Homework 3 Report

Concurrent programming, ID1217

Group members
[Sai Santhoshi Srinithya , darbha@kth.se]

Task 1: The Hungry Birds Problem(one producer - multiple consumers)

Implementation

This problem consists of a parent bird (producer) and multiple baby birds (consumers) sharing a bowl of worms. The baby bird eats worms from the bowl all the time. When the bowl is empty, the baby bird who realizes this condition wakes up the parent bird, who refills the bowl.

- Each baby bird thread attempts to eat a worm by reducing the bowl value inside a critical section protected by`mutex`.
- If the bowl is empty, the baby bird signals the parent using `emptyBowl` and waits on `fullBowl` until the parent refills the bowl.
- The parent bird thread waits for `emptyBowl` before refilling the bowl and then signals all waiting baby birds using `fullBowl`.

Results

The program correctly synchronizes the birds' actions, preventing race conditions and ensuring proper coordination. Sample output:

```
Bird 1 eats a worm, bowl has 9 worms left
Bird 2 eats a worm, bowl has 8 worms left
Bird 3 eats a worm, bowl has 7 worms left
Bird 1 eats a worm, bowl has 6 worms left
Bird 3 eats a worm, bowl has 5 worms left
Bird 2 eats a worm, bowl has 4 worms left
Bird 1 eats a worm, bowl has 3 worms left
Bird 1 eats a worm, bowl has 2 worms left
Bird 2 eats a worm, bowl has 1 worms left
Bird 3 eats a worm, bowl has 0 worms left
Bird 2 finds an empty bowl, chirps to refill
Parent bird refills the bowl with 10 worms
Bowl refilled with worms
```

Task 2: The Bear and Honeybees Problem(multiple producers - one consumer)

Implementation

This problem involves a number of honeybees (producers) that add honey to a shared pot and a bear (consumer) that eats all the honey once the pot is full. The bear sleeps as it waits for the pot to be filled and wakes up to eat the honey.

- Honeybees add honey one at a time, ensuring mutual exclusion using `mutex`.
- When the pot is overflows, the last bee that fills it informs the bear with.`fullPot`.
- The bear wakes up , eats honey and signals the pot is empty to the bees using `emptyPot`.

Results

The program successfully models the behavior. Sample output:

```
honeybee 1 added honey,pot now has 1 portions.
honeybee 1 added honey,pot now has 2 portions.
honeybee 2 added honey,pot now has 3 portions.
honeybee 2 added honey,pot now has 4 portions.
pot reached its capacity, honeybee 2 is waking up the bear!
bear wakes up! bear eats all the honey,pot is now empty.
bear goes back to sleep.
honeybee 3 added honey,pot now has 1 portions.
honeybee 4 added honey,pot now has 2 portions.
honeybee 5 added honey,pot now has 3 portions.
honeybee 3 added honey,pot now has 4 portions.
pot reached its capacity, honeybee 3 is waking up the bear!
bear wakes up! bear eats all the honey,pot is now empty.
bear goes back to sleep.
```