Ethan Tran

COP4634

11/13/2024

Dr. Mishra

Lab Report: Thread Synchronization for Lizards and Cats Simulation

1. Introduction

The goal of this project is to implement a multi-threaded simulation where a herd of lizards attempts to cross a driveway to reach food, while avoiding being caught by two monitoring cats. The challenge is to synchronize the lizards' crossing of the driveway to ensure that no more than a maximum number of lizards cross at any given time and that the cats do not detect them.

2. Problem Description

In this simulation:

- **Lizard Threads**: Each lizard thread wakes up after a random amount of sleep, attempts to cross the driveway to the monkey grass, eats, and then returns home. The challenge is to ensure that not too many lizards attempt to cross the driveway at once, which would lead to the cats detecting and "playing" with the lizards.
- **Cat Threads**: The two cat threads periodically check the number of lizards crossing the driveway. If the number exceeds the allowed maximum, the cats terminate the program.

The problem is solved using synchronization techniques, where a maximum number of lizards crossing the driveway at once is enforced, preventing busy waits and ensuring safe crossing.

3. Approach and Solution

Synchronization Techniques

The key part of the solution was implementing proper synchronization to avoid race conditions and manage the crossing of the lizards.

Algorithm

- **Lizard Life Cycle**:
  1. A lizard sleeps for a random amount of time.
  2. The lizard checks if it is safe to cross the driveway (using a semaphore).
  3. If safe, the lizard crosses the driveway.
  4. The lizard eats in the monkey grass for a random amount of time.
  5. The lizard returns home to the sago palm when the crossing is safe again.

- **Cat Life Cycle**:
    1. The cat sleeps for a random amount of time.
    2. The cat checks if too many lizards are crossing.
    3. If more than the allowed number of lizards are crossing, the cat prints a message and terminates the simulation.

Modifications to the Starter Code

The starter code provided a basic structure for the lizard and cat threads, as well as hints for the synchronization mechanisms. Key modifications included:

- **Implementing Semaphore and Mutex Locks**:
    - A semaphore was used to control the amount of lizards allowed to cross.
    - Mutexes were added to protect shared variables and to prevent race conditions.
- **Random Sleep Times**:
    - Randomized the sleep times of the lizards and cats to simulate a more dynamic and realistic environment.
- **Cat Check Logic**:
    - Added logic for the cats to periodically check if too many lizards were crossing, with a safe termination mechanism when the limit was exceeded.

4. Testing

Test Setup

Several tests were run to evaluate the behavior of the program under different conditions. The following parameters were varied.

Table 1: Test Results

WORLDEND (s)	**Max Lizards Cross**	**Lizards safe?**
30	4	Yes
180	4	Yes
30	6	No
60	3	Yes
120	5	Yes

From the results, it is evident that the number of lizards crossing the driveway safely depends on the simulation time and the number of lizards. In tests where more than the maximum allowed number of lizards crossed the driveway, the cats detected the situation, and the simulation was terminated.

Issues Encountered

During the development of this simulation, several issues were encountered:

1. **Race Conditions**: Early versions of the program did not correctly synchronize the lizard threads, leading to inconsistent results where the number of lizards crossing could exceed the allowed limit.
2. **Deadlocks**: Incorrect use of mutexes resulted in deadlocks in some cases, where lizards were unable to cross because the mutex was held for too long.
3. **Cat Termination Logic**: The cat logic needed refinement to ensure that the cats did not prematurely terminate the simulation and that they correctly identified when the number of crossing lizards exceeded the allowed limit.

5. Discussion

The program was successful in simulating the lizards' crossing behavior and the cats' monitoring. By utilizing synchronization mechanisms like mutexes and semaphores, we ensured that the lizards did not exceed the maximum allowed crossing number, while allowing for efficient and safe thread execution.

6. Conclusion

This project demonstrated the use of multi-threading and synchronization primitives to manage concurrency and ensure safe interaction between threads. By simulating the scenario of lizards crossing a driveway while avoiding detection by cats, the project provided practical experience in managing thread synchronization in a real-world problem. The program successfully prevents race conditions and ensures that the simulation behaves as expected, meeting all specified requirements.