Université Toulouse III – Paul Sabatier
118 route de Narbonne
31062 Toulouse cedex 9

Travaux pratiques – n°3
Threads Posix et synchronisation de type « moniteur »

Documentation

Le concept de « moniteur » peut être utilisé pour synchroniser des threads Posix en utilisant des conditions Posix (type pthread_cond_t). Voir la documentation à votre disposition sous Moodle.

Gestion d'une voie unique

On considère des activités parallèles (threads) qui simulent le comportement de véhicules circulant dans un certain sens :

Vehicule (monSens)

Boucle nbFois sur :

Rouler normalement sur la voie à double sens, dans monSens

demanderAccesVU (monSens)

Rouler sur la voie unique dans monSens

libererAccesVU()

Sur la portion de voie unique considérée, afin d'éviter des collisions, ne peuvent circuler que des véhicules allant dans le même sens.

Écrire une application dans laquelle N1 véhicules allant dans un sens et N2 véhicules allant dans le sens inverse se synchronisent (ce qui revient à écrire demanderAccesVU() et libererAccesVU()) de manière à respecter les contraintes imposées dans les variantes suivantes :

- **Variante 1** : un seul véhicule au plus peut circuler sur la voie unique.
- **Variante 2** : un nombre illimité de véhicules peuvent circuler sur la voie unique à condition qu'ils aillent dans le même sens.

Rappel : Si les affichages sont trop rapides, il est possible de temporiser l'exécution d'un thread pendant quelques microsecondes ou nanosecondes à l'aide des primitives :

int usleep (useconds_t usec) ;

int nanosleep(const struct timespec *req, struct timespec *rem) ;

Voir le manuel en ligne pour leur utilisation (man 3 usleep ou man 2 nanosleep).

On peut utiliser une valeur générée aléatoirement (voir les fonctions srand et rand) pour varier les délais d'attente d'un thread à un autre.

Mais, **attention**, la temporisation n'est pas là pour résoudre les problèmes d'accès concurrents à des variables partagée. En d'autres termes : toute exécution d'une application parallèle doit donner un résultat cohérent **sans** temporisation !