Systèmes d'exploitation

Exercices accompagnant le cours : Processus et threads (partie 3)

Objectifs des exercices

Implémenter des threads et comprendre les mécanismes de synchronisation

1 Implémentation des threads

1- Ecrire un programme qui crée deux threads. L'un affiche les entiers de 1 à 50, l'autre de 51 à 100.

2 Synchronisation des processus

2.1 Exercice 1

Soient trois processus concurrents P1, P2 et P3 qui partagent les variables n et out. Pour contrôler les accès aux variables partagées, un programmeur propose les codes suivants :

```
int n=10;
int out=0;
Semaphore mutex1 = 1 ;
Semaphore mutex2 = 1 ;
```

Code du processus P1

```
P(mutex1) ;
P(mutex2) ;
out=out+1 ;
n=n-1 ;
V(mutex2) ;
V(mutex1);
```

Code du processus P2

```
P(mutex2) ;
out=out-1 ;
V(mutex2) ;
```

Code du processus P3

```
P(mutex1) ;
n=n+1 ;
V(mutex1) ;
```

1- Cette proposition est-elle correcte?

2- Sinon, indiquer parmi les 4 conditions requises pour réaliser une exclusion mutuelle correcte, celles qui ne sont pas satisfaites?

3- Proposer une solution correcte.

2.2 Exercice 2

Soient maintenant trois threads concurrents T1, T2 et T3 qui partagent une variable globale x. Le thread 1 multiplie x par 3. Le thread 2 ajoute 5 à x. Le thread 3 soustrait 1 à x.

main

```
int x=0;
int main(int argc, char **argv){
        // Code qui cree les trois threads
        // avec les fonctions correspondantes
        return 0;
        }
```

Fonction du thread P1

```
void multiplie(){
        x=3*x;
}
```

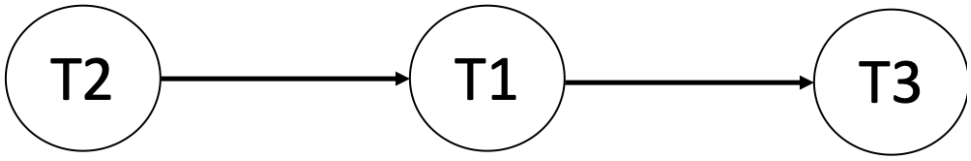
Fonction du thread 2

```
void ajout(){
        x=x+5;
}
```

Fonction du thread 3

```
void retire(){
        x=x-1;
}
```

1- Proposez une solution avec deux sémaphores pour assurer le graphe de dépendance suivant (ordre souhaité d'exécution des threads pour obtenir 14)?



3 Le problème producteur/consommateur

Considérons le problème producteur/consommateur, vu en classe.

```
Semaphore Vide=Max, Plein=0 ;
Message tampon [Max] ;

Producteur ( ){
        int ip=0;
        Message m;
        while(1){
                m = creerMessage();
                P(Vide);
                tampon[ip]=m;
                ip++;
                V(Plein);
        }
}

Consommateur ( ){
        int ic=0;
        Message m;
        while(1){
                P(Plein);
                m=tampon[ic];
                ic++;
                V(Vide);
        }
}
```

1- Comme vu en cours, ce schéma de synchronisation assure l'exclusion mutuelle entre le producteur et le consommateur. Mais que se passe-t-il dans le cas de plusieurs producteurs et plusieurs consommateurs?

2- Proposez une modification du code pour corriger le problème.