Programmation Temps Réel - TP3
Modèle lecteurs/écrivains et verrous 'rwlock'

A - Consignes générales

Réalisation des exercices :

- Exercices 1 et 2 : mettez les réponses dans un fichier texte
- Exercices 3 et 4 : Les programmes doivent être réalisés en langage C et tourner sous linux.
- Privilégier un style clair et lisible
- Nommer les fonctions et les variables avec des noms appropriés
- Commentez votre code
- Soignez votre indentation
- **La compilation ne doit retourner ni erreur, ni warning.**

Remise des exercices :

- Chaque fichier doit être nommé avec le numéro de l'exercice (ex: "exo1" ou "exercice1").
- **Déposez les fichiers dans le dépôt moodle prévu à cet effet sur** https://moodle.univ-paris8.fr/moodle/course/view.php?id=1373

B - Énoncés des exercices

Exercice 1 - Verrous rwlock

Récupérez le fichier `pthread_rwlock_exemple.c` sur la page Moodle du cours. Examinez le code.

1. Indiquez les commandes utilisées pour compiler/exécuter le programme et donnez un exemple de retour du programme.
2. Quel type d'ordonnancement est utilisé ? Lecteurs favorisés, écrivains favorisés ou équitable ?
3. Commentez l'exemple de retour du programme de manière à prouver ce que vous affirmez.

Le fichier `pthread_rwlock_retour.txt` donne un exemple de retour du même programme sous MacOS.

1. Quel type d'ordonnancement est utilisé ? Commentez l'exemple de retour du programme de manière à prouver ce que vous affirmez.

Exercice 2 - Comprendre les verrous rwlock, fonctionnement / limitations

Recherchez de la documentation pour comprendre comment est effectué l'ordonnancement avec les rwlock POSIX en C sous Mac et sous Linux et comment on peut modifier l'ordonnancement par défaut.

1. Indiquer les informations trouvées concernant l'ordonnancement sous Mac et sous Linux et où vous les avez trouvées (souvenez-vous qu'une partie du code de MacOS est basé sur BSD)
2. Indiquez si les informations trouvées correspondent à ce que vous avez observé précédemment.
3. Peut-on favoriser les lecteurs ? les écrivains ? faire un système équitable ? si oui comment ?
4. D'après les informations données dans la documentation, peut-on garantir une exécution identique d'un programme utilisant les `pthread_rwlock` sous Linux et sous MacOS ?

Si vous vous demandez comment savoir si la « Threads Execution Scheduling option » est supportée sur votre système, utilisez le petit programme `support_thread_posix.c` *fourni dans le cours. Voici le retour de ce programme sous MacOS :*

```
$ ./a.out
_POSIX_VERSION=200112
avec thread posix
sans ordonnancement des threads
```

Exercice 3 - Implémentation du modèle lecteurs/écrivains avec des mutex

Modifier le code précédent pour implémenter une version équitable avec des mutex (l'algo est dans le cours).

1. Commentez votre code pour expliquer ce que vous faites
2. Commentez un exemple de retour du programme qui montre que tout se passe comme souhaité.

Exercice 4 - Variante du modèle lecteurs/écrivains

Deux groupes A et B d'étudiants veulent utiliser la même salle pour réviser... Des étudiants du même groupe peuvent utiliser la salle en même temps, mais pas des étudiants de groupes différents.

1. Expliquez en quoi ce problème est proche du problème précédent
2. donnez l'algorithme à appliquer en pseudo-code
3. Faites une copie du code que vous avez écrit en 3 et modifiez le pour implémenter ce problème d'utilisation de salle de manière équitable pour les 2 groupes

Si vous avez fini avant la fin du cours, essayez d'implémenter les versions favorisant un groupe (lecteurs/écrivains ou groupeA/groupeB) pour chacun des exercices 3 et 4.