Project 3

Rahjee Manuel

Odalys Rodriguez

COP 4634

7 Nov 2021

Each of the cat threads will periodically sleep. When they are awake, they will look at the driveway to check on the lizards. If there are too many lizard threads on the driveway then they will play with them causing the program to terminate. So, we must not allow too many lizards to cross the driveway at once, but we must allow the maximum number of lizards to cross simultaneously. However, we are not allowed to use busy waits.

In the code we are using a mutex and a condition variable to create a counting semaphore. The mutex is used as a lock to increment and decrement the semaphore value exclusively. The semaphore value is initialized to the maximum number of lizards allowed to cross. When a lizard begins to cross, the value will be decremented and incremented after they cross. Therefore, the lizards will only cross if the value is greater than zero. In order to prevent busy waiting, a condition variable is used to suspend a thread when the semaphore value is zero. When a thread finishes crossing it will notify a waiting thread that the semaphore value has changed so the thread can continue.

WORLDEND (s)	Maximum Number of Lizards Crossing	Lizards safe?
30	4	Yes
60	4	Yes
90	4	Yes
180	4	Yes
180	8	Yes
180	10	Yes
180	12	Yes
180	16	Yes

When developing the program, the main issue was creating the counting semaphore without a busy wait. Using a while loop to continuously check the semaphore value was the easiest solution. However, this used a lot of CPU time. In order to prevent the waste of CPU time, a conditional variable was used to suspend the threads until the semaphore value was changed.

Additionally, there were output issues because the console printed out partial strings from different threads at different times. To ensure the output statements were printed as a single block, the strings were concatenated.