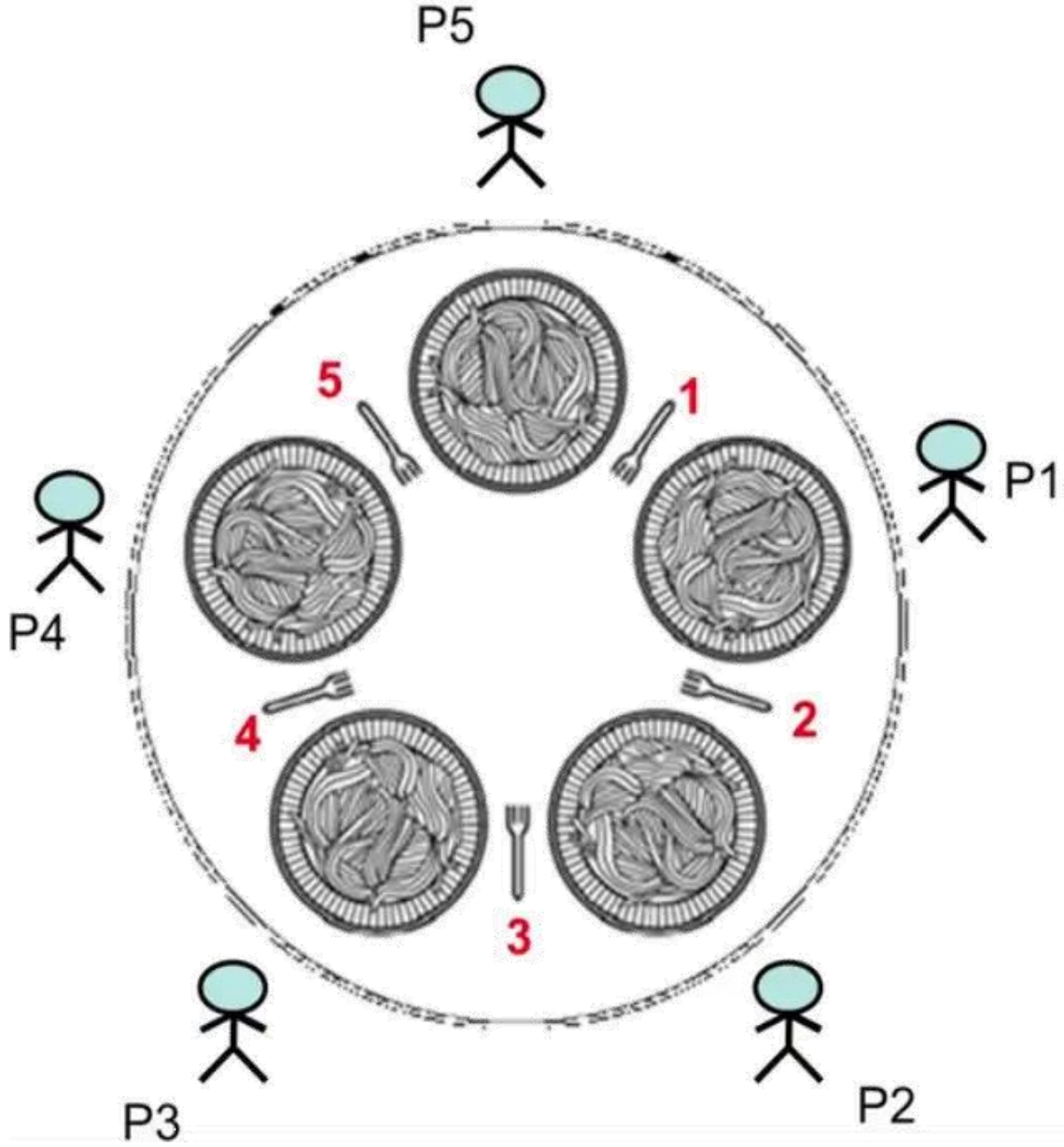
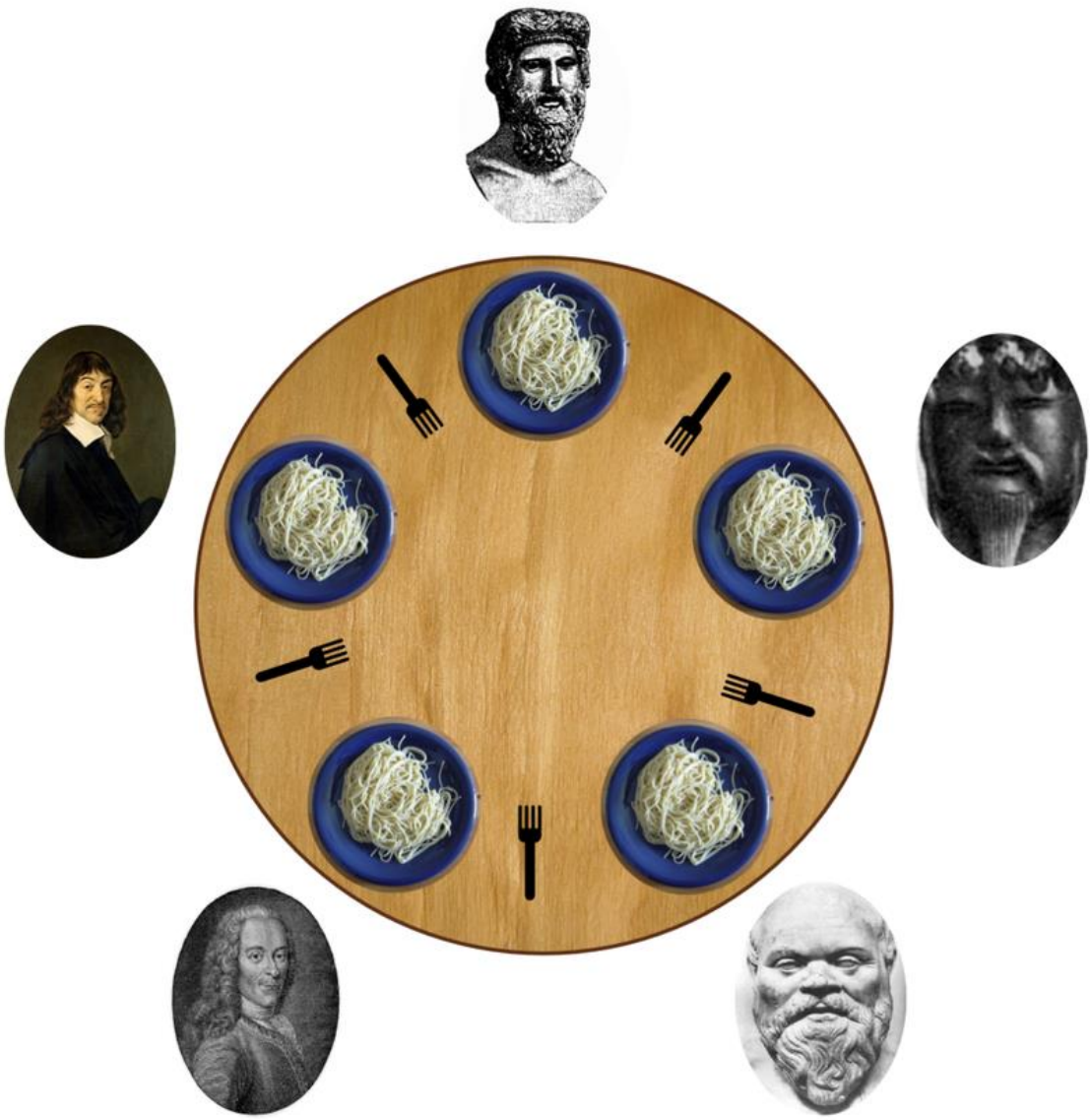
The Dining Philosophers


P5
P1
P4
P3
P2
1
2
3
4
5

Introduction

One of the best classic examples which is used to describe ***synchronization*** issues in a multi-threaded background is the 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.



Problem statement

Five philosophers dine together at the same table. Each philosopher has their own place at the table. There is a fork between each plate. The dish served is a kind of spaghetti which has to be eaten with two forks. Each philosopher can only alternately think and eat. Moreover, a philosopher can only eat their spaghetti when they have both a left and right fork. Thus two forks will only be available when their two nearest neighbors are thinking, not eating. After an individual philosopher finishes eating, they will put down both forks. The problem is how to design a regimen (a concurrent algorithm) such that no philosopher will starve; *i.e.*, each can forever continue to alternate between eating and thinking, assuming that philosopher can know when others may want to eat or think

Example Of Deadlock:

IF Each Philosopher takes the left chopstick, *then* The Philosopher try to take the right Chopstick that will lead to deadlock And We Lost The Progress

Solution For Deadlock:

We solve the deadlock by make the process of take the chopstick, Synchronous Operation So I make Sure, that the Philosopher Will Take The right chopstick and left chopstick At same time

Example Of Starvation:

We solve the deadlock Problem, but we still have a Problem what about one of the philosophers doesn't have a chance to eat So that will lead to starvation

Solution Of Starvation:

We make finite time for each philosopher to eat then the next philosopher eats and so on

Problems

The problem was designed to illustrate the challenges of avoiding deadlock, a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows:

1- **think until the left fork is available; when it is, pick it up ;**
2- **think until the right fork is available; when it is, pick it up ;**
3- **when both forks are held, eat for a fixed amount of time ;**
4- **put the left fork down ;**
5- **put the right fork down ;**
6- **repeat from the beginning.**

pseudo code

```
while(true)

{

think(); //Initially thinking about the whole
universal things.

pick_up_left_fork(); //Readying to eat as
philosopher gets hungry eventually.

pick_up_right_fork();

eat();

put_down_right_fork();

put_down_left_fork();

think(); //Back to start thinking as hungry is over.

}
```

Real World Application

(Hotel Reservation):

We have Website to book rooms in hotel and there is only 1 room available

And 2 Clients Try to book the room

Without Synchronization, it might happen that both might end up booking up room, and **We only have only 1 room, which is, of course, going to create a problem**

What is The Solution????

The solution is to solve this problem by using Synchronization Method

Let's Go to Code