文章编号：1007 - 144X(2001)04 - 0033 - 03

操作系统线程实现技术及特性

肖竟华

(武汉科技大学 信息科学与工程学院，湖北 武汉 430081)

摘 要：详细描述了线程的特性及两种线程实现技术，并分析了各自的优缺点，总结了现代操作系统采用的一种混合实用技术，并客观分析了线程技术的实用范围。

关键词：线程技术；操作系统；核心线

中图法分类号：TP 316 文献标识码：A

1 线程调度及进程调度

随着多处理机的发展，为了提高系统的执行效率，减少处理机的空转时间和切换调度时间，进程内多线程概念被提了出来。一个进程内的基本调度单位称为线程或轻权进程，这个调度单位既可以由操作系统内核控制，也可由用户程序控制。进程是资源分配的基本单位，也是抢占处理机的调度单位，它拥有一个完整的虚拟地址空间[1]。而线程与资源分配无关，它属于某一个进程，并与进程内的其他线程一起共享进程的资源。当进程发生调度时，不同的进程拥有不同的虚拟地址空间，而同一进程内的不同线程共享同一地址空间。线程由相关堆栈(系统栈或用户栈)寄存器和线程控制表 TCB 组成。寄存器可被用来存储线程内的局部变量，但不能存储其他线程的相关变量。进程切换时将涉及到有关资源指针的保存及地址空间的变化等问题。线程切换时由于同一进程内的线程共享资源和地址空间，将不涉及资源信息的保存和地址变化问题，从而减少了操作系统的开销时间。而且，进程的调度与切换都是由操作系统内核完成，而线程即可由操作系统内核完成，也可由用户程序完成。

2 线程的执行特性

2.1 线程的 3 个基本状态

线程有执行、就绪和阻塞 3 个基本状态。但是，线程没有进程中的挂起状态，即线程是一个只与内存和寄存器相关的概念，它的内容不会因交换而进入外存。针对线程的 3 种基本状态，存在以下 5 种基本操作来转换线程的状态。

① 派生。线程在进程内派生出来，它既可由进程派生，也可由线程派生。用户一般用系统调用(或相应的库函数)派生自己的线程。一个新派生出来的线程具有相应的数据结构指针和变量，这些指针和变量作为寄存器上下文放在相应的寄存器和堆栈中。

② 阻塞。若一个线程在执行过程中需要等待某个事件发生，则被阻塞。阻塞时寄存器上下文、程序计数器及堆栈指针都会得到保存。

③ 激活。若阻塞线程的事件发生，则该线程被激活并进入就绪队列。

④ 调度。选择一个就绪线程进入执行状态。

⑤ 结束。若线程执行结束，它的寄存器上下文及堆栈内容等将释放。

在某些情况下，某个线程被阻塞也可导致该线程所属的进程被阻塞。

2.2 线程的同步

由于同一进程中的所有线程共享该进程的所有资源和地址空间，任何线程对资源的操作都会对其他相关线程带来影响。因此，系统必须为线程的执行提供同步控制机制，以防止因线程的执行而破坏其他的数据结构和给其他线程带来不利的影响。线程的同步控制机制与进程同步控制机制相同，在此不作进一步论述。

3 线程的实现方式

为了支持用户进程内的多线程，有 2 种典型的线程实现方式，即用户线程和系统线程(核心级线程)。在同一系统中，有的使用纯用户级线

收稿日期：2001-08-27.
作者简介：肖竟华(1955-)，女，湖北来凤人，武汉科技大学信息科学与工程学院高级工程师.

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

程,有的使用纯核心级线程,例如 Windows NT 和 OS/2 就是这样的系统;有的则混合使用用户级线程和核心级线程,例如Solaris 操作系统。

3.1 用户级线程实现

用户级线程是对线程进行全面管理,操作系统内核只对进程进行管理。为了对用户级线程进行管理,操作系统提供在用户空间执行的线程库,该线程库提供创建、调度、撤消线程功能。同时该线程库也提供线程间的通信、线程的执行及存储线程上下文的功能。用户级线程只使用用户堆栈和分配给所属进程的用户寄存器。当一个线程被派生时,线程库为其生成相应的线程控制块 TCB 等数据结构,并为 TCB 中的参数赋值和把该线程置于就绪状态。其处理过程与进程创建大致相似,其特点如下:

① 为用户提供一个简明的同步并行编程环境,加入原文线程库将用户的同步请求转化成对操作系统的异步请求。

② 用户级线程的调度算法和调度过程全部由用户自行选择和确定,由用户级线程库实现,与操作系统内核无关,因而不需操作系统内核的特殊支持[2]。

③ 用户级线程的调度算法只进行线程上下文切换而不进行处理机切换,且其线程上下文切换是在内核不参与的情况下进行的,即线程上下文切换只是在用户栈、用户寄存器等之间进行切换,不涉及处理机的状态,无运行模式的切换开销。新线程通过程序调用指针的变化使得程序计数器变化而得以执行,因而系统开销小。

④ 因为用户级线程的上下文切换与内核无关,所以可能出现如下情况:即当一个进程由于 I/O 中断或时间片用完等原因造成该进程退出处理机时,而属于该进程的执行中线程仍处于执行状态。也就是说,尽管相关进程的状态是阻塞(或等待)的,但所属线程状态却是执行的,因而不能转到其他线程执行。

⑤ 不能实现进程内线程在多处理机上真正并行运行。如果系统拥有多个处理机而操作系统的调度单位是进程,无法感知进程内多线程存在,因而无法使用户进程在多处理机上同时运行。

3.2 核心级线程实现

核心级线程由操作系统内核进行管理,负责线程在处理机上的调度与切换,操作系统内核给应用程序提供相应的系统调用和应用程序接口 API,以使用户程序可以创建、执行和撤消线程。

操作系统内核必须为每个线程保存一系列描述信息,线程被用作处理机的执行单位,操作系统对处理机分配改为以线程为对象。因此核心级线程可以完全并行地在系统的不同处理机上运行,操作系统可以支持核心级线程与物理处理机之间的捆绑,让用户干预核心级线程选定处理机运行。当核心级线程从用户程序转操作系统核心程序运行后,等某事件发生引起线程阻塞或有更高优先级线程可以运行,操作系统调度程序会作线程切换处理,现场保护和恢复均在线程相关的数据结构之中进行。

核心级线程实现有如下特点:① 可以支持进程内多线程在多处理机上真正并行执行;② 核心级线程技术也可用于内核程序自身,从而提高操作系统内核程序的执行效率;③ 核心级线程不会出现进程处于阻塞或等待状态,而线程处于执行状态的情况;④ 核心级线程比用户级线程开销要大一些,因为核心级线程的管理都由内核程序进行,需要运行模式的切换。

表 1 给出了在SPARC 工作站上用户级线程、核心级线程及进程进行上下文切换时各自的时间开销。

表 1 SPARC 工作站上下文切换所需时间 /s

操 作	用户级线程	核心级线程	进程
创 建	52	350	1 700
使用信号量同步	66	390	200

3.3 用户级线程和系统级线程混合实现

为了向用户提供既能并行编程环境,又能最大限度地发挥多处理机的并行性,有些系统,例如 Solaris 2.x 提供了用户级线程和系统级线程 2 种功能。在这种实现系统中,用户利用线程库提供的并行程序设计界面编写用户并行程序。核心级线程的操作系统调用界面只提供给线程库使用,用户可以完全不知线程库使用了几个核心级线程,也可以通过线程库的接口申请更多的核心级线程,所以核心级线程都由线程库管理使用,由线程库选择用户级线程在哪一个核心级线程上运行。

用户利用线程库函数编写的并行程序作为用户执行文件。在进程创建时,被映射到进程虚地址空间中,第一个核心级线程随进程同时创建,用户程序的主函数在该线程上运行,随后,用户程序可以调用线程库创建新的用户级线程,也可请求线程库到核心去申请更多核心级线程。每当线程库申请创建一个核心级线程,线程库让该核心级线程运行,并当核心级线程相关的初始化工作后,即转到用户级线程调度程序运行。在该核心级线

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

程上运行的线程库调度程序可以调度用户级线程, 将控制交给被调用户线程。进程内使用多少个核心级线程可以由系统提供默认值, 也可由用户动态提出申请。一般来说, 核心级线程个数与系统处理机个数相当为宜, 太多则占用系统资源, 太少则不能让进程内多任务真正在处理机上最大程度地并行。

4 线程的适用范围

尽管线程可以提高系统的执行效率, 但并不是在所有的计算机系统中线程都是通用的。事实上在那些很少做进程调度和切换的实时系统、个人数字处理系统中, 由于任务的单一性, 设置线程相反会占用更多的内存空间和寄存器。

使用线程的最大好处是在多个任务需要处理机时, 减少处理机的切换时间, 而且线程的创建和结束所需的系统开销也比进程的创建和结束要小得多。由此可见, 最适合使用线程的系统是多处理机系统。在多处理机系统中, 同一用户程序可以根据不同的功能划分为不同的线程, 放在不同的处理机上执行。

当用户程序可以按功能划分为不同的小段时, 单处理机系统也可因使用线程而简化程序的结构和提高执行效率。其典型应用如下:

① 服务器中的文件管理或通信控制。在局域网的文件服务器中, 对文件的访问要求可被服务器进程派生的线程进行处理。由于服务器同时可能接受许多个文件访问要求, 则系统可以同时生成多个线程来进行处理。如果计算机系统是多处理机的, 这些线程还可安排到不同的处理机上执行。

② 前后台处理。许多用户都有过前后台处理经验, 即把一个计算量较大的程序或实时性要求不高的程序安排在处理机空闲时执行, 对于同一个进程中的上述程序来说, 线程可被用来减少处理机切换时间和提高执行速度。例如在表处理进程中, 一个线程可被用来显示菜单和读取用户输入, 而另一个线程则可用来执行用户命令和修改表格。由于用户输入命令和命令执行分别由不同的线程在前后台执行, 从而提高了操作系统的效率。

③ 异步处理。程序中的两部分如果在执行上没有顺序规定, 则这两部分程序可用线程执行。

另外, 线程方式还可用于数据的批处理以及网络系统中的信息发送与接收和其他相关处理等。

参考文献:

[1] 孟庆昌. 操作系统教程[M]. 西安: 西安电子科技大学出版社, 1997.

[2] Tanenbaum A S. 分布式操作系统[M]. 北京: 电子工业出版社, 1999.

Thread Implementation Technology of an Operating System

XIAO Jin-hua

Abstract: The characteristics of threads and technologies to implement user-level threads and kernel-level threads are described. The advantages and disadvantages of these technologies are analyzed. An advanced technology to implement the thread is presented based on the above analysis and its applicable range is pointed out.

Key words: thread; operating system; kernel-level thread

XIAO Jin-hua: Senior Engineer; Department of Information Science and Engineering, Wuhan University of Science and Technology, Wuhan 430081, China.

[责任编辑: 刘美玲]

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net