Locks

February 7, 2020

In this optional assignment, you are to implement `sleep` and `wakeup` primitives and fix the memory leak in `thread_exit` in the `NPTlib` library. This assignment will be graded if the output of the `make run` in your assignment-2 submission is the same as expected.

1 NPTlib interfaces

In this assignment, you have to implement two new interfaces: `sleep` and `wakeup`, and change the current `wait_for_all` interface.

2 `sleep`

```
‘‘void sleep(struct lock *l)’’
```
takes a `struct lock` as input. `struct lock` contains a `wait_list`. In sleep, you have to insert the current thread to the end of the `wait_list` in the `struct lock` and `schedule` a new thread without adding the `cur_thread` to the `ready_list`.

3 `wakeup`

```
‘‘void wakeup(struct lock *l)’’
```
takes a `struct lock` as input. `wakeup` can be called even if no thread is sleeping on the input lock. If the `wait_list` in `struct lock` is not empty, `wakeup` pops the first thread form the `wait_list` and adds to the `ready_list`.

4 wait_for_all

The `wait_for_all` routine yields until there are no other threads to `schedule`, and no thread in sleeping on the `wait_list` of a `struct lock`.

5 Memory leak

You need to free the stack and `struct thread` corresponding to exited thread. You have to make sure that at a given program point, there is at most one exited thread whose stack and `struct thread` are not reclaimed.

6 race1.c

You have to insert a call to `thread_yield` in `foo` routine in `race1.c` file, in a way that `assert(counter == 2)` in `main` fails. After getting the assertion failure, you need to insert `acquire` and `release` in `foo` and `bar` to prevent the assertion failure.

7 Implementation

You have to implement everything in the `thread.c` and `race1.c` files. You can change the `struct thread` and add additional global lists if required.

8 Environment

For this assignment, you need to synchronize the assignment repo from `https://github.com/Systems-IIITD/NPTlib` (run `git pull` in the existing repo). ``‘‘make’’`` command builds the test cases (`race1.c, race2.c, and leak.c`) and the `NPTlib` library. You are not supposed to change the test cases `race2.c` and `leak.c`. You have to start with the same `thread.c` implementation that you have submitted in assignment-2.

8.1 Design documentation

You also have to submit design documentation along with your implementation; otherwise, the assignment will not be graded. Answer the following questions in your design documentation.

- Paste your `struct thread` structure.
- Paste any new global variables or struct that you have added to the existing code.
- Paste your code corresponding to `sleep`.
- Paste your code corresponding to `wakeup`.
- Paste your code corresponding to the `foo` routine in `race1.c`.
- Dump the output of ``‘‘make test2’’``.
- Does running `race2` cause deadlock in your submission?

- Does your strategy for eliminating memory leak is different from what you suggested in the `assignment-2` design documentation. If yes, please highlight the changes.

8.2 How to submit.

To be done individually. Submit a zip folder that contains three files: “thread.c”, “race1.c”, and design documentation (in pdf format). Please make sure that your implementation is not printing any debug messages before submitting the final code. The submission link is on backpack.