الجامعة اللبنانية
كلية التكنولوجيا

CCNE Department
Lebanese University

Real-Time Programming

Lecture 6- Deadlock and models of synchronization

Rola El Osta

Dining philosophers problem

- In computer science, the **dining philosophers problem** is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them.
- It was originally formulated in 1965 by Edsger Dijkstra as a student exam exercise, presented in terms of computers competing for access to tape drive peripherals. Soon after, Tony Hoare gave the problem its present formulation[.

Problem statement

- Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers.
- Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After an individual philosopher finishes eating, they need to put down both forks so that the forks become available to others. A philosopher can take the fork on their right or the one on their left as they become available, but cannot start eating before getting both forks.
- Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed.
- The problem is how to design a discipline of behavior (a concurrent algorithm) such that no philosopher will starve; *i.e.*, each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think.

Conditions of deadlock

The Dining Philosophers Problem illustrates how careless use of synchronization primitives can lead to *Deadlock*. Specifically, recall that the following four conditions are necessary for deadlock to occur:

- Mutual exclusion: Only one thread can hold a single resource.
- No preemption: Once a thread acquires a resource, no other thread can steal it.
- Hold and wait: Threads can hold one resource while trying to acquire another.
- Circular wait: Multiple threads are simultaneously waiting on resources held by each other.

The Dining Philosophers Problem

The Dining Philosophers Problem illustrates deadlock by envisioning a table of philosphers sitting at a circular table under the following conditions:

- Each philosopher rotates through three states: thinking, hungry, and eating.
- At the center of the table is a bowl of spaghetti. When a philosopher gets hungry, they serve themselves from the bowl and eat.
- Between each pair of philosophers is a single serving fork. Each philosopher can serve themselves by grabbing the fork to their left and the fork to their right, then scoop food onto their plate.
- Once a philosopher grabs one fork, they hold it until they can grab the other and eat.

Possible solutions

1. To prevent it: to impose a constraint so that one of 4 circumstances does not appear.
2. To Control it: Add controls to lock a resource only if there is no risk of deadlock.
3. To heal it: finish one of the tasks to unlock the others.

Deny one of the circumstances:

1. Mutual exclusion: difficult to do without mutex.
2. hold and wait:

    - Acquire all resources at the same time: atomic section, the acquisition phase is itself a critical section.

    - Release an acquired resource(trylock → risk of livelock: nobody locks anything).

3. No preemption: The resources obtained by a process are hardly removed, it ends or go back (difficult, tasks not necessarily removed, rollback → requires to memorize the previous state for a backtrack).
4. Circular wait: An order is placed on resources. All entities must request all resources always in the same order so as never to "close" the cycle.

Some solution attempts

- Deny circumstance 2: Use a mutex to protect the acquisition of the forks. Disadvantage: we slowed down everybody ...
- Deny circumstance 2: use timedlock or trylock. All philosophers are likely to wait and nobody eat.
- Deny circumstance 4: introduce a left-hand person. One of the philosophers will take the left fork before the right. In the case of an even number of philosophers, we let one of two to be left-hand: N / 2 philosophers eat at the same time.
- Deny circumstance 4: add a semaphore that allows only N-1 philosophers to eat at the same time. One of the philosopher leaves the forks to his neighbors. This semaphore makes sure that everyone will eat in his turn.
- Deny circumstance 4: Chandy / Misra's solution (1984) Introduce a concept of scheduling and request. Philosophers politely ask for forks. If two philosophers want to acquire the same fork, we give it to the one with the smallest ID. If one of the philosophers has the fork, he gives it to his neighbor if he has finished eating. If he has not eaten yet, he waits for the second fork and eats a little before pass it to him.

Go to Practice exercises!