Important: the Q&A for the exercise will take place at a public forum Piazza only. Critical updates about the HW will be published in pinned notes in the piazza forum. These notes are mandatory and it is your responsibility to be updated. A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
- Be polite, remember that course staff does this as a service for the students
- You’re not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
- When posting questions regarding hw3, put them in the hw3 folder

Postponements can be authorized only by Arthur, the TA in charge. In case you need a postponement, contact him directly.

Dry part submission instructions:

1. Please submit the dry part to the electronic submission of the dry part on the course website.
2. The dry part submission must contain a single dry.pdf file containing the following:
   (a) The first page should contain the details about the submitters - Name, ID number and email address.
   (b) Your answers to the dry part questions.
3. Only typed submissions will be accepted. Scanned handwritten submissions will not be accepted.
4. Only PDF format will be accepted.
5. You do not need to submit anything in the course cell.

Question 1

1.1 Describe the advantages and disadvantages of a multiple-readers/single-writer lock (RWLock). Please give example scenarios for each case.

1.2 "Fair" and "unfair" RWLock:
   (a) Explain the differences between a "fair" and an "unfair" RWLock.
(b) Provide an example where a Reader-Preferred RWLock (RPRWLock) is better suited for the task, and an example where a writer-preferred lock (WPRWLock) is better suited for the task.1
(c) What is the advantage of using an RPWRlock or an WPRWLock instead of a fair RWLock? When does it apply?

Question 2

Let BST be a simple binary-search-tree that is not balanced. Assume the tree structure provides the following operations:

- \(\text{insert}(BST, V)\): Inserts a value \(V\) in the tree, while maintaining the proper sorting order.
- \(\text{remove}(BST, V)\): Removes a value \(V\) from the tree.
- \(\text{contains}(BST, V)\): Returns true if-and-only-if the value \(V\) is present in the tree.

2.1 Describe the differences between coarse-grain locking and fine-grain locking.

2.2 Describe the impact of each of these approaches on the data structure above, in terms of runtime performance and complexity of implementation.

2.3 Is one approach strictly better than the other? If so, explain why. If not, give an example where one is strictly better than the other and example where the opposite is true. Explain how your examples support your claims.

2.4 What is the minimal number of locks needed to implement fine-grain locking in the tree above? Explain your answer.

2.5 With fine-grain locking, how many locks need to be acquired in each
   - (a) Contains operation.
   - (b) Insert operation. In your answer, distinguish between the search-phase and the actual insertion.
   - (c) Remove operation. In your answer, distinguish between the search-phase and the actual removal.

Question 3

3.1 In Linux, all threads in a process share the same memory space, and share access to the same resources (e.g. open files). In general terms, please explain

   - (a) how the operating system allows each thread to have a separate user-space stack.
   - (b) why each thread has a separate kernel stack

3.2 Sometimes users do not use pthreads to implement multithreaded applications. Instead they implement the threads in user space without the help from the OS. This method is known by many names such as: User-Level Threads, Green Threads and Cooperative multi-threading. A common characteristic of such threading systems is that they are not preemptive. That is, a thread has to yield the CPU to other threads (invoke a context switching manually).

   In the above description regarding the sentence “ has to yield the CPU”, does the thread use sched_yield to switch to other threads? If your answer is yes, please explain.

   If your answer is no, explain why not and explain in general terms what this yield needs to do to switch from one thread to the next.

3.3 Please list at least two primary disadvantages of such a threading system, and one primary advantage.

---

1As their names suggest, a reader-preferred RWLock will allow new readers to continue even when a writer is pending, at the cost of starving writers, while a writer-preferred lock will allow a new writer to continue after a writer released the lock even when readers are pending, at the cost of starving readers. The RWLock implementation that we saw in the tutorial is that of an RPRWLock. By adding a check whether a “writer is pending” we can turn that lock into a WPRWLock.