_imp ←-0; //initualisation du semaphore de d'impression Ps

M:taille du tampon;

processus Pi(i=1...N)

Début

N_m=0

Mi :<fabrique message>

P(Mutex_depo)

S=S-1 //S=1-1=0 et 0>=0 donc

<Deposer message>

N_m=N_m+1

Si N_m=M //taille de la tampon

Alors

V(Mutex_imp) //verrouillage et incrémentation de ps

Sinon

V(Mutex_depo)

FinSi

Aller a Mi

FIN

Processus Ps

Debut

Ms:P(Mutex_imp)

<impression message>

V(Mutex_imp)

FIN

Exercice 4

À quelle situation anormale peut conduire l'exécution de ces deux processus ?

L'exécution de ces deux processus peut conduire à un interblocage. Par exemple, si P0 acquiert le sémaphore s1 et P1 acquiert le sémaphore s2, ils seront bloqués indéfiniment, car aucun d'entre eux ne pourra libérer le sémaphore que l'autre attend.

1.**Solution au problème d'interblocage**: Pour éviter l'interblocage, nous pouvons utiliser l'algorithme du **banquier**. Avant d'allouer une ressource, chaque processus doit demander l'autorisation au gestionnaire de ressources. Le gestionnaire vérifie si l'allocation de la ressource entraînera un état sûr (c'est-à-dire qu'il existe une séquence d'exécution dans laquelle tous les processus terminent). Si l'allocation est sûre, la ressource est allouée ; sinon, le processus doit attendre.

Exercice 5 Ordonnancement avec Semaphore

1 indiquons la succession des differentes execution des processus P1,P2,P3

P1	P1	P1	P3	P3	P3	P3	P3	P2	P2	P1	P1
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**

2. Voici les états des sémaphores pour chaque cycle :

- **Cycle 1 : S1=0, S2=0**
- **Cycle 2 : S1=1, S2=0**
- **Cycle 3 : S1=1, S2=1**

Exercice 6

Temps : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

P1 | CPU | CPU | E/S | CPU | CPU | E/S | E/S | E/S | CPU | CPU | CPU | Fin

---

P2 | | | | | CPU | CPU | CPU | CPU | Fin

P3 | | | CPU | E/S | E/S | E/S | CPU | CPU | CPU | Fin

S1 | | | | Fin

S2 | | | | | Fin

Déroulement de l'exécution et changements d'états :

- **t = 0 :** P1 démarre l'exécution (priorité 1).
- **t = 3 :** P3 devient prêt (priorité 2).
- **t = 4 :** P2 devient prêt (priorité 3).
- **t = 5 :** P1 fait une opération d'E/S.
- **t = 6 :** P1 est préempté, P3 devient en cours d'exécution (priorité 2).
- **t = 8 :** P3 fait une opération d'E/S.
- **t = 9 :** P3 est préempté, P2 devient en cours d'exécution (priorité 3).
- **t = 10 :** P2 fait une opération d'E/S.
- **t = 11 :** P2 est préempté, P3 reprend (priorité 2).
- **t = 12 :** P3 fait une opération d'E/S.
- **t = 13 :** P3 est préempté, P1 reprend (priorité 1).
- **t = 15 :** P1 fait une opération d'E/S.

- **t = 16 :** P1 est préempté, P3 reprend (priorité 2).
- **t = 18 :** P3 fait une opération d'E/S.
- **t = 19 :** P3 est préempté, P2 reprend (priorité 3).
- **t = 20 :** P2 fait une opération d'E/S.
- **t = 21 :** P2 est préempté, P1 reprend (priorité 1).
- **t = 25 :** P1 fait une opération d'E/S.
- **t = 26 :** P1 est préempté, P3 reprend (priorité 2).
- **t = 28 :** P3 fait une opération d'E/S.
- **t = 29 :** P3 est préempté, P1 reprend (priorité 1).
- **t = 30 :** P1 termine son exécution.

Temps de sortie (Is(Pi)) de chaque processus :

- **Is(P1) :** 30 unités de temps.
- **Is(P2) :** 21 unités de temps.
- **Is(P3) :** 29 unités de temps.