INTRODUCTION AUX SYSTEMES D'EXPLOITATION

TP2
Exclusion mutuelle / Sémaphores

S O M M A I R E

1.Le rendez-vous à l'aide des sémaphores

1.1. Présentation

Deux processus P1 et P2 réalisent des séquences indépendantes (le 1er processus exécute par exemple sleep (1) et le 2ème un sleep(5)) et doivent, pour continuer leur exécution, attendre que l'autre ait lui-même terminé sa propre séquence (voir exercice sur les rdv dans le TD2).

A l'aide des fonctions que vous développerez, réalisez la synchronisation entre deux processus P1 et P2 qui souhaitent établir un rendez-vous (tel que vu en TD). On considère que l'ensemble de sémaphores dont P1 et P2 ont besoin, a été préalablement créé par un programme sema.c indépendant.

1.2. Exercice - Rendez-vous avec P et V

Réaliser une première implantation de P1 et P2 avec des fonctions P et V que vous développerez.
Aide :

1. Dans le programme sema.c (adapté pour l'occasion), on crée dans un premier temps une clé avec un appel à la fonction `ftok()`, on crée par la suite un ensemble de sémaphores avec la fonction `semget()`, et enfin, on initialise l'ensemble des sémaphores crées à l'aide de la fonction `semctl()` en utilisant la commande SETALL (voir le cours).
2. Dans le programme de rendez vous (RDV.c), il faut ouvrir le sémaphore crée par le programme précédent (sema.c) en utilisant `semget()` mais sans l'option de création, il faut ensuite développer les fonctions P() et V() qui utilisent la primitive `semop()` et dont les prototypes sont les suivants :

   ```
   int P(int semid, int noSem)
   int V(int semid, int noSem)
   ```

   ou semid est l'identifiant de l'ensemble de sémaphores crée (retourné par la fonction `semget()`) et noSem le numéro de sémaphore (de l'ensemble) sur lequel on veut faire l'opération. Il ne faudrait pas oublier de supprimer l'ensemble de sémaphores après que le programme se termine à l'aide de la fonction `semctl()` mais cette fois-ci on précisera la commande IPC_RMID.

   Il y aura donc 2 programmes à faire à partir du fichier RDV.c, vous pourrez les appeler RDV1.c et RDV2.c, un pour chaque processus impliqué dans le RDV.

2. Lecteurs / écrivains

Implantez à l'aide des fonctions programmées dans l'exercice précédents la solution au problème des lecteurs / écrivains vu en TD.
Je reprends ci-dessous la solution pour plus de clarté.

Processus REDACTEUR	Processus LECTEUR

Redacteur () {
P(donnée) ;
ecrire (fichier) ;
V(donnée) ;
}
	Lecteur () {
	
	P (mutex_l) ;
	NbLecteur ++ ;
	If (nbLecteur == 1) {
	P(donnée) ;
	}
	V (mutex_l) ;
	
	lire (fichier);
	
	P (mutex_l) ;
	NbLecteur -- ;
	If (nbLecteur == 0) {
	V(donnée) ;
	}
	V (mutex_l) ;
	
	}

Faites 2 programmes : le rédacteur, que l'on appelera **redacteur.c** et le lecteur que l'on appelera **lecteur.c.**

Pour cet exercice, vous avez besoin de 2 sémaphores : **donnee** et **mutex_l**, vous pouvez vous appuyer sur le même fichier sema.c réalisé précédement pour la création des sémaphores. Attention à l'initialisation.

Redacteur.c

La seule difficulté ici (par rapport au problème des RDV) réside dans la fonction « ecrire(fichier) ». Cette fonction peut contenir les étapes suivantes :

- Ouverture du fichier concerné avec la fonction fopen()
- Ecriture d'un texte quelconque avec la fonction fprintf()
- Fermeture du fichier avec la fonction fclose()

Vous pouvez réutiliser les fonctions P() et V() de l'exercice précédent en les recopiant dans le code du Redacteur (et le Lecteur.c aussi).

Lecteur.c

Pour cette fonction, on se pose la question de la mise en œuvre de la variable *nbLecteur* qui est partagée entre l'ensemble des processus lecteurs et la fonction lire(fichier).

Concernant la fonction lire(fichier) :

- Ouverture du fichier avec fopen()
- Lecture avec la fonction fscanf()
- Affichage de ce qui a été lu
- Fermeture du fichier avec fclose()

Pour ce qui est de la variable nbLecteur, celle-ci se trouvera dans un fichier :

- Créez un fichier en dehors de votre programme (avec votre éditeur de texte préféré ou le shell), on le nommera **nombre**. Dans ce fichier, écrivez la valeur 0 (grâce à l'éditeur ou le shell). Cela correspondera à la valeur initiale de la variable nbLecteur.
- Dans votre programme :

- Ouvrez le fichier avec la fonction fopen()
- Pour lire la valeur dans le fichier, vous pouvez utiliser la fonction fscanf()
- Une fois la valeur testée et eventuellement incrémentée, remettez là dans le fichier avec un rewind() (pour remettre le pointeur de fichier au début) puis un fprintf().
- N'oubliez pas de fermer le fichier avec un fclose().

Afin de simuler un opération de lecture ou d'écriture qui prend du temps, vous pouvez utiliser la fonction sleep(). Cela permettera de visualiser les attentes.

Vos programmes doivent être suffisamment commentés (cela sera pris en compte dans la note).