Exercices sur les variables sémaphores

Exercice 1

On désire réaliser un "rendez-vous" pour 2 threads.

Soient deux threads A et B. Le threads A effectue les opérations a_1 puis a_2 et le thread B effectue les opération b_1 puis b_2. A l'aide des sémaphores garantir que l'opération a_1 se produira toujours avant b_2 et l'opération b_1 se produira toujours avant a_2.

Exercice 2

Soit un programme comportant 20 threads. Chaque thread va exécuter à un moment donné la fonction `foo()`.

On aimerait garantir qu'au maximum 5 threads exécutent la fonction `foo()` en même temps.

Comment garantir ce comportement à l'aide de sémaphores?

Exercice 3

Comment peut-on implémenter une barrière de synchronisation réutilisable en utilisant seulement des sémaphores ?

Veuillez respecter la sémantique des barrières POSIX: une fois tous les threads passés, la barrière est réinitialisée à sa valeur initiale.

Voici la structure de barrière à compléter :

```
typedef struct {
    // complete please
} barrier_t;
```

Et ci-dessous les fonctions à implémenter :

```
void barrier_init(barrier_t *bar, int count) {
    // complete please
}

void barrier_wait(barrier_t *bar) {
    // complete please
}
```

```
void barrier_destroy(barrier_t *bar) {
    // complete please
}
```

Bien entendu, il vous faudra aussi développer un programme de test permettant de vérifier que votre implémentation fonctionne correctement. Encore une fois, vous devriez pouvoir remplacer n'importe quel code utilisant les barrières POSIX avec votre implémentation à base de variables de condition.

Exercice 4

Comment peut-on implémenter une un sémaphore en utilisant seulement une variable de condition et un verrou ?

Voici la structure de sémaphore à compléter :

```
typedef struct _sephamore_t {
    // complete please
} sephamore_t;
```

Et ci-dessous les fonctions à implémenter :

```
void sephamore_init(sephamore_t *seph, int count) {
    // complete please
}

void sephamore_destroy(sephamore_t *seph) {
    // complete please
}

void sephamore_wait(sephamore_t *seph) {
    // complete please
}

void sephamore_post(sephamore_t *seph) {
    // complete please
}
```

Bien entendu, il vous faudra aussi développer un programme de test permettant de vérifier que votre implémentation fonctionne correctement. Encore une fois, vous devriez pouvoir remplacer n'importe quel code utilisant les sémaphores POSIX avec votre implémentation à base de variable de condition et de verrou.