# 操作系统复习

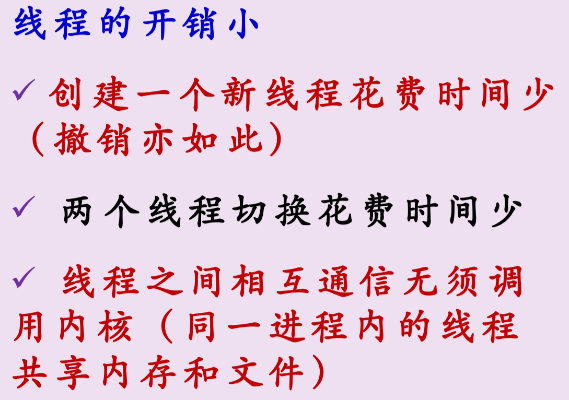
第四章： 线程

第四章总纲：

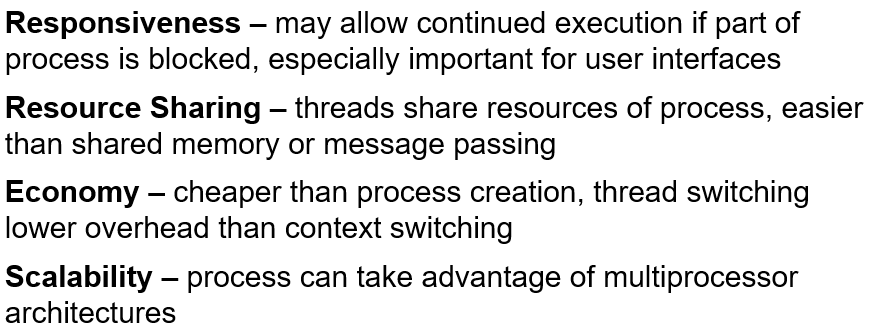
* Overview
* Multicore Programming
* Multithreading Models
* Thread Libraries
* Implicit Threading
* Threading Issues
* Operating System Examples

线程是CPU使用的基本单位，它由线程ID、程序计数器、寄存器集合和栈组成。

**#为什么要引入线程？**

1. 应用的需要。如web服务器

2. 开销的考虑。

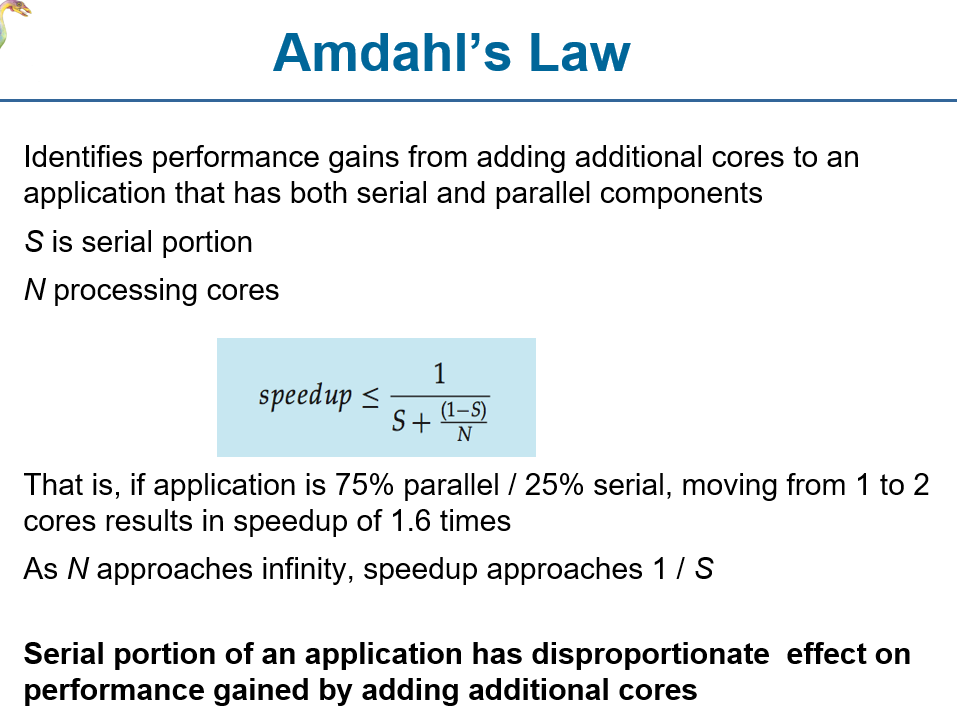
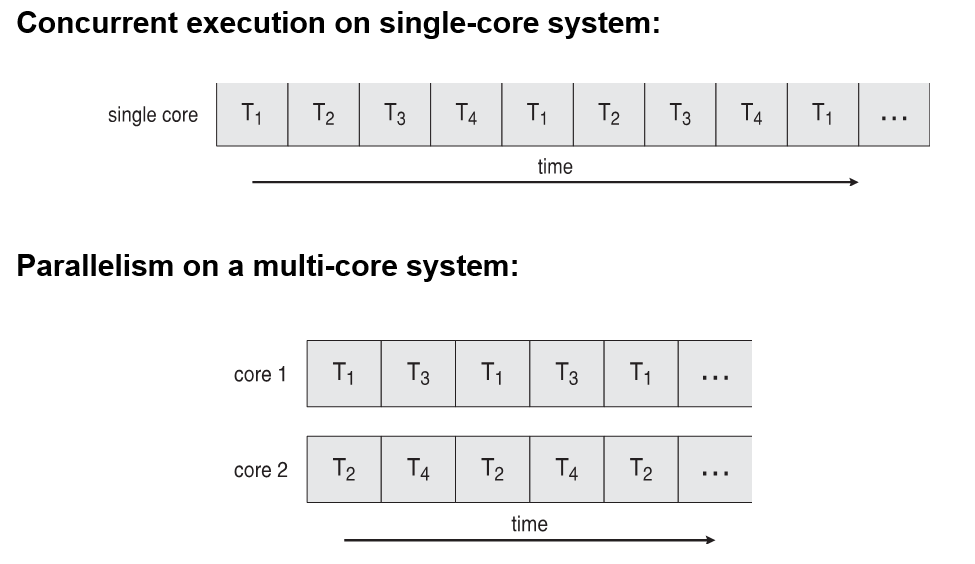
3. 性能的考虑。Kernels are generally multithreaded.

**#多核心编程**

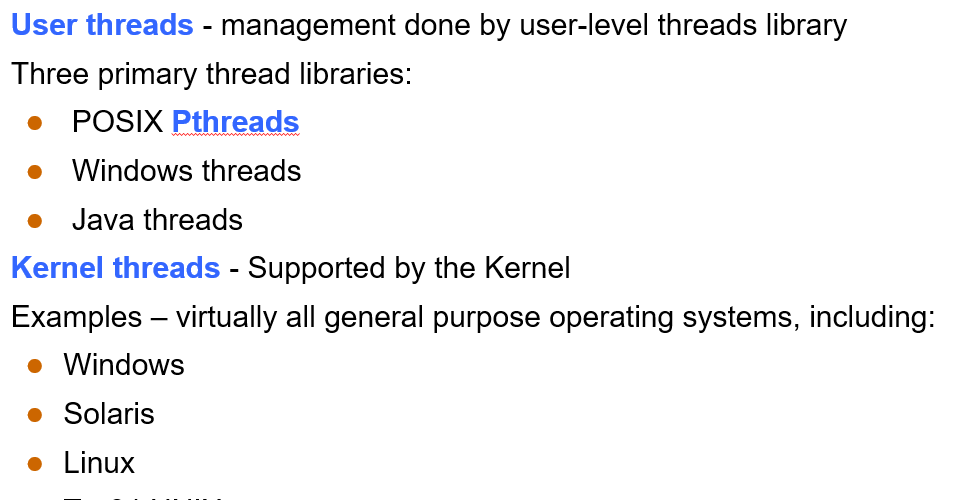
并行 parallelism

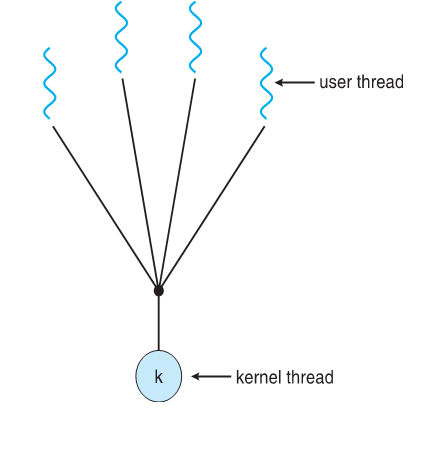
数据并行：将相同数据的子集同时分布在多个核心上对它们进行相同的操作。

任务并行：将线程分布在多个核心上，每个线程独立操作。



**#用户线程与内核线程**





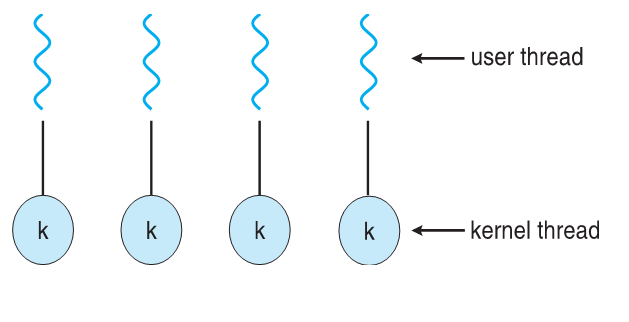
**#多线程模型**

多对一模型

将许多用户级线程映射到一个内核线程。

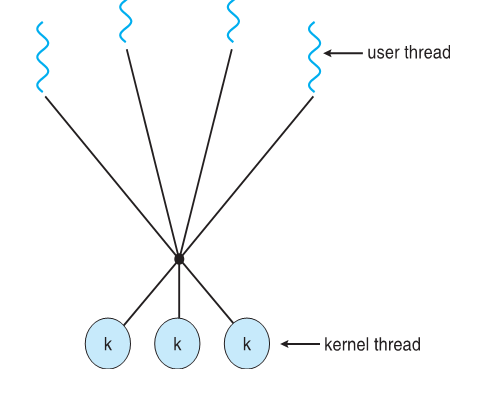
如果一个线程执行了阻塞系统调用，那么整个进程会阻塞。而且任意时刻只有一个线程能访问内核，多个线程不能运行在多处理器上。

一对一模型

将每个用户线程映射在一个内核线程上。

一个线程阻塞不影响另一个线程继续执行，因此有更好的并发性。

缺点是每创建一个用户线程就需要创建一个内核线程，会影响程序性能，因此限制了系统所支持的线程数量。

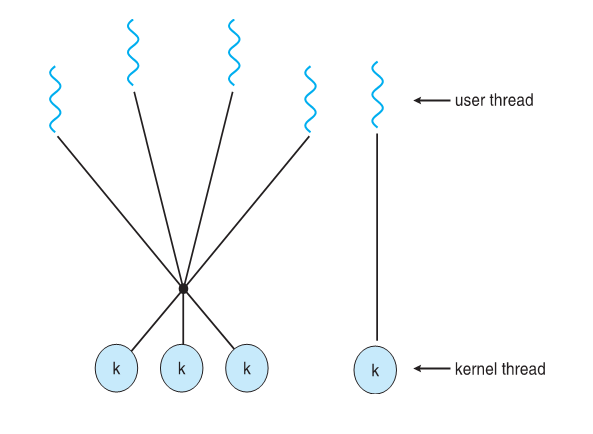


多对多模型

多路复用了许多用户线程到同样数量或更少数量的内核线程。

Allows the operating system to create a sufficient number of kernel threads.

二级模型



**#线程库**

线程库为程序员提供了创建和管理线程的API。Thread library provides programmer with API for creating and managing threads.

两种方法实现：

* + Library entirely in user space
  + Kernel-level library supported by the OS

\*Pthreads

独立线程是通过特定函数执行。创建线程pthread\_create(\*function)，新的线程会从\*function中开始执行。Pthread\_join()实现父线程等待子线程。

\*Win32线程

类似于Pthreads。创建线程CreateThread（DWORD function），新的线程从function中开始执行。WaitForSingleObject ()实现父线程等待子线程。

\*Java线程

Thread类 start()方法创建线程，在构造器Thread(method)中的method开始执行。

**#多线程问题**

\*隐式线程问题

随着线程数的增加，程序的正确性随着显式线程的增加变得更加困难。

方法：

线程池

OpenMp

Grand Central Dispatch

\*线程池

进程在开始时创建一定数量的线程，并放到池中等待工作。当服务器收到请求时，它会唤醒池中的一个线程（如果有可用的线程），并将要处理的请求传递给他。一旦完成了服务，返回池中等待工作。如果池没有可用线程，那么服务器会一直等到有空线程为止。

优点：

1. 通常用现有线程处理请求要比等待创建新的线程要块

2. 线程池限制了在任何时候可用的线程的数量。这对哪些不能支持大量并发线程的系统非常重要。

\*fork()和exec()

如果程序中的一个线程调用fork()，那么新进程会**复制所有线程**，或者新进程**只有单个线程**？都会。

如果一个线程调用了exec()，那么exec()参数所指定的程序会替换整个进程包括线程。

\*信号处理

信号在UNIX中用来通知进程某个特定事件已发生。根据需要通知信号的来源和事件的理由，信号可以同步或异步接收。不管信号是同步或是异步，所有信号具有相同的模式：

1. 信号是由特定事件的发生所产生的。

2. 产生的信号要发送到进程。

3. 一旦发送，信号必须被处理。（default; user-define）

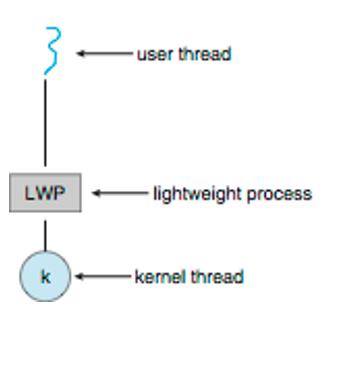
\*线程取消cancellation

线程取消是在线程完成之前来终止线程的任务。

* + **Asynchronous cancellation** terminates the target thread immediately
  + **Deferred cancellation** allows the target thread to periodically check if it should be cancelled

\*调度程序激活

内核与线程库之间的通信问题。这种协调目的是动态调整内核线程的数量以保证其最好的性能。

系统在用户和内核线程之间设置一种中间数据结构，这种结构是LWP，它表现为一种应用程序可以调度用户线程来运行的虚拟处理器，如果内核线程阻塞，LWP也阻塞，用户线程也阻塞。

通常每个并发阻塞系统都需要一个LWP，当LWP少时，则新的请求需要等待。

一种解决用户线程库与内核通信的方法被称为**调度程序激活**，内核提供一组LWP给应用程序，应用程序调度用户线程到一个可用的LWP上。