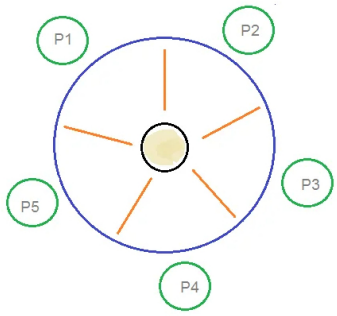
Dining Philosophers Problem

The dining philosophers problem is another classic synchronization problem which is used to evaluate situations where there is a need of allocating multiple resources to multiple processes.

What is the Problem Statement?

Consider there are five philosophers sitting around a circular dining table. The dining table has five chopsticks and a bowl of rice in the middle as shown in the below figure.



Dining Philosophers Problem

At any instant, a philosopher is either eating or thinking. When a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.

Here's the Solution

From the problem statement, it is clear that a philosopher can think for an indefinite amount of time. But when a philosopher starts eating, he has to stop at some point of time. The philosopher is in an endless cycle of thinking and eating.

An array of five semaphores, `stick[5]`, for each of the five chopsticks.

The code for each philosopher looks like:

```
while(TRUE)
{
    wait(stick[i]);
    /*
        mod is used because if i=5, next
        chopstick is 1 (dining table is circular)
    */
    wait(stick[(i+1) % 5]);

    /* eat */
    signal(stick[i]);

    signal(stick[(i+1) % 5]);
    /* think */
}
```

When a philosopher wants to eat the rice, he will wait for the chopstick at his left and picks up that chopstick. Then he waits for the right chopstick to be available, and then picks it too. After eating, he puts both the chopsticks down.

But if all five philosophers are hungry simultaneously, and each of them pickup one chopstick, then a deadlock situation occurs because they will be waiting for another chopstick forever. The possible solutions for this are:

- A philosopher must be allowed to pick up the chopsticks only if both the left and right chopsticks are available.
- Allow only four philosophers to sit at the table. That way, if all the four philosophers pick up four chopsticks, there will be one chopstick left on the table. So, one philosopher can start eating and eventually, two chopsticks will be available. In this way, deadlocks can be avoided.

← Prev Next →

Learn Coding (for beginners)
Tutorial Library
Interview Tests
Curious
Practice Coding
Educators Program

Coding Courses
Learn HTML
Learn CSS

Resources
C Language
C++/STL
Java
DBMS
Python
PHP
Android
Game Development
Data Structure & Alog.
Operating System
Computer Network
Computer Architecture
More...

Interview Tests
Java Interview Tests
Python Interview Tests
DBMS Interview Tests
Linux Interview Tests
Aptitude Tests
GATE 2022 Tests
More...