TD n°1 : Sémaphores

Exercice 1

On considère la gestion du compte client par trois processus P1, P2 et P3 exécutant le programme suivant :

```
void Pi (int somme) {
      P(compte);
      solde = lire_solde();
      solde = somme + solde;
      écrire_solde(solde);
      V (compte);
}
```

avec :

- **solde** une variable partagée qui contient le solde courant du compte client déterminé par la fonction **lire_solde()**.
- **compte** un sémaphore.

1. Quel est l'intérêt de l'utilisation du sémaphore compte ?
2. Que doit être la valeur initiale du sémaphore compte ? Pourquoi ?
3. Dresser un tableau qui récapitule le déroulement simultané des trois processus P1(100), P2(200) et P3(300) sachant que le solde initiale est 1000DT. La file d'attente du sémaphore compte est implémentée par une file FIFO.

Processus	Instruction	Solde	compte.E	compte.F	Utilisation de la ressource (oui/non)

Exercice 2

1. Expliquer le concept des sémaphores et comment ils assurent le mécanisme de l'exclusion mutuelle.
2. Soit deux processus P1() et P2() qui se partagent deux sémaphores S1 et S2 initialisés à 0.

```
void P1() {
      A1();
      V(S2);
      P(S1);
      B1();
}
```

```
void P2() {
      A2();
      V(S1);
      P(S2);
      B2();
}
```

Quelle synchronisation a-t-on imposée sur les exécutions des fonctions A1(), A2(), B1() et B2(). Dresser le diagramme de précédence.

Exercice 3

Soient deux processus P1() et P2(). Le processus P1() effectue deux traitements A() et B(). Le processus P2() effectue un seul traitement C().

```
void P1() {
      A();
      B();
}
```

```
void P2() {
      C();
}
```

1. Une contrainte impose que le traitement B() doit être fait après la fin des traitements A() et C(). Modifier le code des deux processus P1() et P2() pour respecter cette contrainte moyennant des sémaphores.
2. Même question si la contrainte impose l'ordre d'exécution suivant : A() puis C() puis B().

Exercice 4

Étant donné N processus indépendants P1, P2, …, PN. On souhaite exécuter les processus dans cet ordre, c'est-à-dire chaque processus Pi doit être exécuté avant le processus Pi+1. Comment on peut respecter cette contrainte avec les sémaphores. Comment on doit transformer chaque processus Pi.

Exercice 5

Il s'agit de synchroniser les deux processus P1 et P2 avec des sémaphores S1, S2 et S3.

```
void P1() {                          void P2() {
      P(S1);                               P(S3);
      if(NB < N) {                         OK2();
            NB ++;                         P(S1);
            V(S1);                         NB --;
            OK1();                         V(S1);
            V(S3);                         traitement2();
            P(S2);                         V(S2);
            traitement1();           }
      }
      else {
            V(S1);
            traitement3();
      }
}
```

1. Quels sont les types et les rôles de chacun des sémaphores S1, S2 et S3 ?
2. Donner leurs valeurs initiales.

Exercice 6

Soit un ensemble de six processus séquentiels A, B, C, D, E ,F. Le processus A doit précéder les processus B, C et D. Les processus B et C doivent précéder le processus E. Les processus D et E doivent précéder le processus F.

1. Représenter le diagramme de précédence de ces processus.
2. Réaliser la synchronisation de ces processus en utilisant le **minimum** de sémaphores.

Exercice 7

Soit trois processus P1, P2 et P3. Le processus P1 produit des objets qu'il dépose dans un tampon T1 de taille N1. P2 prélève les objets contenus dans T1, les traite puis dépose les objets traités dans un tampon T2 de taille N2. P3 prélève les objets contenus dans T2 et les consomme. L'accès aux deux tampons ne peut se faire que par un et un seul processus.

```
void P1() {          void P2() {          void P3() {
  ProduireObjet();     RetirerObjet(T1);    RetirerObjet(T2);
  DéposerObjet(T1);    TraiterObjet();      ConsommerObjet();
}                      DéposerObjet(T2);  }
                     }
```

A l'aide des sémaphores, modifier le code source des processus P1, P2 et P3 afin de garantir le fonctionnement décrit ci-dessus.

Exercice 8 – Piscine

Une piscine peut accueillir au maximum N nageurs. Ce nombre N est le nombre de paniers disponibles pour les habits des nageurs. A l'entrée comme à la sortie les nageurs entrent en compétition pour l'acquisition d'une cabine pour se déshabiller et se rhabiller. La piscine dispose uniquement de C cabines (C << N). Chaque nageur effectue les opérations :

```
void nageur() {
  se_déshabiller(); /* nécessite un panier et une cabine */
  nager();
  se_rhabiller(); /* nécessite une cabine */
}
```

Nous pouvons assimiler ces nageurs à des processus concurrents; les cabines et les paniers sont les ressources partagées.

1. Quels sont les sémaphores nécessaires pour assurer la gestion de la piscine et e bon partage des ressources.
2. Modifier le code du processus nageur pour assurer le fonctionnement souhaité.