TD UNIX n°2

I) Sémaphores

Le problème des lecteurs et des rédacteurs :
On dispose d'une boîte à lettres en mémoire dans laquelle on peut déposer et retirer des messages. On ne se préoccupera pas ici du problème de saturation de la boîte.
Certains threads sont des écrivains qui rédigent des messages qu'ils déposent dans cette boîte, les autres sont des lecteurs qui les lisent.
Le fonctionnement que l'on désire est le suivant :

- Lorsqu'un écrivain veut écrire il faut qu'il n'y ait aucun autre écrivain pour éviter qu'ils ne modifient la même zone de mémoire. Mais il faut qu'il n'y ait non plus aucun lecteur sinon on risquerait qu'un lecteur lise un message incomplet parce que le rédacteur n'a pas fini de l'écrire.
- Les lecteurs peuvent accéder en concurrence à la boîte car ils font des lectures en mémoire de messages de sorte qu'ils ne se gênent pas entre eux. Mais ils ne peuvent pas rentrer s'il y a un écrivain pour ne pas lire un message en cours de rédaction.

On va donc avoir un verrou qui contrôle l'accès à cette boîte à lettres (autorisé/interdit)

1°) Quand un écrivain ferme-t-il ce verrou ? Quand l'ouvre-t-il ? Quand est-il bloqué ?

2°) Quand un lecteur ferme-t-il ce verrou ? Quand l'ouvre-t-il ? Quand est-il bloqué ?

Faire la correction en utilisant les notions de 1er lecteur et dernier lecteur puis demander comment on peut implémenter ça.

3°) Ecrire l'algo des écrivains et celui des lecteurs qui traite le problème de 1er et dernier lecteur.

4°) On lance simultanément 1 écrivain et 2 lecteurs
Ecrire un message correspond à 3 instructions (2 pour préparer le message 1 pour l'écrire)
Lire un message correspond à 1 instruction
Les traitements correspondent à 6 instructions
Faire un chronogramme de ce qu'il se passe (1 ligne par étape)

II) Threads et mutex
On veut écrire un programme qui utilise 2 threads pour rechercher la valeur maximale des éléments d'un tableau de grande taille. Ce programme initialise un tableau de N entiers naturels et une variable globale 'max' à 0. Puis il crée 2 threads qui exécutent la même fonction (avec paramètre comme vu au TD précédent). Chaque thread traitera une moitié du tableau et mettra à jour variable 'max' chaque fois qu'il trouve une valeur supérieure au max actuel dans sa partie du tableau.
Le processus attend la fin des 2 threads puis affiche le maximum ainsi calculé.
Ce programme ressemble beaucoup à celui écrit au TD1 sauf qu'il fait une recherche de maximum au lieu d'une recherche d'une valeur donnée mais comme la modification de la variable 'max' sera faite en concurrence par les 2 threads il faut la protéger par un sémaphore de type mutex afin que cette modification soit faite en section critique.

Ecrire ce programme en C