·软件技术实现与研究·

线程的模型及其在操作系统中的实现

刘 忠 夏为民 邓胜兰
(国防科大计算机系 长沙 410073)

摘要 线程在当代操作系统中是一个非常重要的概念。该文讨论了线程的模型，并研究线程在操作系统中的实现。

关键词 线程 并行及分布式操作系统 模型

Thread's Model and It's Implementation in Operating System

Liu Zhong Xia Weimin Deng Shenglan
(Department of Computer, National University of Defense Technology Changsha 410073)

【 Abstract 】 Thread is important in modern operating system. In this paper, we discuss the model of the thread and how to implement the thread mechanism in operating system kernel.

【 Key words 】 Thread; Parallel or distributed operating system; Model

在传统的操作系统中，每个进程只有一个地址空间和一个控制流。但这种方式有两个缺点：(1)当需要在同一地址空间产生多个控制流，如共享内存的并行程序设计时，进程就无能为力；(2)并发执行时进程间切换的开销特别大。为此，现代操作系统中引入线程的概念。*

1 线程的模型

1.1 线程的概念

线程是进程中一个执行实体，是一个单独控制流。它包含如下一些内容：(1)程序计数器；(2)寄存器集；(3)栈；(4)子线程指针；(5)状态；(6)优先数。

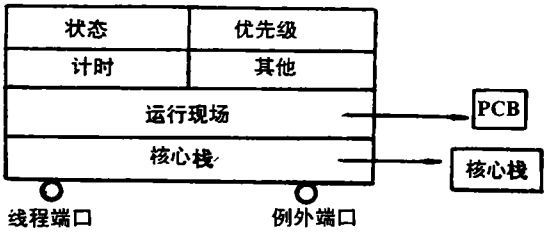

图1 线程概念图

线程是系统调度和运行的基本单位，拥有自己的核心栈和处理机现场。在同一进程内可有多个线程，各线程可独立被调度运行，但共享的资源(包括地址空间)却不能为某线程保护，因此在线程中安排同步互斥是非常重要的。

与传统的进程概念一样，线程也有各种调度运行状态。如 Wait、Susspend、Terminated 以及 Swapped 等。除此之外，线程还有优先数，用以核心进行 CPU 调度。详见图 1。

1.2 线程作用

由于共享地址空间，线程间的调度开销是较小的，因此有时线程又称为"轻型线程"(lightweight processes)。

当在单 CPU 状态下，线程间共享同一 CPU，按时间片轮转运行。当有多 CPU 时，线程可分布到各个 CPU 上运行，这样，就可以实现共享内存的并行程序。

除了用于共享内存的并行程序设计，在分布式系统中线程也是十分重要的。在分布式环境中，通常采用 Client / Server 结构和 RPC 机制，对于 Server 来说，如果只有单控制流，则当为一个 Client 服务时阻塞(如读盘)，则其它客户的服务也被阻塞，当 Server 具有多线程时，每个 Client RPC 调用时 Server 将创建一个线程进行服务。当一个线程阻塞时，其它 Server 线程将会被调度执行。

另外，在复杂程序设计，如经常要创建和撤消一

---

*刘 忠 男，29 岁，博士生，主要从事分布式和并行操作系统的研制工作

收稿日期：1997-05-07

个任务和异步处理(APC)时，线程是十分关键的。

1.3 线程的组织

一进程中的线程可组织成多种结构，一种是总控结构。在进程中有一个线程作为总控线程，根据外部要求安排一个空闲线程或创建一个线程进行服务。

另一种是并列结构，每个线程都是有平等的功能，独自地应答各处请求进行服务。最后一种是流水结构，将某一任务分为若干个线程，分步顺序执行。

1.4 线程的同步、互斥

由于同一进程中线程共享内存空间，因此对于共享数据的访问必须安排同步与互斥。对于这些临界段，我们可以采用信号灯和 monitor、条件变量、barrier 等方法。例如对于作为 OSF／1 基础的 Mach，它提供了锁的同步机制。在读写临界区时，先判锁，已有锁则阻塞等待，没有锁则加锁，读写完成后再解锁。在这里，锁是 mutex_t 说明的：

```
mutex_t    lock;        /*二值信号灯，如
mutex_lock(lock);
/* 这是一个原子操作，判锁，如加锁则阻塞等待；  否则加锁后继续执行          */
/*读写*/
mutex_unlock(lock)
/*解锁*/
```

1.5 线程实现的两种方法

有两种方法实现线程机制。一种在操作系统的内核中实现线程的机制，对线程的操作通过系统调用的方法来实现。另一种在用户空间仿真线程的机制，通过一个线程的运行时库，其中包括各种操作线程的函数。当一个用户程序中对线程进行操作时，它调用线程运行时库的相应函数来完成。这些函数还负责对线程进行切换。它负责保护寄存器的现场(但在用户空间)，调度新的线程执行。由于不陷入内核执行，它的调度会更加快一些。这种实现方法多在语言中实现，如 UNIX 下的一些 C 语言 Thtead 运行时库以及 Internet 上的编程语言 Java 的线程都采用此种方法，但这种方法难以实现阻塞，也难以实现剥夺，不如操作系统的线程机制优越。

2 线程在操作系统中的实现

Mach 是卡内基·梅隆大学开发的一个新型操作系统。由于它采用了许多新的设计思想，如微内核、面向对象、多线程、基于端口的通信、虚存与存储对象等，因此，能很好地支持并行多处理和分布式环境，成为当代操作系统的一个典范。从现代商业操作系统的两大阵营看，在 UNIX 方面，Mach 是 OSF(Open Software Foundation)开发的操作系统 OSF／1 的基础，从 Windows NT 看，其设计思想与具体实现中处处闪耀着 Mach 的光辉。

Mach 中的线程实现涉及任务、线程和调度。

2.1 Mach 的任务

任务是 Mach 资源的集合，由虚地址空间和一组在这个空间内执行的线程的集合。任务是被动的。

除了虚空间和线程，任务还包含很多端口与其它属性。任务所固有的端口有 4 类：任务端口、引导端口、例外端口、注册端口。除这些端口外，任务还可以动态地创建端口。任务端口是任务的标志自己的端口。当任务请求内核进行某项服务时，它向此端口发消息。引导端口用来初始化。UNIX 进程还通过这个端口与 UNIX 仿真器通信。例外端口用来使系统能向任务报错，调试器要用到例外端口。注册端口是一组端口，用来提供与标准系统服务器的通信。这些服务器包括名字服务器和网络服务器等。

任务还有其它属性，如是否阻塞以及调度参数等。例如可以指定某一线程在某个处理机上运行。

Mach 提供了一组原语来创建或中止、吊起和恢复某一任务，还用原语来设置线程的优先级，以及指定某线程在哪个处理机上运行。除此之外，还可获得运行时间，占有内存空间的信息，以及获取任务中线程的表等。

2.2 Mach 中的线程

线程是 Mach 中的执行实体，它们执行自己的指令并管理自己的处理机集，备有自己的核心线以用来线程切换。每一线程必须属于某一任务，同一任务中的线程共享统一的地址空间，因此同一任务中不同线程间的切换是相当容易的。

线程有自己的状态，如睡眠、吊起、运行、不可中断、空闲、换出等状态，还有计数和优先数，以备调度之用。线程还有自己固有的两个端口：一个是线程端口，用来标志自身；另一个是例外端口，当线程发生例外文件时，核心向线程的例外端口发消息，让此例外端口的接收者处理例外。

Mach 的线程减轻了上下文切换的耗费，同时创建与线程间的通信开销大大低于进程，从而提高了并发性，并可支持各种并行模型，特别是中、小程度的并行程序。

对线程，Mach 也提供了一组控制原语，能创建、终止线程，还能阻塞或非阻塞地等待另一线程的终止，以及暂时放弃 CPU 权利等。

2.3 Mach 的线程调度

为了支持多处理机的并行，Mach 引入了处理机集的概念。处理机集是一个抽象的数据结构，它由 CPU 组成 。每个 CPU 只能属于一个处理机集。一组线程可以通过软件的方法指定到某一处理机集工作。线程调度是公平而有效的。除此之外，Mach 让任务对其中的线程有很大的控制权。一个任务可以指定一个重要的线程独占一个 CPU 处理机集，从而确保这个线程总在运行. 同时它还能动态改变线程所属的处理机集，保护负载平衡。

Mach 的线程调度是基于优先数的。优先数从 0 到 31，0 优先权最高而 31 最小。每个线程有 3 个优先数：一个是基础优先数(base priority)，线程可以在一定范围内自已设定，第二个优先数标志线程自己所能设置基础优先数，由于优先数越小优先权越大，因此它界定了本线程优先数所能达到的范围。第三个优先数是当前优先数，用于调度，它是由基础优先数加上一个变量构成。这个变量通过线程的当前使用的 CPU 量的一个函数来计算。

每个处理器集都有一个运行队列集。对每个优先级都有一个队列，共有 32 个队列。当一个优先数为 n 的线程变得可以运行，则它放到队列 n 的尾部。

每个运行队列都有 3 个变量与之相关。一个是同步信号灯，用来锁住队列的数据结构。它保证每次只有一个处理机访问队列。第二个是所有队列中线程的计数，用来看本处理机集还有几个可行线程在等待。第三个是指向最低优先数的线程的指针 。

除此之外，每个 CPU 还有一个自己的运行线程队列。每个 CPU 自己的运行队列保持着那些永久地捆绑在 CPU 上的线程。例如 I／O 设备的设备驱动程序。

当现在正在运行线程被阻塞、死亡或用尽它的时间片时，CPU 运行内核调度。首先判断本 CPU 的线程队列是否为空，不为空则调度优先级最高的线程运行。否则再判断 CPU 所居的处理机集的线程队列是否为空，是则运行 IDLE 线程，以等待新的可运行线程，否则找到其中优先级最高的线程，投入运行。

很明显，本 CPU 运行队列总高于处理机集的线程运行线程。注意，Mach 的调度是可以剥夺的。因为线程随着时间的推移，每个时钟中断时它的优先数都在增加，最终会移到优先级较低的队列中。这时即使它不是阻塞或死亡，也会因时间的推移而降入低优先级队列，由另一最高优先级的线程接替而被剥夺。

线程的运行是分时间片的。每时间片到期时，都要运行一次调度线程进行调度。在多处理机集并行时，时间片的长度是可变的，可运行的线程越多，CPU 越少，则时间片越短。这种办法在线程较多时能给各个线程以很好的即时反应，而在相对负载较轻的系统，提供较高的效率。

线程的运行参见图 2。

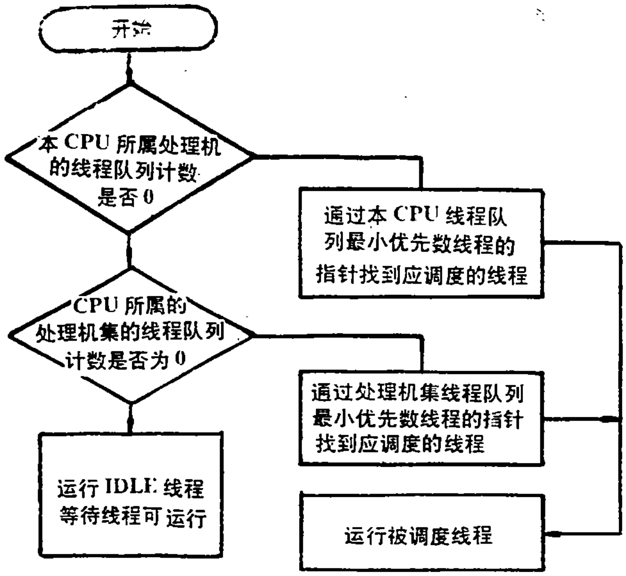

图 2　线程运行图

本文讨论了线程的基本模型及其在操作系统中的实现方法。随着并行化的提高和网络的发展，线程必将发挥更大的作用。

参考文献

1 Uahalia U. Unix Internals. Prentice Hall,1996

≈≈≈

读者意见选登

自从本刊设置"意见征询卡"以来，许多读者对本刊给予很高的评价，这是对我们的鞭策和鼓励；同时，也有许多读者提出许多改进意见，我们将在今后工作中认真总结，加以改进．但也有一些意见（如希望出彩色版），由于条件限制，不能满足读者的要求，请给予谅解．因篇幅有限，下面仅举几个代表性的读者的反馈信息．例如：山西赵瑞宏同志认为，"刊物内容体现了最新科技的宗旨，但稿件来源太局限于华东地区"；北京陈安同志认为，"刊物体现了'工程'特色，所发表文章短小精悍，但理论性太薄弱"；山东陈俊霞同志认为，"理论性、技术性、实用性并重，是计算机工程技术人员不可多得的技术资料"．这只是几个例子．我们希望广大读者今后更加关心我们的刊物，给予更多的支持和帮助．在此，对所有关心、支持和爱护本刊的作者、读者表示深深的谢意!