西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

4-2 用户线程和内核线程

Threads
线程

- Kernel-supported threads内核支持的线程 (Mach and OS/2).

- User-level threads; supported above the kernel, via a set of library calls at the user level (Project Andrew from CMU). 用户级线程；在内核之上，通过用户级的库调用

Kernel Threads
内核线程

Supported by the Kernel
由内核支持，在内核空间执行线程创建、调度和管理

Thread is unit of CPU scheduling

Examples例子

- Windows XP/2000
- Solaris
- Digital UNIX

kernel-level thread
内核线程

依赖于OS核心，由内核进行创建、撤销和切换

Windows NT和OS/2支持内核线程；

- 内核维护进程和线程的上下文信息；
- 线程切换由内核完成；
- 一个线程发起系统调用而阻塞，不会影响其他线程的运行。
- 时间片分配给线程，所以多线程的进程获得更多CPU时间。

User Threads
用户线程

- Thread Management Done by User-Level Threads Library

  由用户级线程库进行管理的线程

- 线程库提供对线程创建\调度和管理的支持，无需内核支持。

- Process is still unit of CPU scheduling from OS kernel perspective

- Examples例子

- POSIX Pthreads
- Mach C-threads
- Solaris threads

user-level thread
用户线程

不依赖于OS核心，应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制用户线程。调度由应用软件内部进行，通常采用非抢先式和更简单的规则，也无需用户态/核心态切换，所以速度特别快。

- 用户线程的维护由应用进程完成；
- 内核不了解用户线程的存在；
- 用户线程切换不需要内核特权；
- **DRAWBACKS:**
  - 如果内核是单线程的,那么一个用户线程发起系统调用而阻塞，则整个进程阻塞。
  - 时间片分配给进程，多线程则每个线程就慢。

User threads and kernel threads

用户线程与内核线程

调度方式：

内核线程的调度和切换与进程的调度和切换十分相似，用户线程的调度不需OS的支持。

调度单位：

用户线程的调度以进程为单位进行，在采用时间片轮转调度算法时，每个进程分配相同的时间片。对内核级线程，每个线程分配时间片。