考点 6：线程的概念、线程的类别、线程与进程、管程的区别

PWB 原话：线程的概念，线程的类别，有什么用户级线程、内核级线程，然后线程与进程的关系。比如说一个线程阻塞了另外一个，这个进程为什么会阻塞？然后管程别搞混了，线程有 TCB，进程有 PCB。管程，它其实只是一个名称，它是属于操作系统。）

（无雨课堂练习、无小组讨论题）

3.8 线程机制 p142

1、线程的概念：

线程是隶属于进程的一个实体，是比进程“更小”的能独立运行的基本单位，能进一步提高系统的并发程度和吞吐量，又不至于过高地增加系统开销。

线程拥有的资源：

1.一个线程 ID：用以唯一地标识线程。

2.一组寄存器：表示线程运行时的处理器状态，包括程序计数器、程序状态字、通用寄存器、栈指针等。

3.堆栈：一个用户栈、一个内核栈，分别供线程在用户态和内核态下运用时使用。

4.一个私有存储区：存放线程很少的私有数据。如全局变量 errno，用来存放某线程系统调用失败时返回的错误码，自然是不能和进程中其他线程共享的，不然会混淆，不知道是哪个线程系统调用失败了。

5.线程控制块 TCB：存放线程的管理信息，如线程 ID、寄存器的值、线程状态、调度优先级、相关统计信息、信号掩码等。

3、进程与线程、管程的区别：

（p144 3.线程与进程的比较）

“然后管程别搞混了，线程有 TCB，进程有 PCB。管程，它其实只是一个名称，它是属于操作系统。”——PWB

将传统进程的两个属性分开，能显著提高系统的并发程度，降低 CPU 切换开销。

形象一点的理解：进程是任务，则线程就是子任务。

几个方面对比进程&线程：调度、并发性、拥有资源、系统开销

（1）调度：线程作为调度和执行的基本单位，进程作为资源分配和拥有的基本单位。

同一进程中的线程切换不会引起进程切换。

不同进程的两个线程切换会引起进程切换。

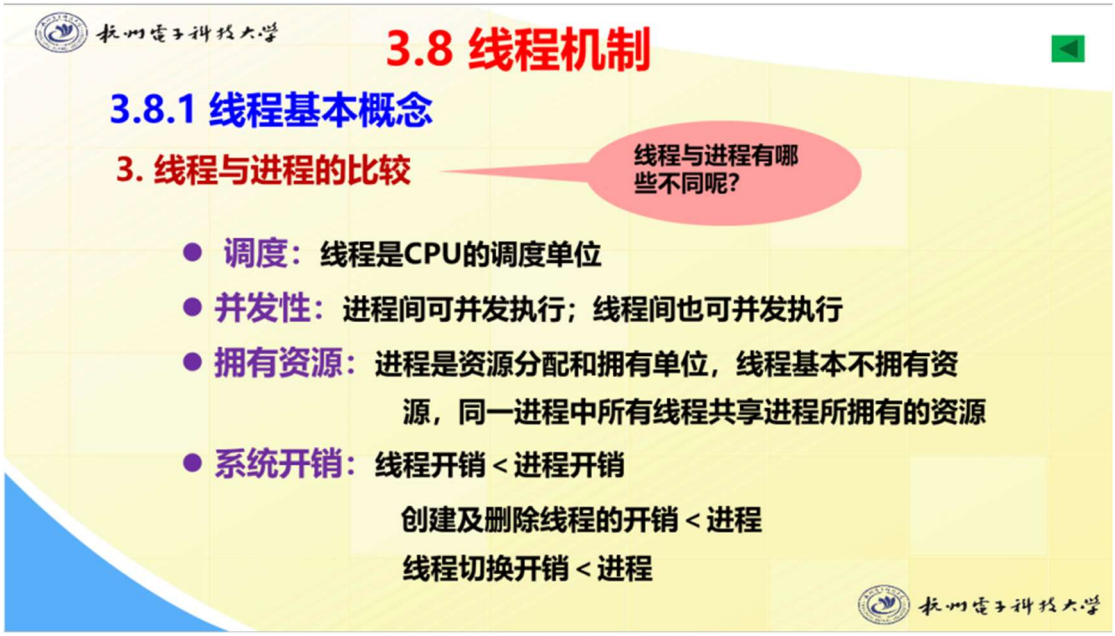
（2）并发性：引入线程后，不仅进程间可并发执行，且一个进程中的多个线程间也能并发执行。系统并发度更高，能有效使用系统资源和提高系统吞吐量。

Because：没有引入线程的操作系统，若只有一个文件服务进程，则只要有一个原因导致阻塞，整个进程就阻塞了。但如果一个进程里有多个线程，一个线程阻塞了，其他线程还是可以继续工作的。

（3）拥有资源：进程是拥有资源的独立单位。线程基本上不拥有资源（除了一点点运行时必不可少的资源），但可以访问所属进程的全部资源，即一个进程的代码段、数据段、系统资源（如已打开的文件、I/O 设备等）

（4）系统开销：线程开销<进程开销

开销：创建、删除、切换、通信



管程 p80

本章习题 19（p167）一个管程由哪几部分组成？管程中引入条件变量有什么用处？

1. 管程的组成

（1）管程的名字

（2）局部于管程的共享数据结构的说明

（3）对该数据结构进行操作的一组过程，每个过程完成进程对上述数据结构的某种操作。

（4）对局部于管程的共享数据结构进行初始化的代码

```
Monitor monitor_name {          /*管程名*/
    variable declarations;      /*共享数据结
```

第 3 章 进 程

```
    procedure P1(…) {… }   /*操作过程*/
    procedure P2(…) { … }
                ⋮
    procedure Pn(…) { … }
    initialization code;        /*设初值语句*/
}
```

语法结构

2、管程特点

（1）任何进程只能通过调用管程提供的过程入口才能进入管程访问共享数据；就如同使用临界资源，就要先通过其信号量的申请。

（2）任何时刻，仅允许一个进程在管程中执行某个内部过程。

管程主要具有如下几个特性：

(1) 局部于管程的数据结构只能被管程内的过程访问，任何外部过程不能访问；而管程内的过程也只能访问该管程内部的数据结构。

(2) 一个进程若想访问管程内的数据结构(共享资源)，只能通过调用管程内的某个过程实现间接访问。

(3) 任一时刻，管程中只能有一个活跃进程，即只能有一个进程在管程中执行管程的某个过程，其他任何调用管程的进程都将被阻塞，直到管程变成可用，这一特性使管程能有效地实现互斥。管程是编程语言的组成部分，编译器知道它们的特殊性，因此可以采用

3、条件变量

局部于管程的变量有两种：

普通变量

条件变量（用于控制进程阻塞和唤醒）

一个条件变量通常对应一个等待队列，当条件不满足时，进程将进入这个等待队列阻塞，直到条件满足才被唤醒。

系统只提供两个操作：wait（）阻塞和 signal（）唤醒

4、管程的优点

(1)保证进程互斥地访问共享变量，并方便地阻塞和唤醒进程。管程可以以函数库的形式实现。相比之下，管程比信号量好控制。

(2)管程可增强模块的独立性：系统按资源管理的观点分解成若干模块，用数据表示抽象系统资源，使同步操作相对集中，从而增加了模块的相对独立性

(3)引入管程可提高代码的可读性，便于修改和维护，正确性易于保证：采用集中式同步机制。一个操作系统或并发程序由若干个这样的模块所构成，一个模块通常较短，模块之间关系清晰。

5、管程的缺点

（1）大多数常用的编程语言中没有实现管程，如果某种语言本身不支持管程，那么加入管程是很困难的。

（2）虽然大多数编程语言也没有实现信号量，但可将 P、V 操作作为一个独立的子例程或操作系统的管理程序调用加入。

线程管理（p144）

1）线程状态：就绪、运行、阻塞、终止

2）线程控制

（1）线程创建：创建一个进程，再为进程创建第一个线程，分配并设置 PCB，建立堆栈，最后插入就绪队列，等待 CPU 调度。

（2）线程终止：完成任务自行终止 or 运行时出现异常状况被外界强行终止。

注意：线程终止后不立即释放 TCB，等待父进程完成信息收集后再彻底撤销。

（3）线程阻塞：运行中的线程磁盘 I/O 完成，主动调用线程阻塞函数（或系统调用）阻塞自己，线程释放 CPU，变成阻塞，插入相应阻塞队列，等待系统重新调度。

（4）线程唤醒：阻塞中的线程，利用相关线程调用线程唤醒函数，变成就绪状态，插入就绪队列，等待 CPU 调度。

3）线程同步：所有进程的同步机制适用于线程间的同步。

4）线程调度：所有进程的调度算法适用于线程调度。

5）线程通信：同一进程中的线程通信可在用户态直接进行，不需要调用内核的基本调用。不同进程中的线程通信时仍然需要内核的干涉以提供保护和通信机制。

多线程的应用：word 程序、Web 服务器、媒体播放器

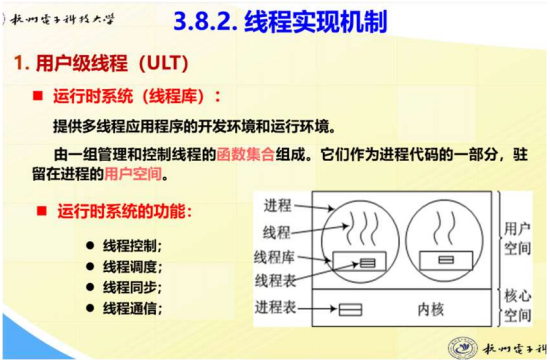
条件：同一进程中多个线程具有相同的代码和数据，这些线程之间或是合作的（执行代码的不同部分），或是重复完成相同的功能（执行相同的代码）。

2、线程的类别

（p146 3.8.2 线程的实现机制）

不同系统的实现方式：用户级线程、内核级线程、用户级/内核级相结合的组合方式。

1.用户级线程 ULT（User Level Threads）

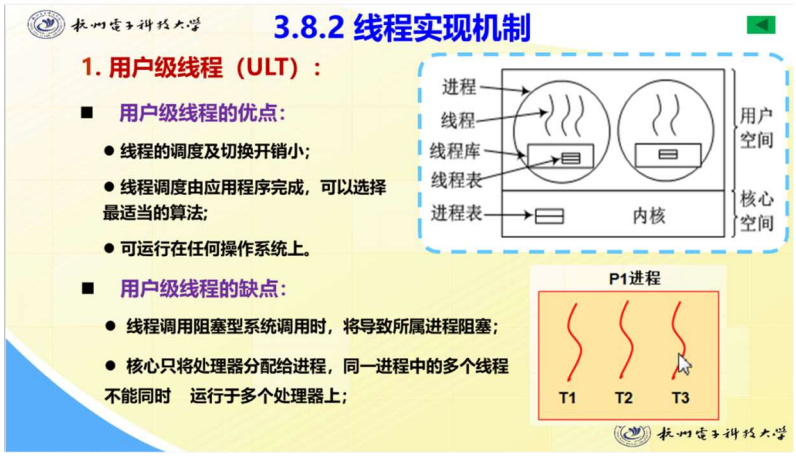


用户级线程是在用户空间中实现的。

操作系统内核不知道线程的存在，支队常规进程进行管理，调度以进程为单位。

线程表：记录该进程各线程情况，每个线程占据一个表项，保存在用户空间，由线程库进行管理。

进程表：记录每个进程的情况，保存在内核中。



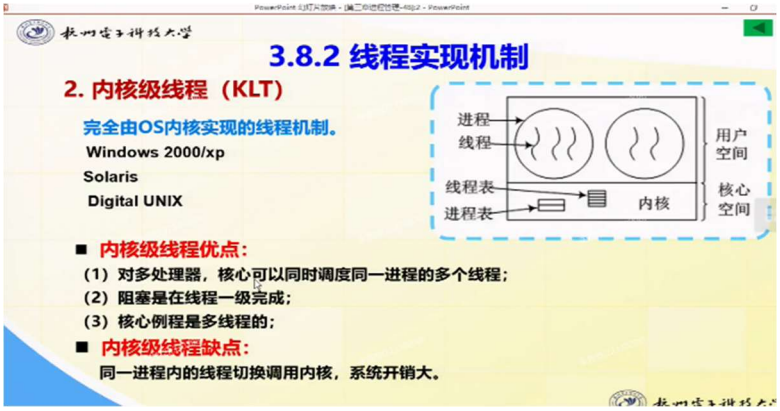
缺点解释：

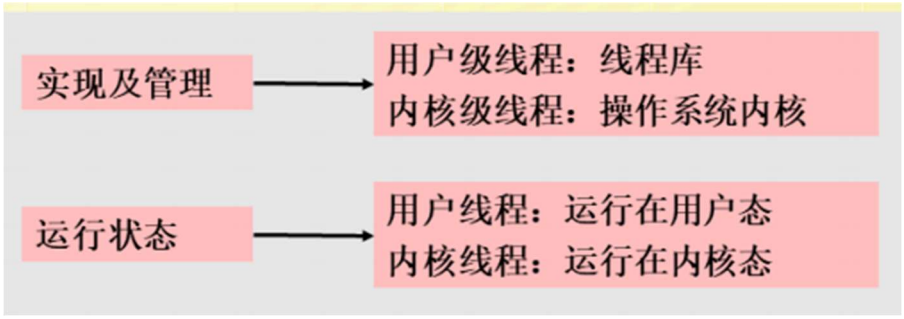
（1）因为一个用户级线程执行系统调用的时候，操作系统会把这次系统调用当作整个进程的行为，引起该进程整个的阻塞。

（2）因为操作系统不知道用户级线程的存在，所以只会给一个进程分配一个 CPU，所有用户级进程共享这个 CPU，而不能在多个 CPU 上并行执行。

2. 内核级线程 KLT（Kernel Level Threads）

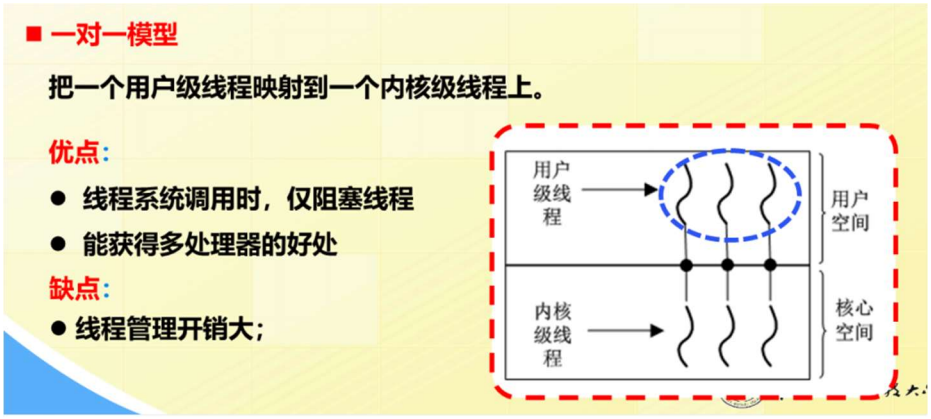
内核级线程在内核空间中实现。


杭州电子科技大学
3.8.2 线程实现机制
2. 内核级线程（KLT）
完全由OS内核实现的线程机制。
Windows 2000/xp
Solaris
Digital UNIX
进程
线程
线程表
进程表
内核
用户空间
核心空间
■ 内核级线程优点：
（1）对多处理器，核心可以同时调度同一进程的多个线程；
（2）阻塞是在线程一级完成；
（3）核心例程是多线程的；
■ 内核级线程缺点：
同一进程内的线程切换调用内核，系统开销大。


实现及管理
用户级线程：线程库
内核级线程：操作系统内核
运行状态
用户线程：运行在用户态
内核线程：运行在内核态

3. 组合方式


杭州电子科技大学
3.8.2 线程实现机制
3. 组合方式
内核支持内核级线程，线程库支持用户级线程，如Solaris
■ 多对一模型
把多个用户级线程映射到一个内核级线程上
优点：
● 线程管理开销小，效率高；
缺点：
●线程系统调用时，将导致所属进程阻塞；
●核心只将处理器分配给进程，同一进程中
的两个线程不能同时 运行于两个处理器上；
用户级线程
内核级线程
用户空间
核心空间


■ 一对一模型
把一个用户级线程映射到一个内核级线程上。
优点：
● 线程系统调用时，仅阻塞线程
● 能获得多处理器的好处
缺点：
● 线程管理开销大；
用户级线程
内核级线程
用户空间
核心空间

■ 多对多模型

把多个用户级线程映射到较少或同样数量的内核级线程上。

优点：

- **线程系统调用时，仅阻塞线程**
- **能获得多处理器的好处**
- **线程管理开销 不至于增加太大**