3ème Année Licence SI (2021/2022) Durée : 1h30

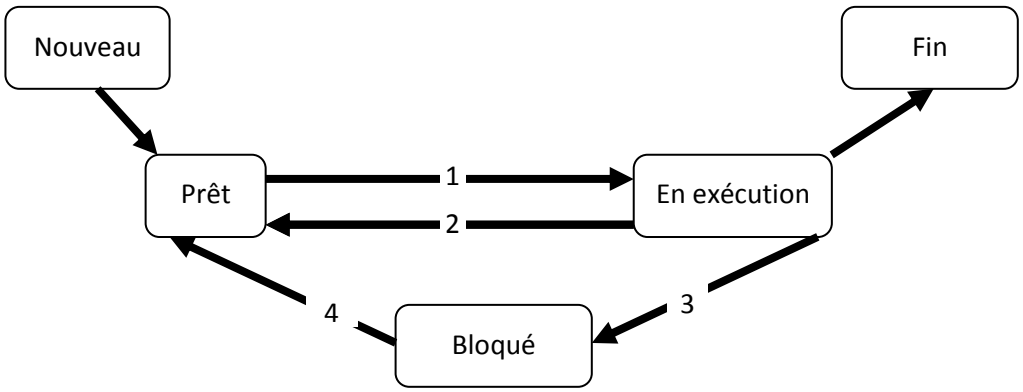
Exercice 1 (4 pts) :

1. Que font les fonctions : ***pthread_self()*** , *pthread_join() ;* *fork() ?*
   0.5 pthread_self(): Retourne le TID du thread.
   0.5 *pthread_join():* Le Processus exécutant cette fonction attend la fin de l'exécution du thread passé en arguments, pour continuer sa propre exécution.
   1 *fork():* est un moyen offre par SE afin de créer des processus, par duplication d'un processus existant.
2. Quelle est la différence entre **section critique** et **ressource critique** ?
   1 Une **section critique** (SC): est un ensemble d'instruction d'un programme qui peuvent engendrer des résultats imprévisibles (ou incohérents) lorsqu'elles sont exécutées simultanément par des processus différents. Une SC est une suite d'instructions qui opèrent sur une ou plusieurs ressources partagées (critiques) et qui nécessitent une utilisation **exclusive** de ces ressources.

   1 On appelle **ressource critique** tout objet informatique qui peut faire l'objet d'un accès concurrent (ou simultané) par plusieurs processus.

Exercice 2 (6 pts) :

Soit le schéma suivant décrivant les transitions d'un processus. Précisez dans le tableau ci-dessous à quoi correspondent les transitions numérotées par 1, 2, 3, 4 et précisez quels sont les événements qui provoquent chacune de ces transitions.



N° transition	Transition 0.5	Événements 1/évenement
1	Élection	Le processus obtient l'affectation du processeur
2	Interruption	Fin de quantum ou arrivée d'un processus plus prioritaires
3	Attente	Attente E/S ou attente d'un événement
4	Fin d'attente	Fin E/S ou arrivée d'un événement.

Exercice 3 (5 pts):

Considérons un objet (une base de données par exemple) qui n'est accessible que par deux catégories d'opérations : les lectures et les écritures. Plusieurs lectures (consultations) peuvent avoir lieu simultanément ; par contre les écritures (mises à jour) doivent se faire en exclusion mutuelle. Ce problème induit plusieurs types de contraintes de synchronisation qui sont:

- Si un lecteur demande à lire et qu'il y a une écriture en cours, la demande est mise en attente. De même que si un rédacteur demande à écrire et qu'il y a au moins une lecture en cours, la demande est mise en attente.

- Si un rédacteur demande à écrire et qu'il y a une écriture en cours, la demande est mise en attente.

Question :

- Écrire l'algorithme des processus lectures et rédacteur en utilisant les sémaphores.

```
var S1,S2 : sémaphore init 1,1 ;
nl : entier init 0 ;
```

Processus Lecteur	Processus Rédacteur
Début Début . . . P(S2) nl := nl + 1 si nl=1 alors P(S1) finsi V(S2) Lecture P(S2) nl := nl − 1 si nl=0 alors V(S1) finsi V(S2) . . . Fin	Début P(S1) Ecriture V(S1) . . . Fin

Exercice 4 (5 pts) :

En se basant sur les sémaphores, écrire un programme utilisant 3 threads afin d'afficher la séquence suivante: A, B, C, A, B, C...,

Où, le thread T1 affiche: A, le thread T2 affiche: B, et le thread T3 affiche: C,

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

sem_t my_sem1,my_sem2,my_sem3;  0.5

void *T1 (void * arg){     1
  while (1) {
   sem_wait(&my_sem1);
   printf ("A,");
   sem_post(&my_sem2);
        }
 }
void *T2 (void * arg){      1

  while (1) {
  sem_wait(&my_sem2);
   printf ("B,");
   sem_post(&my_sem3);

 }
}
```

```
void *CT3 (void * arg){            1
  while (1) {
  sem_wait(&my_sem3);
   printf ("C,");
   sem_post(&my_sem1);

 }
}
void main (int ac, char **av){
 pthread_t th1, th2,th3;
 sem_init (&my_sem1, 0, 1);                .5
 sem_init (&my_sem2, 0, 0);                .5
 sem_init (&my_sem3, 0, 0);              0.5
 pthread_create (&th1, NULL, &T1, NULL) ;
 pthread_create (&th2, NULL, &T2, NULL) ;
 pthread_create (&th3, NULL, &T3, NULL) ;
 pthread_join (th1, NULL);
 pthread_join (th2, NULL);
 pthread_join (th3, NULL);
}
```