# 哲学家就餐问题

**哲学家就餐问题**（英语：Dining philosophers problem）是在计算机科学中的一个经典问题，用来演示在并发计算中多线程同步（Synchronization）时产生的问题。

在1971年，著名的计算机科学家艾兹格·迪科斯彻提出了一个同步问题，即假设有五台计算机都试图访问五份共享的磁带驱动器。稍后，这个问题被托尼·霍尔重新表述为哲学家就餐问题。这个问题可以用来解释死锁和资源耗尽。

## 问题描述

哲学家就餐问题可以这样表述，假设有五位哲学家围坐在一张圆形餐桌旁，做以下两件事情之一：吃饭，或者思考。吃东西的时候，他们就停止思考，思考的时候也停止吃东西。餐桌中间有一大碗意大利面，每位哲学家之间各有一只餐叉。因为用一只餐叉很难吃到意大利面，所以假设哲学家必须用两只餐叉吃东西。他们只能使用自己左右手边的那两只餐叉。哲学家就餐问题有时也用米饭和五根筷子而不是意大利面和餐叉来描述，因为吃米饭必须用两根筷子。

哲学家就餐问题的演示

哲学家从来不交谈，这就很危险，可能产生死锁，每个哲学家都拿着左手的餐叉，永远都在等右边的餐叉（或者相反）。

即使没有死锁，也有可能发生资源耗尽。例如，假设规定当哲学家等待另一只餐叉超过五分钟后就放下自己手里的那一只餐叉，并且再等五分钟后进行下一次尝试。这个策略消除了死锁（系统总会进入到下一个状态），但仍然有可能发生“活锁”。如果五位哲学家在完全相同的时刻进入餐厅，并同时拿起左边的餐叉，那么这些哲学家就会等待五分钟，同时放下手中的餐叉，再等五分钟，又同时拿起这些餐叉。

在实际的计算机问题中，缺乏餐叉可以类比为缺乏共享资源。一种常用的计算机技术是资源加锁，用来保证在某个时刻，资源只能被一个程序或一段代码访问。当一个程序想要使用的资源已经被另一个程序锁定，它就等待资源解锁。当多个程序涉及到加锁的资源时，在某些情况下就有可能发生死锁。例如，某个程序需要访问两个文件，当两个这样的程序各锁了一个文件，那它们都在等待对方解锁另一个文件，而解锁永远不会发生。

# 解法

## 服务生解法

一个简单的解法是引入一个餐厅服务生，哲学家必须经过他的允许才能拿起餐叉。因为服务生知道哪只餐叉正在使用，所以他能够作出判断避免死锁。

为了演示这种解法，假设哲学家依次标号为A至E。如果A和C在吃东西，则有四只餐叉在使用中。B坐在A和C之间，所以两只餐叉都无法使用，而D和E之间有一只空余的餐叉。假设这时D想要吃东西。如果他拿起了第五只餐叉，就有可能发生死锁。相反，如果他征求服务生同意，服务生会让他等待。这样，我们就能保证下次当两把餐叉空余出来时，一定有一位哲学家可以成功的得到一对餐叉，从而避免了死锁。

## 资源分级解法

另一个简单的解法是为资源（这里是餐叉）分配一个偏序或者分级的关系，并约定所有资源都按照这种顺序获取，按相反顺序释放，而且保证不会有两个无关资源同时被同一项工作所需要。在哲学家就餐问题中，资源（餐叉）按照某种规则编号为1至5，每一个工作单元（哲学家）总是先拿起左右两边编号较低的餐叉，再拿编号较高的。用完餐叉后，他总是先放下编号较高的餐叉，再放下编号较低的。在这种情况下，当四位哲学家同时拿起他们手边编号较低的餐叉时，只有编号最高的餐叉留在桌上，从而第五位哲学家就不能使用任何一只餐叉了。而且，只有一位哲学家能使用最高编号的餐叉，所以他能使用两只餐叉用餐。当他吃完后，他会先放下编号最高的餐叉，再放下编号较低的餐叉，从而让另一位哲学家拿起后边的这只开始吃东西。

尽管资源分级能避免死锁，但这种策略并不总是实用的，特别是当所需资源的列表并不是事先知道的时候。例如，假设一个工作单元拿着资源3和5，并决定需要资源2，则必须先要释放5，之后释放3，才能得到2，之后必须重新按顺序获取3和5。对需要访问大量数据库记录的计算机程序来说，如果需要先释放高编号的记录才能访问新的记录，那么运行效率就不会高，因此这种方法在这里并不实用。

## Chandy/Misra解法

1984年，K. Mani Chandy和J. Misra提出了哲学家就餐问题的另一个解法，允许任意的用户（编号 $P_1, \cdots, P_n$）争用任意数量的资源。与资源分级解法不同的是，这里编号可以是任意的。

- 把餐叉凑成对，让要吃的人先吃，没餐叉的人得到一张换餐叉券。
- 饿了，把换餐叉券交给有餐叉的人，有餐叉的人吃饱了会把餐叉交给有券的人。有了券的人不会再得到第二张券。
- 保证有餐叉的都有得吃。

这个解法允许很大的并行性，适用于任意大的问题。

## 延伸阅读


- Silberschatz, Abraham; Peterson, James L. Operating Systems Concepts. Addison-Wesley. 1988. ISBN 0-201-18760-4.
- Chandy, K.M.; Misra, J.（1984）. The Drinking Philosophers Problem . ACM Transactions on Programming Languages and Systems.
- Dijkstra, E. W.（1971, June）. Hierarchical ordering of sequential processes . Acta Informatica 1 (2): 115-138.
- Lehmann, D. J., Rabin M. O,（1981）. On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem. Principles Of Programming Languages 1981（POPL'81）, pages 133-138.