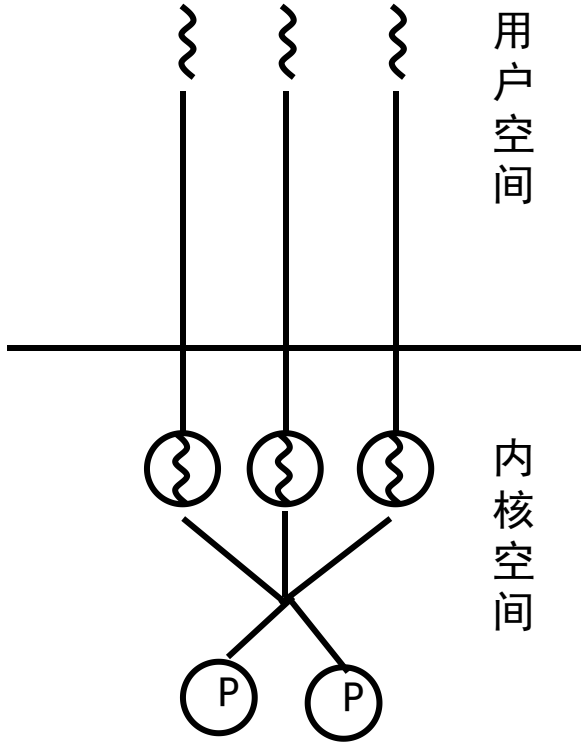
内核级线程KLT, Kernel-Level Threads

- 线程管理的所有工作由OS内核来做
- OS提供了一个应用程序设计接口API，供开发者使用KLT
- OS直接调度KLT

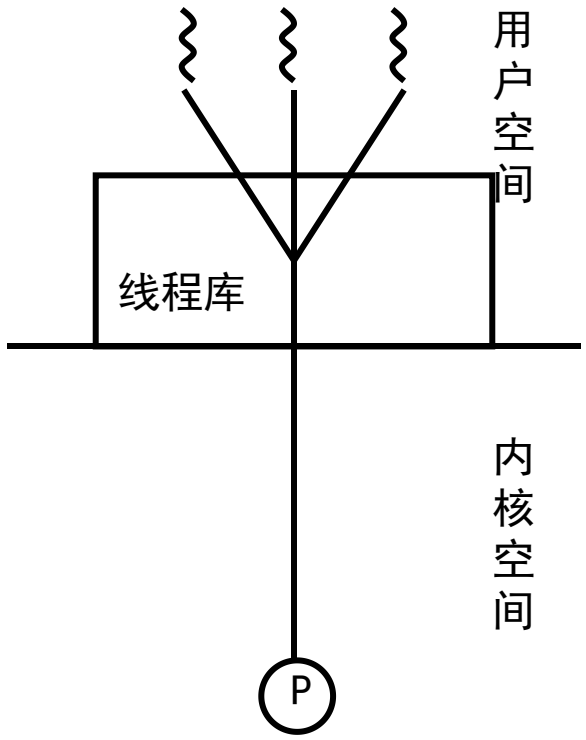


内核级线程的特点

- 进程中的一个线程被阻塞了，内核能调度同一进程的其它线程占有处理器运行
- 多处理器环境中，内核能同时调度同一进程中多个线程并行执行
- 内核自身也可用多线程技术实现，能提高操作系统的执行速度和效率
- 应用程序线程在用户态运行，线程调度和管理在内核实现，在同一进程中，控制权从一个线程传送到另一个线程时需要模式切换，系统开销较大

用户级线程ULT, User-Level Threads

- 用户空间运行的线程库，提供多线程应用程序的开发和运行支撑环境
- 任何应用程序均需通过线程库进行程序设计，再与线程库连接后运行
- 线程管理的所有工作都由应用程序完成，内核没有意识到线程的存在



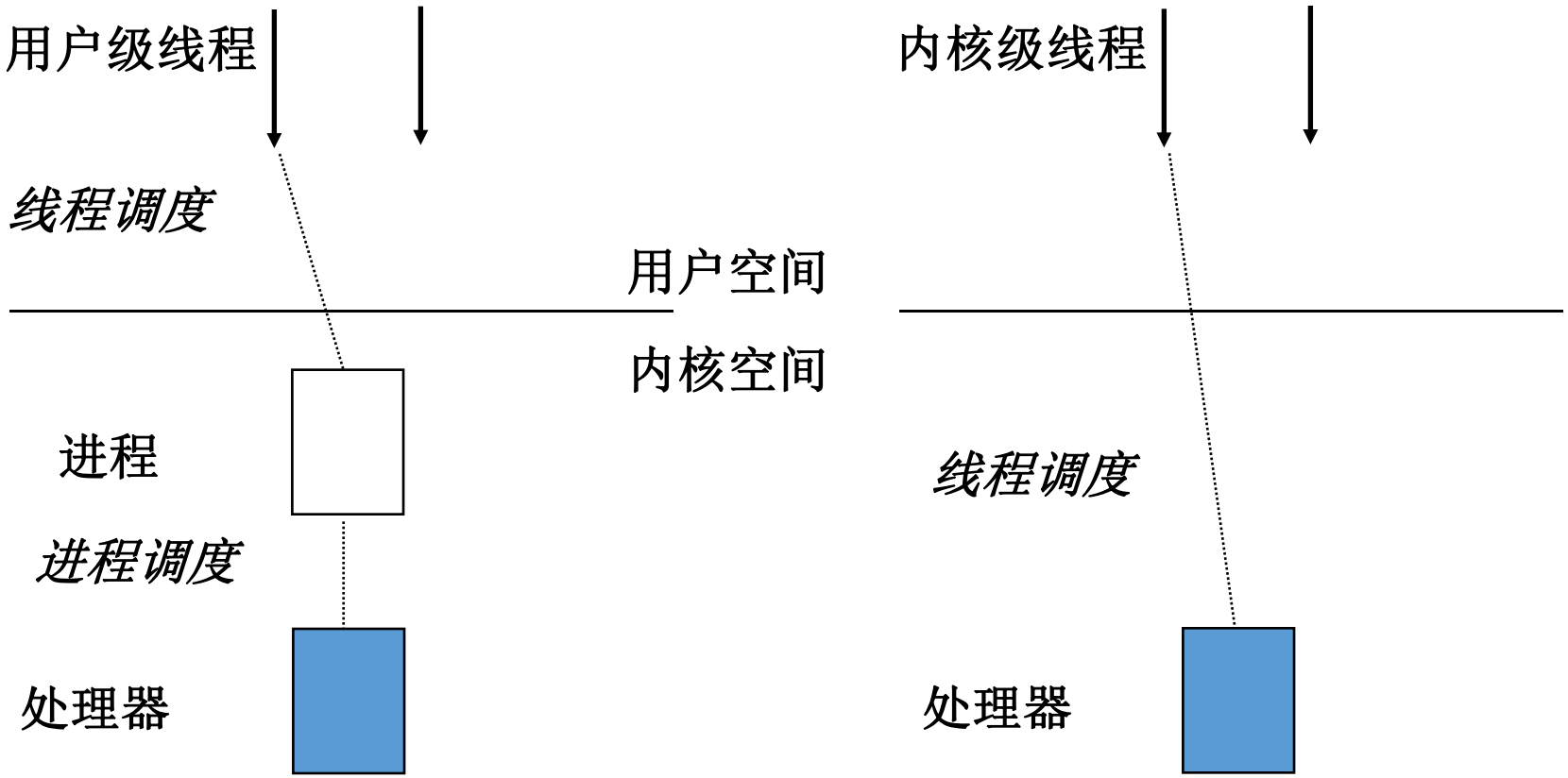
用户级线程的特点

- 所有线程管理数据结构均在进程的用户空间中，线程切换不需要内核模式，能节省模式切换开销和内核的宝贵资源
- 允许进程按应用特定需要选择调度算法，甚至根据应用需求裁剪调度算法
- 能运行在任何OS上，内核在支持ULT方面不需要做任何工作
- 不能利用多处理器的优点，OS调度进程，仅有一个ULT能执行
- 一个ULT的阻塞，将引起整个进程的阻塞

Jacketing技术

- 把阻塞式系统调用改造成非阻塞式的
- 当线程陷入系统调用时，执行jacketing程序
- 由jacketing 程序来检查资源使用情况，以决定是否执行进程切换或传递控制权给另一个线程

用户级线程 vs. 内核级线程



- ULT适用于解决逻辑并行性问题
- KLT适用于解决物理并行性问题