章节导学

线程同步

进程同步

既是重要的理论知识，也是重要的实践知识

笔试面试、开发高性能服务等都非常有用

生产者-消费者问题

有一群生产者进程在生产产品，并将这些产品提供给消费者进程进行消费，生产者进程和消费者进程可以并发执行，在两者之间设置了一个具有n可缓冲区的缓冲池，生产者进程需要将所生产的产品放到一个缓冲区中，消费者进程可以从缓冲区取走产品消费。

哲学家进餐问题

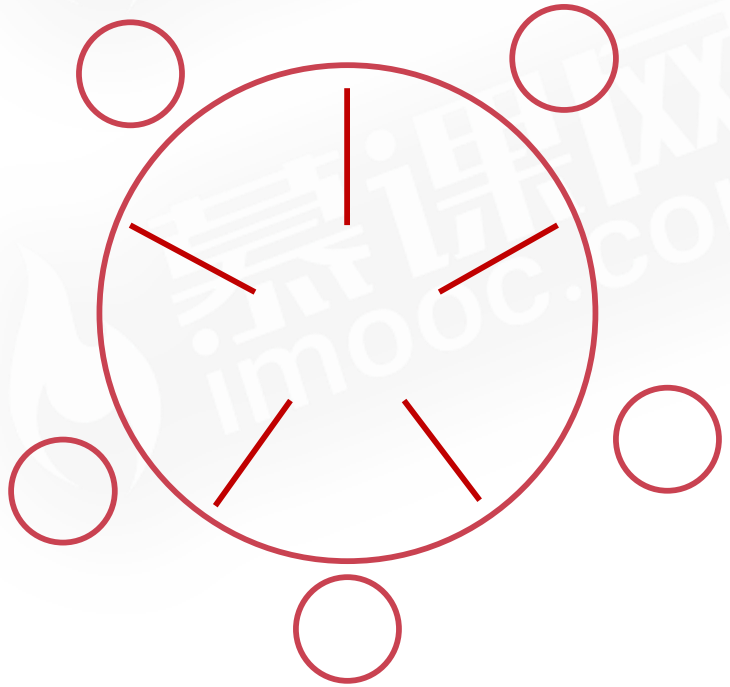
有五个哲学家，他们的生活方式是交替地进行思考和进餐，哲学家们共同使用一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五支筷子。平时哲学家们只进行思考，饥饿时则试图取靠近他们的左、右两支筷子，只有两支筷子都被他拿到的时候就能进餐，进餐完毕之后，放下左右筷子继续思考。

拿起左边筷子

拿起右边筷子

进餐

哲学家进餐问题



哲学家进餐问题

拿起左边筷子

发现右边筷子被拿了

等待右边筷子释放

拿起右边筷子

进餐

五个哲学家同时拿起左边筷子

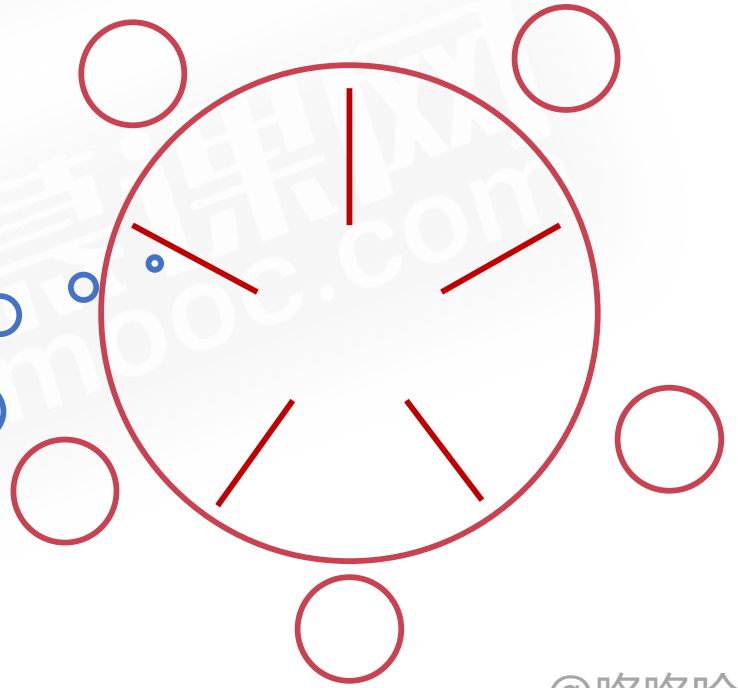
发现右边筷子被拿了

五个哲学家都等待右边筷子释放

相互等待

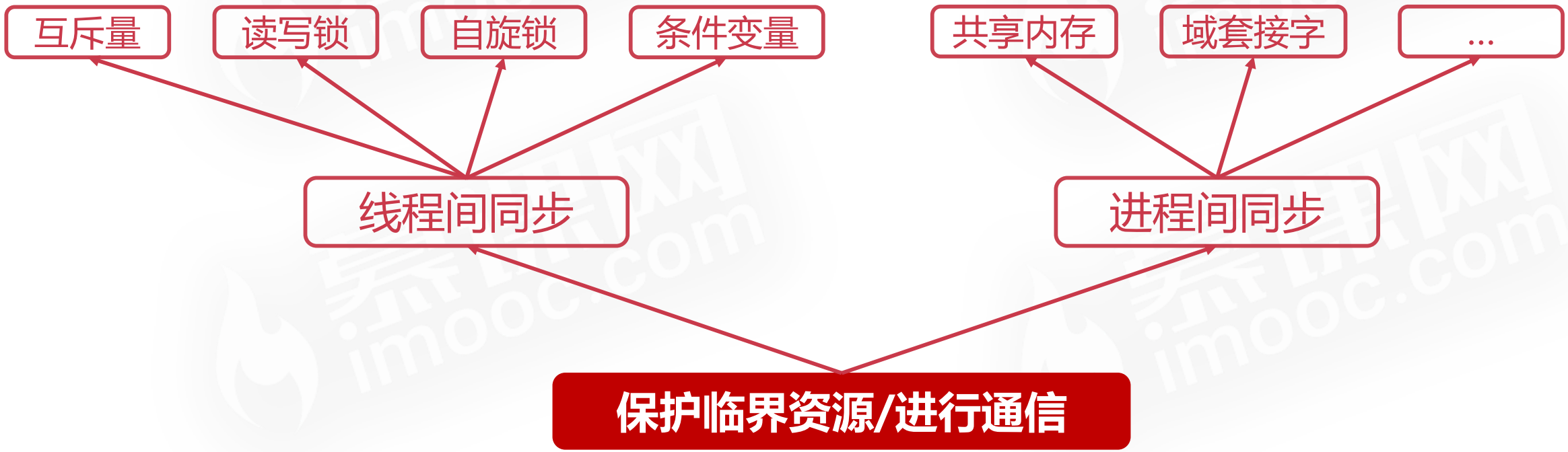
五个哲学家饿死

临界资源



哲学家进餐问题

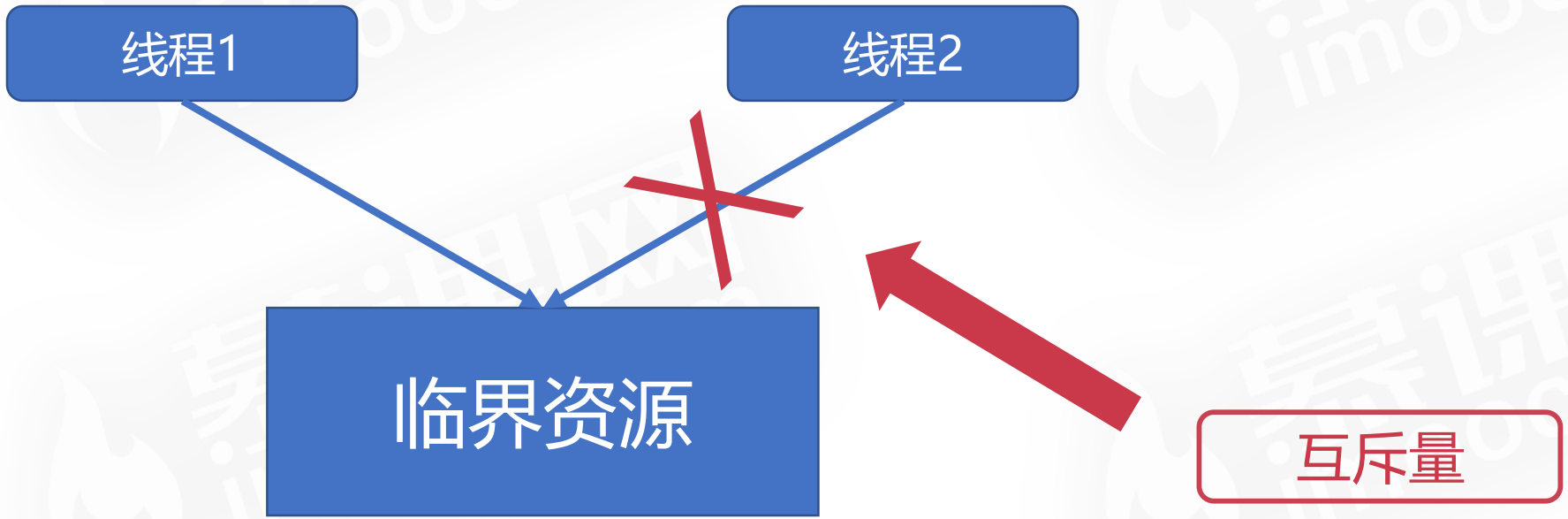
临界资源



重要概念

- 用户态与内核态
- 上下文切换
- 协程
- 编写性能良好的程序指南

线程同步之互斥量



线程同步之互斥量

10

register=count

register=register+1

register=count

register=register-1

count=register

count=register

11

- 两个线程的指令交叉执行
- 互斥量可以保证先后执行

原子性

register=count

register=register+1

count=register

register=count

register=register-1

count=register

线程同步之互斥量

- 原子性是指一系列操作不可被中断的特性
- 这一系列操作要么全部执行完成，要么全部没有执行
- 不存在部分执行部分未执行的情况

原子性

register=count

register=register+1

count=register

线程同步之互斥量

- 互斥量是最简单的线程同步的方法
- 互斥量（互斥锁），处于两态之一的变量：解锁和加锁
- 两个状态可以保证资源访问的串行

线程同步之互斥量

- 操作系统直接提供了互斥量的API
- 开发者可以直接使用API完成资源的加锁、解锁操作

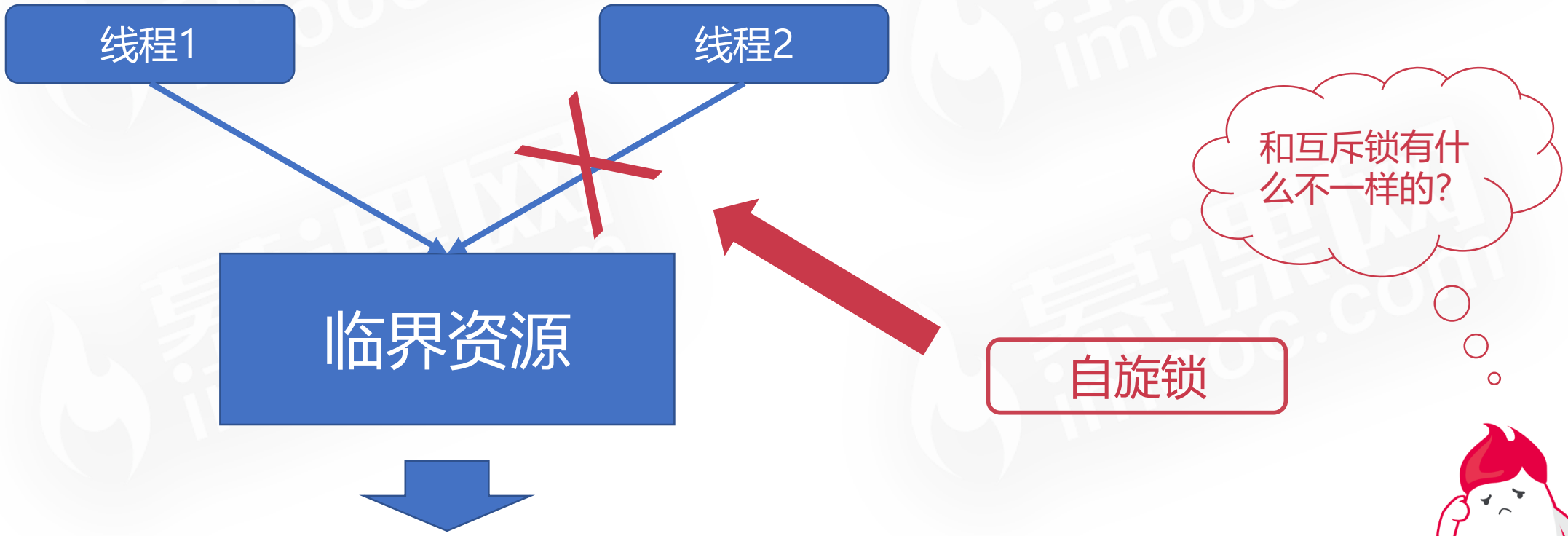
线程同步之互斥量

- pthread_mutex_t
- pthread_mutex_lock
- pthread_mutex_unlock

线程同步之互斥量

互斥锁的例子→

线程同步之自旋锁



线程同步之自旋锁

- 自旋锁也是一种多线程同步的变量
- 使用自旋锁的线程会反复检查锁变量是否可用
- 自旋锁不会让出CPU，是一种忙等待状态

死循环等待锁被释放

线程同步之自旋锁

- 自旋锁避免了进程或线程上下文切换的开销
- 操作系统内部很多地方使用的是自旋锁
- 自旋锁不适合在单核CPU使用

线程同步之自旋锁

- pthread_spinlock_t
- pthread_ spinlock _lock
- pthread_ spinlock _unlock

自旋锁的例子→

线程同步之自旋锁

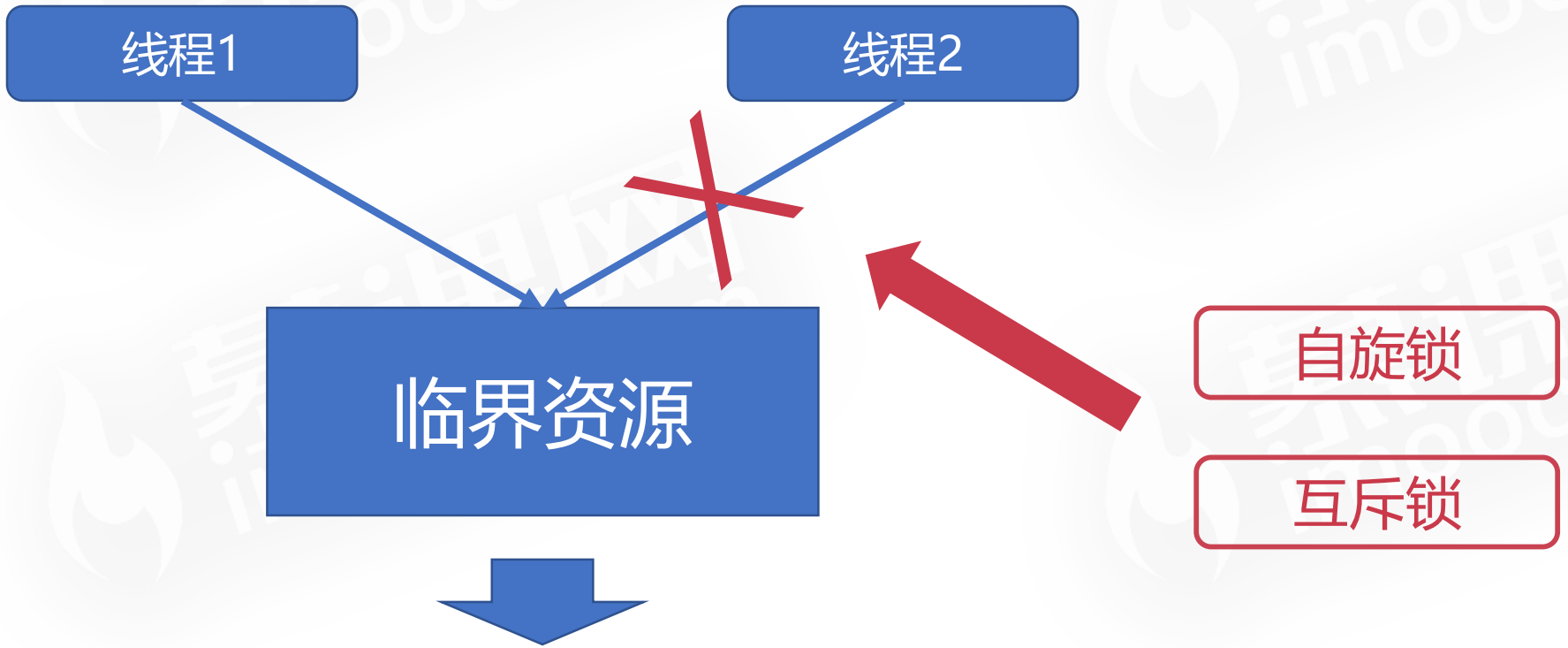
- 自旋锁也是一种多线程同步的变量
- 使用自旋锁的线程会反复检查锁变量是否可用
- 自旋锁不会让出CPU，是一种忙等待状态

死循环等待锁被释放

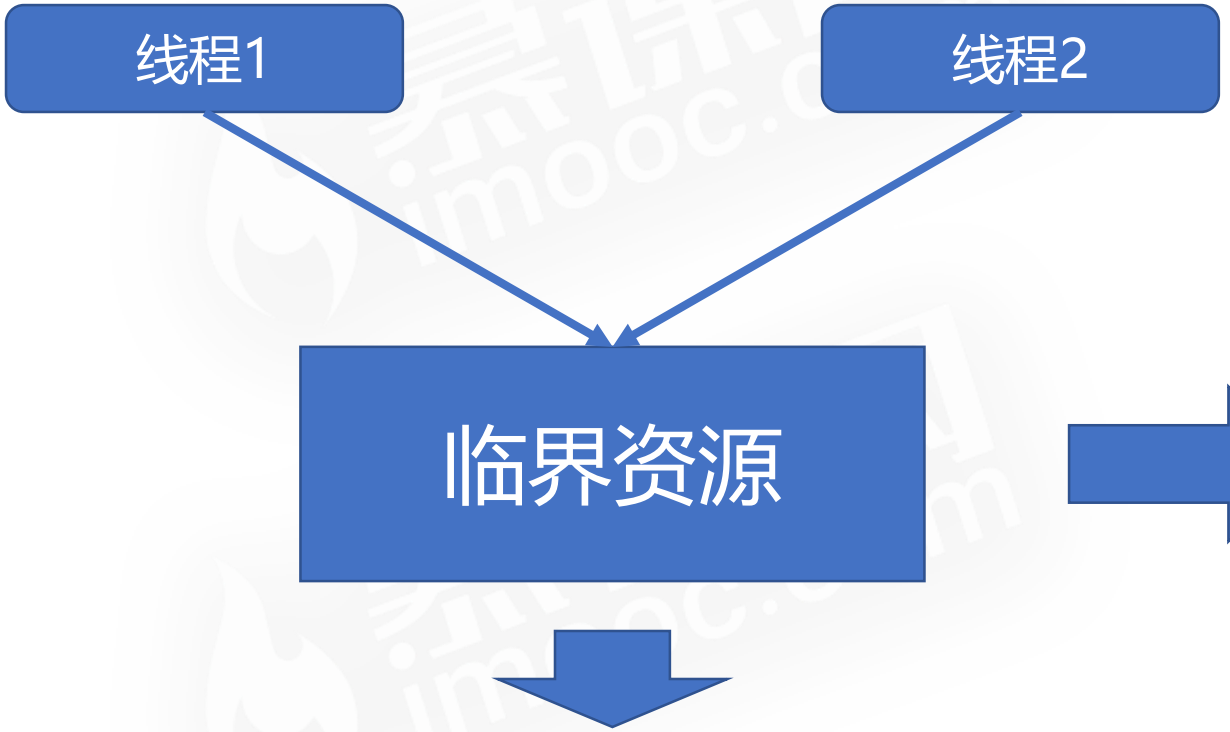
线程同步之自旋锁

- 自旋锁避免了进程或线程上下文切换的开销
- 操作系统内部很多地方使用的是自旋锁
- 自旋锁不适合在单核CPU使用

线程同步之读写锁



线程同步之读写锁



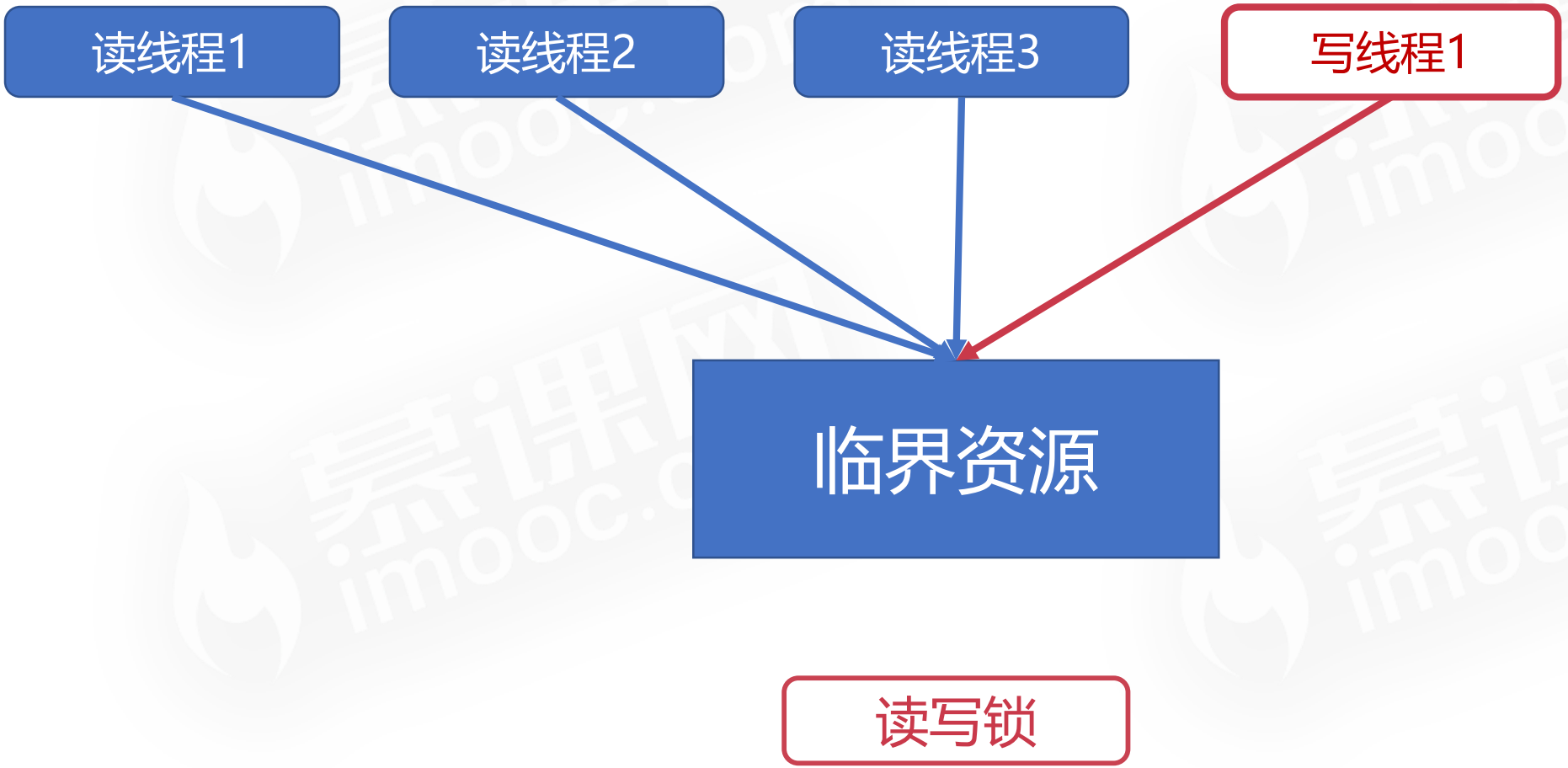
- 临界资源多读少写
- 读取的时候并不会改变临界资源的值
- 是否存在效率更高的同步方法?

读写锁

线程同步之读写锁

- 读写锁是一种特殊的自旋锁
- 允许多个读者同时访问资源以提高读性能
- 对于写操作则是互斥的

线程同步之读写锁



线程同步之读写锁

- pthread_rwlock_t
- pthread_rwlock_rdlock（读锁）
- pthread_rwlock_wrlock（写锁）

读写锁的例子→

线程同步之读写锁

- 读写锁是一种特殊的自旋锁
- 允许多个读者同时访问资源以提高读性能
- 对于写操作则是互斥的

线程同步之条件变量

- 条件变量是一种相对复杂的线程同步方法
- 条件变量允许线程睡眠，直到满足某种条件
- 当满足条件时，可以向该线程信号，通知唤醒

线程同步之条件变量

- 缓冲区小于等于0时，不允许消费者消费，消费者必须等待
- 缓冲区满时，不允许生产者往缓冲区生产，生产者必须等待

当生产者生产一个产品时，唤醒可能等待的消费者

当消费者消费一个产品时，唤醒可能等待的生产者

线程同步之条件变量

- pthread_cond_t

配合互斥量使用

- pthread_cond_wait(等待条件满足)
- pthread_cond_signal(等待被唤醒)

条件变量的例子→

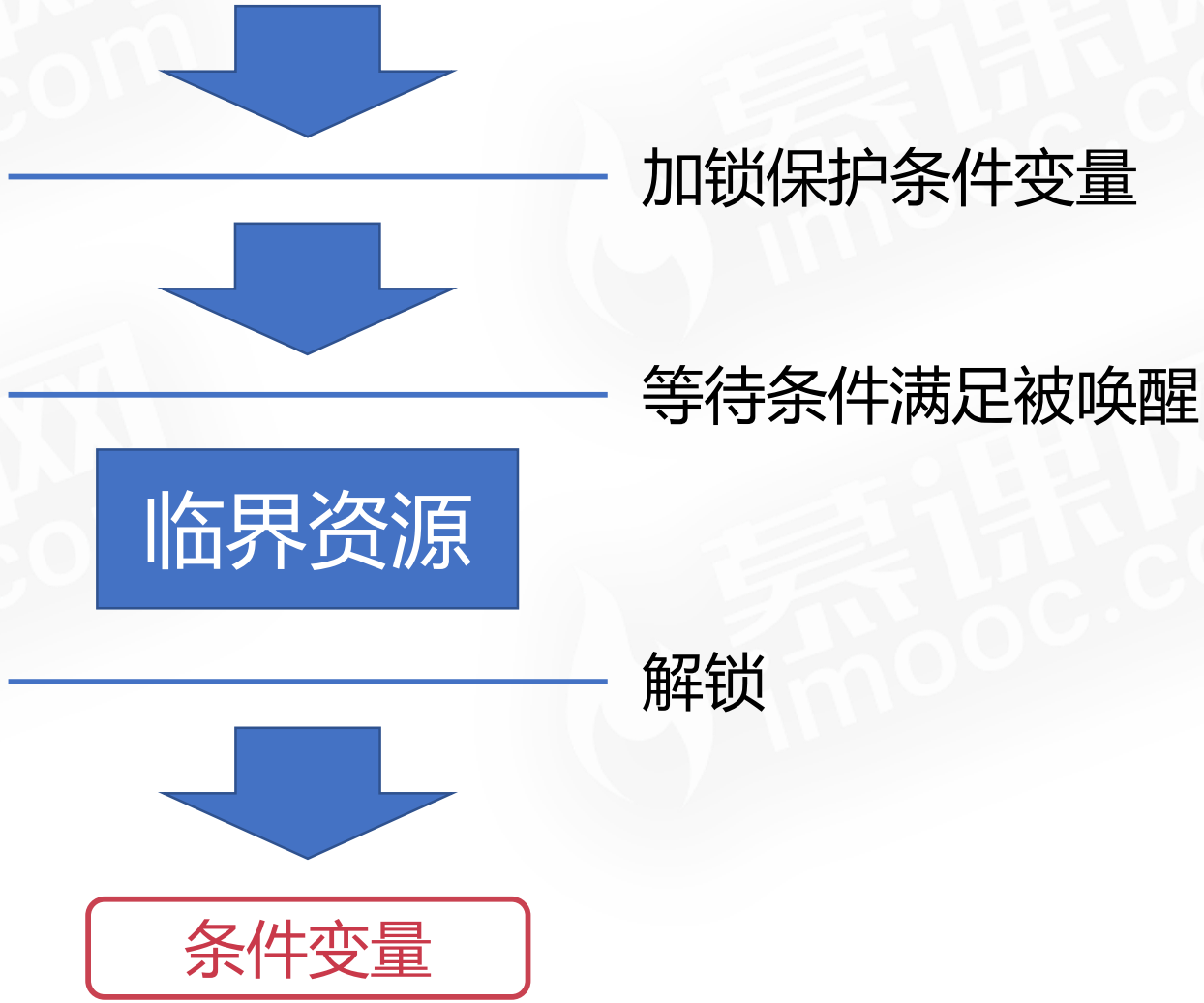
线程同步之条件变量

- 条件变量是一种相对复杂的线程同步方法
- 条件变量允许线程睡眠，直到满足某种条件
- 当满足条件时，可以向该线程信号，通知唤醒

线程同步方法总结

互斥量、自旋锁、读写锁

线程同步方法总结



线程同步方法对比

同步方法	描述
互斥锁	最简单的一种线程同步方法，会阻塞线程
自旋锁	避免切换的一种线程同步方法，属于“忙等待”
读写锁	为“读多写少”的资源设计的线程同步方法，可以显著提高性能
条件变量	相对复杂的一种线程同步方法，有更灵活的使用场景

使用fork系统调用创建进程

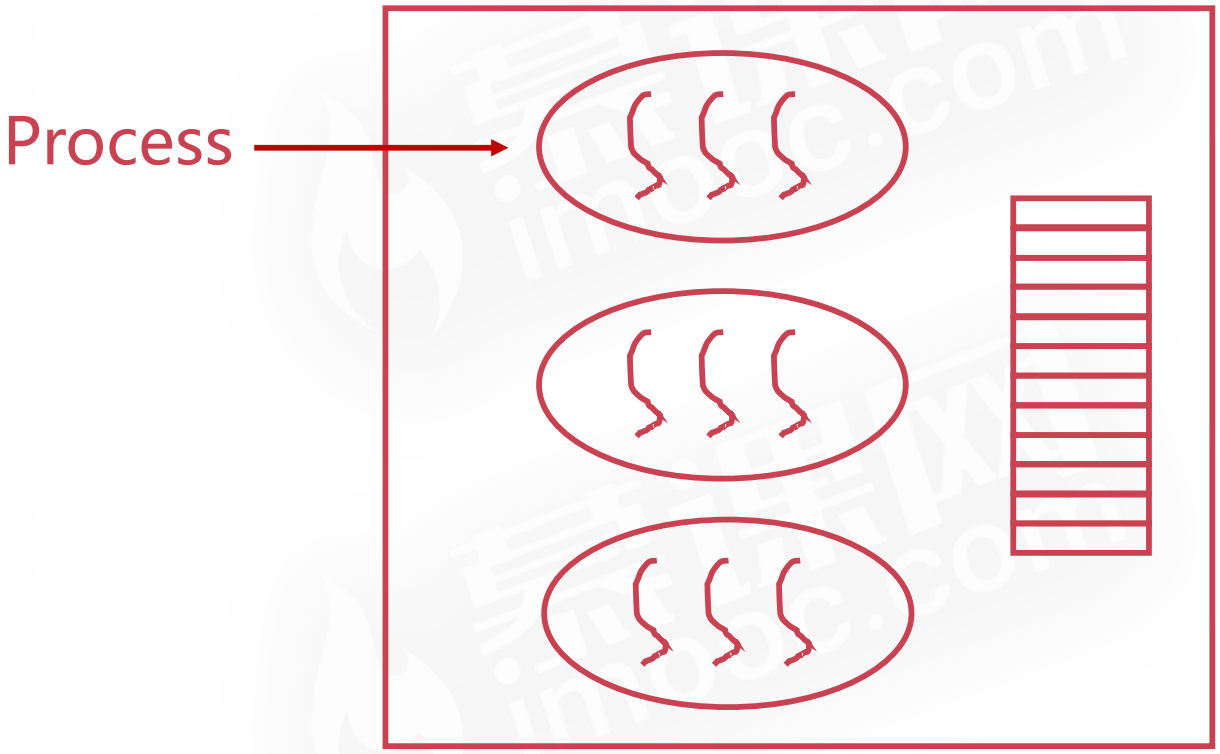
- fork系统调用是用于创建进程的
- fork创建的进程初始化状态与父进程一样
- 系统会为fork的进程分配新的资源

使用fork系统调用创建进程

- fork系统调用无参数
- fork会返回两次，分别返回子进程id和0
- 返回子进程id的是父进程，返回0的是子进程

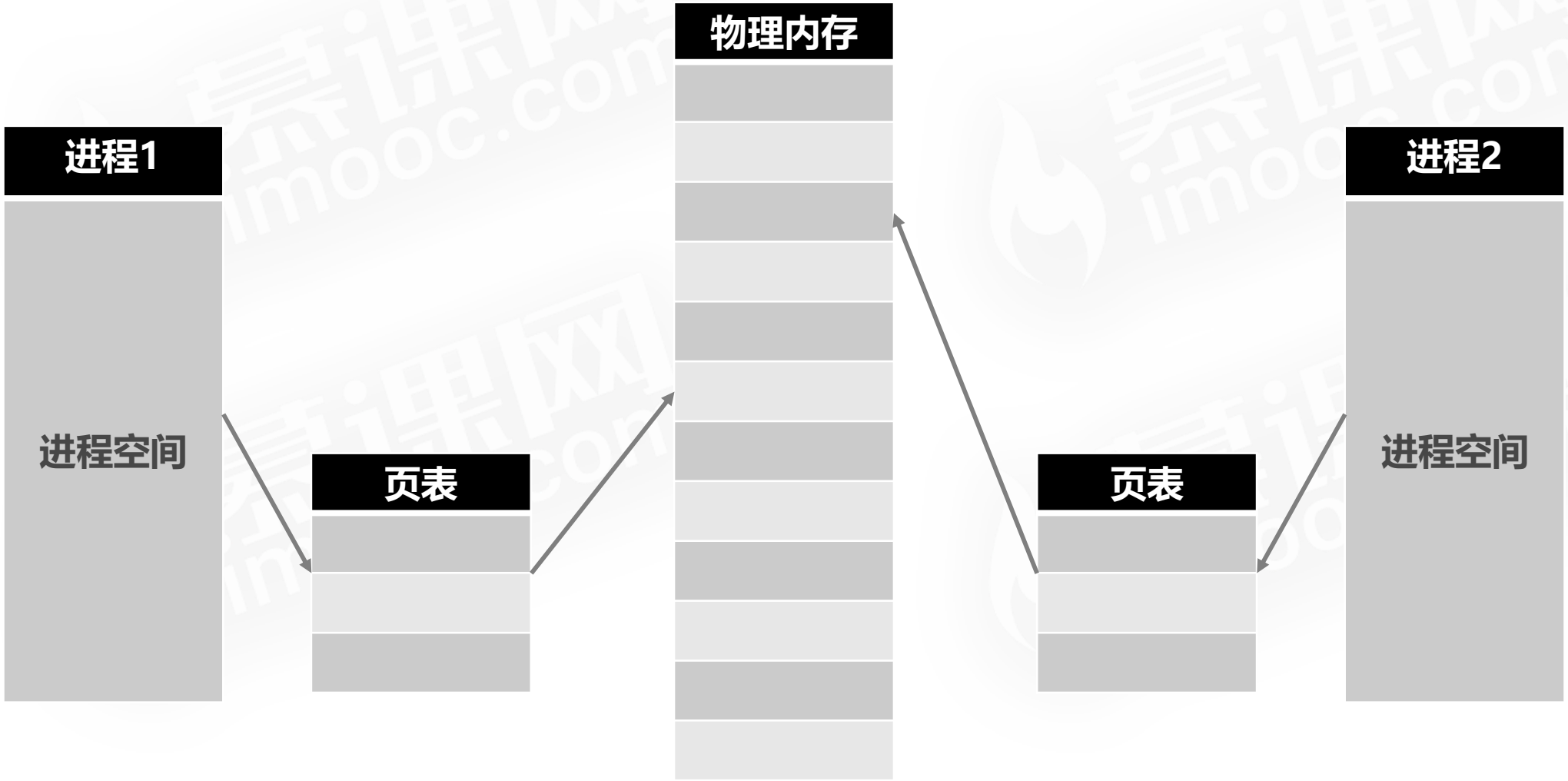
进程同步之共享内存

进程的线程共享进程资源



进程共享计算机资源

进程同步之共享内存



进程同步之共享内存

- 在某种程度上，多进程是共同使用物理内存的
- 由于操作系统的进程管理，进程间的内存空间是独立的

进程默认是不能访问进程空间之外的内存空间的

进程同步之共享内存

进程同步之共享内存

- 共享存储允许不相关的进程访问同一片物理内存
- 共享内存是两个进程之间共享和传递数据最快的方式
- 共享内存未提供同步机制，需要借助其他机制管理访问

进程同步之共享内存

进程同步之共享内存

共享内存的例子→

进程同步之共享内存

进程同步之共享内存

- 共享存储是两个进程之间共享和传递数据最快的一种方式
- 共享内存未提供同步机制，需要借助其他机制管理访问

共享内存是高性能后台开发中最常用的进程同步方式

进程同步之Unix域套接字

- 域套接字是一种高级的进程间通信的方法
- Unix域套接字可以用于同一机器进程间通信

进程同步之Unix域套接字

- 套接字(socket)原是网络通信中使用的术语
- Unix系统提供的域套接字提供了网络套接字类似的功能

Nginx　uWSGI　...

进程同步之Unix域套接字

进程同步之Unix域套接字

域套接字的例子→

进程同步之Unix域套接字

进程同步之Unix域套接字

- 提供了单机简单可靠的进程通信同步服务
- 只能在单机使用，不能跨机器使用

操作系统的用户态与内核态

理解上下文切换

进程、线程与协程