Real Time Programming – TD-TP#

Readers/Writers model and 'rwlock' lock

A - General instructions

Exercises realization:
• Exercises 1 and 2: Put the answers in a text file
• Exercises 3 and 4: The programs must be done in C language and run on linux
• Favor a clear and readable style
• Name functions and variables with appropriate names
• Comment on your code
• **The compilation must not return any error or warning.**

Delivery of exercises:
• Each file must be named with the exercise number (ex: "exo1" or "exercise1").
• **Send your work by email** roula.osta@gmail.com
• **Indicate [PTR] at the beginning of the subject (with the brackets).**
• **Indicate your name and surname! (Otherwise, I do not know who to give the note!).**

B - Exercise Statements

Exercise 1 – Rwlock Lock

Get the `pthread_rwlock_example.c` file from your instructor. Examine the code.

1. Specify the commands used to compile / run the program and give an example of a return from the program.
2. What kind of scheduling is used? Favored readers, favored writers or fair? Or none of three?
3. Discuss an example of a return of the program to prove what you are saying (if it is not obvious in the first result obtained, restart the execution to see if the next log is clearer).

The log_MacOS_High_Sierra.txt and log_Ubuntu14.4.txt files give an example of a return from the same program under MacOS and Ubuntu.

1. What kind of scheduling is used? Comment on the example of return of the program to prove what you are saying.

Exercise 2 - Understanding rwlock locks, operation / limitations

Search for documentation (in the manual and on the internet) to understand how it is done scheduling with POSIX rwlock in C on UNIX, Mac and Linux and how you can

modify the default scheduling (the basic operation of rwlock does not interest us here, what we want to know which is favored by default and how to modify it).
1. Indicate the information found about scheduling on Mac, and Linux, and where you can have found it (sources).
The man_rwlock_MacOS_High_Sierra.txt file contains the manual page obtained under MacOs High Sierra. For more information you have to go see on the internet: notice that the page of MacOS manual is an entry from the "BSD Library Functions Manual". If you can not find information about MacOS on the internet, you can search for BSD ...
2. Indicate if the information found matches what you observed previously.
3. Can we favor readers? the writers ? to make a fair system? If yes, how?
4. Based on the information provided in the documentation, is it possible to guarantee the identical program using `pthread_rwlock` under Linux and under MacOS?

If you are asking how to know if the "Threads Execution Scheduling option" is supported on your system, use the small support_thread_posix.c program provided in the course. Here is the return of this program under MacOS High Sierra supported on your system, use the small program support_thread_posix.c provided in the course. Here is the return of this program under MacOS High Sierra:

```
$ ./a.out
_POSIX_VERSION=200112
with posix thread
without threads scheduling
```

Exercise 3 - Implementing the readers / writer model with mutex

Modify the previous code to implement a fair version with mutexes (the algo is in the course).
1. Comment on your code to explain what you are doing.
2. Comment on an example of a return of the program that shows that everything is going as desired.

Exercise 4 - Variant of the readers / writer model

Two groups A and B of students want to use the same room to revise ... Students in the same group can use the room at the same time, but not students from different groups.
1. Explain how this problem is close to the previous problem
2. Give the algorithm to apply in pseudo-code version.
3. Make a copy of the code you wrote in 3 and modify it in order to implement this problem of room use in a fair way for both groups.