

TD Programmation Système

Série 5 : Accès Concurrent Synchronisation avec Sémaphores

Exercice 1 : Synchronisation (barrière)

Ecrire les procédures d'accès à un objet de type barrière dont le fonctionnement est le suivant :

1. la barrière est fermée à son initialisation,
2. elle s'ouvre lorsque N processus sont bloqués sur elle.

Définition de la barrière :

```
struct{
    Sema  Sema1 ;           /* Sémaphore de section critique */
    Sema  Sema2 ;           /* Sémaphore de blocage sur la barrière */
    int   Count ;           /* Compteur */
    int   Maximum ;         /* Valeur déclenchant l'ouverture */
} Barrière ;
```

Question 1 :

Complétez le code de la procédure d'initialisation :

```
void InitBarrière (Barrière *B; int Maximum)
{
    Init (B→Sema1, ... );
    Init (B→Sema2, ... );
    B→Count = ... ;
    B→Maximum = ... ;
}
```

Question 2 :

Complétez le code de la procédure d'attente donné ci-dessous :

```
void Wait (Barrière *B)
{
    Boolean Do_Wait = True ;
    int I ;

    ... ;
    B→Count++ ;
    if (B→Count == B→Maximum ){
        for (I=0 ; I < (B→Maximum-1) ; I++ ) ... ;
        B→Count = 0;
        Do_Wait = ... ;
    }
    ... ;
    if (Do_Wait) ... ;
}
```

Exercice 2 : Ordonnancement

Programmer un rendez-vous entre 3 processus cycliques P_1 , P_2 et P_3 en utilisant les opérations sur les *sémaphores*. P_3 ne peut exécuter sa 2^{ème} partie que si P_1 et P_2 se sont complètement exécutés. P_2 ne peut exécuter sa 1^{ère} partie que lorsque P_1 s'est entièrement exécuté.

<i>Processus P_1</i>	<i>Processus P_2</i>	<i>Processus P_3</i>
<u>Début</u> <u>Tantque vrai faire</u> <div>.....</div> <u>Fintantque</u> <u>Fin</u>	<u>Début</u> <u>Tantque vrai faire</u> <div>..... // partie_1 // partie_2</div> <u>Fintantque</u> Fin	<u>Début</u> <u>Tantque vrai faire</u> <div>..... // partie_1 // partie_2</div> <u>Fintantque</u> Fin