



[12] 发明专利说明书

专利号 ZL 01139080.8

[45] 授权公告日 2007 年 7 月 25 日

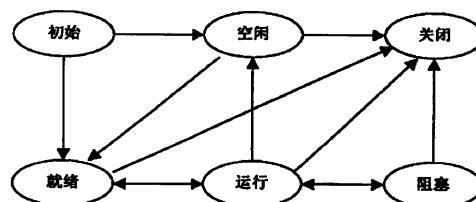
[11] 授权公告号 CN 1328877C

[22] 申请日 2001.12.3 [21] 申请号 01139080.8
 [73] 专利权人 中兴通讯股份有限公司
 地址 518057 深圳市南山区高新技术产业园科技南路中兴通讯大厦法律部
 [72] 发明人 谭 震
 [56] 参考文献
 US6058460A 2000.5.2
 US6223204B1 2001.4.24
 多线程的实现方法 张利霞, 河南师范大学学报, 第 29 卷第 2 期 2001
 多线程技术 李新明 李艺, 小型微型计算机系统, 第 19 卷第 2 期 1998
 审查员 彭 锐

权利要求书 2 页 说明书 8 页 附图 2 页

[54] 发明名称
 共享线程实现和调度方法

[57] 摘要
 一种涉及数据处理操作平台中线程的实现和调度方法, 涉及一种可避免或减少进行内核线程切换, 充分利用处理器效率, 并减少系统资源消耗的共享线程实现和调度技术。所述方法采用下列步骤: 以完成线程控制块的设置和线程绑定的线程创建、起动运行的线程激活、接受事件触发转入运行状态的线程运行、完成线程状态切换, 就绪及运行队列管理等的线程调度和完成线程数据的清除, 线程标识释放, 调整系统线程数量的线程的退出。本发明的方法, 充分减少了线程的切换时间; 并在不增加系统资源消耗的前提下, 充分利用系统资源, 提供数量巨大的线程机制, 以保证多线程编程在线程上的消耗。



1、一种共享线程实现和调度方法，其特征在于，共享线程实现和调度方法采用下列步骤：

(1) 线程创建，以完成线程控制块的设置和线程绑定，线程控制块的设置包括线程入口设置、事件接收区地址申请及线程状态初始化；

(2) 线程激活；

(3) 线程运行；以接受事件触发转入运行状态，并与系统线程绑定，对当前的事件处理结束后，重新转入空闲状态，并与系统线程分离。

(4) 线程调度，线程调度包括下列步骤：

a、线程系统维护就绪线程队列和运行线程队列；

b、由主线程和其他线程创建每一个线程；

c、当空闲线程有事件触发时，事件的发出方在发送事件的同时将接收线程从空闲状态转入就绪状态，并将该线程放入就绪线程队列；

d、系统线程阻塞等待在就绪线程队列上，一但有线程进入就绪线程队列，则系统线程逐个按次序绑定到就绪线程上，并将该线程转入运行线程队列；线程状态转为运行状态，随后控制权转入线程入口，同时输入触发此次运行的事件，线程运行；

e、当线程对当前事件处理完成后，查看其事件接收区是否有其他事件存在，如果有，则该线程从运行线程队列转入就绪线程队列；如果没有，该线程退出运行线程队列，其状态转换成空闲状态；

f、在每次线程状态的切换时，都伴随着系统线程数量的重新计算，根据当前线程总数，就绪线程数，运行线程数，空闲线程数，系统线程数，计算出是否需要增加或是减少系统线程的数量；

(5) 线程的退出；

线程的退出以完成线程数据的清除，线程标识的释放，并调整系统线程数量。

- 2、根据权利要求 1 所述的共享线程实现和调度方法，其特征在于，线程创建中线程入口设置包括：确定一个唯一线程的线程标识、线程入口地址的定义及线程入口参数的指定。
- 3、根据权利要求 1 所述的共享线程实现和调度方法，其特征在于，线程激活中线程的激活可以在线程创建的同时，也可以在线程创建后由调用者进行。
- 4、根据权利要求 1 所述的共享线程实现和调度方法，其特征在于，线程运行的触发事件可以是线程之间的消息、外界输入的信息或线程激活。

共享线程实现和调度方法

技术领域

本发明涉及一种数据处理操作平台中共享线程的实现和调度方法。具体的说,涉及一种可避免或减少进行内核的线程切换,充分利用处理器效率,并减少系统资源消耗的共享线程实现和调度方法。

背景技术

在典型的多任务操作系统中,系统的内存空间由硬件强制分成用户空间和内核空间,用户程序仅能运行在用户空间,当需要使用内核时,需要执行特殊的陷阱指令。传统的进程多任务系统就是基于这样的结构,进程控制结构的全部信息都处于内核空间,每次进程的切换都会涉及到对内核空间的访问。

多线程技术允许一个程序同时执行多个任务,并共享进程空间中的资源。与传统以进程为单位的任务操作系统不同,线程是轻量级实体,线程结构中相当一部分在用户空间,允许快速的访问。正是由于线程的这种特征,多线程技术得到了广泛的应用,编制多线程程序可以得到如下的好处:充分利用多处理器的并发能力而得到优秀的性能;提高程序的吞吐量和反应能力;减少进程间通讯开销,有效的利用系统资源;无需修改程序,即可以在单处理器或多处理器上使用。

线程是通过编写线程库实现的,目前有两种线程库实现方法。第一种方法是编写用户级库,依赖与已有的内核功能,最主要的库调用在用户空间执行;第二种方法是编写内核级库,大部分库调用运行于内核空间,需要系统调用支持。POSIX 标准的某些实现属于第一种方法,而 OS/2 和 Win32 线程属于第二种方法。由于操作系统内核只了解进程结构(后期存在轻量级进程 LWP),线程库的主要任务之一就是线程通过某种方法绑定到内核识别的进程或是 LWP 上,从而由

内核实现线程的并发。随着线程数量的增加，系统将会面临两个问题。一是，线程数量的增加将会消耗内核绑定进程(LWP)的数量，而这一内核资源是有限的；二是，线程数量的增加将导致线程切换次数的增加，每次线程的切换将可能导致内核的访问开销，切换花费的时间完全是额外的。鉴于上述的问题，在多线程的实际应用上存在着一定的限制，首先线程数量不能太大，其次，线程数量到达一定数量时，系统整体性能不能提高，甚至出现下降。同时，每一个线程都是一个独立的运行实体，但是在线程的整个生命周期中，线程并不是时时刻刻都在工作，大量的时间花费在等待可用资源上，这无疑是一种浪费。

发明内容

本发明的目的在于提出一种共享线程实现和调度方法，充分减少线程的切换时间；并在不增加系统资源消耗的前提下，充分利用系统资源，提供数量巨大的线程机制，以减少多线程编程在线程上的消耗。

本方法所涉及的线程分为两种，用户线程和系统线程，其中用户线程是本方法实现的线程，系统线程是由其他线程库提供的线程机制，用户线程构筑在系统线程之上。下文提及的线程均指用户线程。

线程的构成包括线程标识，线程入口和事件接收区几个主要部分，这几个部分由线程控制块管理，线程控制块处于用户空间。整个线程的实现和调度方法如下：

一种共享线程实现和调度方法，采用下列步骤：

1、线程创建，以完成线程控制块的设置和线程绑定，线程控制块的设置包括

线程入口设置、事件接收区地址申请及线程状态初始化；

2、线程激活；

3、线程运行：以接受事件触发转入运行状态，并与系统线程绑定，对当前

的事件处理结束后，重新转入空闲状态，并与系统线程分离。

4、线程调度，线程调度包括下列步骤：

- a、线程系统维护就绪线程队列和运行线程队列；
- b、由主线程和其他线程创建每一个线程；
- c、当空闲线程有事件触发时，事件的发出方在发送事件的同时将接收线程从空闲状态转入就绪状态，并将该线程放入就绪线程队列；
- d、系统线程阻塞等待在就绪线程队列上，一但有线程进入就绪线程队列，则系统线程逐个按次序绑定到就绪线程上，并将该线程转入运行线程队列；线程状态转为运行状态，随后控制权转入线程入口，同时输入触发此次运行的事件，线程运行；
- e、当线程对当前事件处理完成后，查看其事件接收区是否有其他事件存在，如果有，则该线程从运行线程队列转入就绪线程队列；如果没有，该线程退出运行线程队列，其状态转换成空闲状态；
- f、在每次线程状态的切换时，都伴随着系统线程数量的重新计算，根据当前线程总数，就绪线程数，运行线程数，空闲线程数，系统线程数，计算出是否需要增加或是减少系统线程的数量；

5、线程的退出：

线程的退出以完成线程数据的清除，线程标识的释放，并调整系统线程数量。

共享线程实现和调度方法的线程创建中，线程入口设置包括，确定一个唯一线程的线程标识、线程入口地址的定义及线程入口参数的指定。

共享线程实现和调度方法的线程激活中，线程的激活可以在线程创建的同时，也可以在线程创建后由调用者进行。

共享线程实现和调度方法的线程运行的触发事件可以是线程之间的消息、外界输入的信息或线程激活。

本发明提出的线程实现和调度方法应用于多线程系统中，具有以下有益效果：

- (1) 线程结构数据位于用户空间，线程的操作（创建，切换，退出）不涉及任何的内核空间的访问，速度快。
- (2) 线程数量不受到系统资源的限制，可以提供数量巨大的线程。
- (3) 只有运行状态的线程占用系统线程资源，并且线程在结束当前事件处理后会尽可能释放系统线程资源，使系统线程资源得到充分的利用。
- (4) 在线程的生命周期中，多个线程共享一个系统线程，使得系统线程数量相比线程数大大减少，从而减少系统线程的切换时间，提高系统运行效率。

附图说明

图 1 为本发明的线程控制块结构示意图；

图 2 为本发明的线程状态切换示意图；

图 3 为本发明的线程调度示意图；

图 4 为本发明的线程运行和调度流程示意图。

附图中 a 为空闲线程，b 为就绪线程，c 为运行线程，d 为阻塞线程，e 为系统线程。

具体实施方式

具体的讲整个线程的实现和调度方法为：

1. 线程的创建

线程的创建主要完成线程控制块的设置和线程绑定，线程控制块的设置包括线程入口的设置（线程入口是线程被调度时执行代码的起始点，可以使用指向该起始点的地址表示），线程入口的设置就是将该地址保存在线程控制块中；事件接收区申请与初始化（事件接收区为一内存缓冲区，它的申请就是获取一块可用的内存空间，并将其中的内容设置为初始状态），这些工作完全在用户空间完成，

不涉及任何内核空间的访问，因此效率是很高的。线程的绑定主要是根据当前线程使用状况，决定是否需要创建新的系统线程参与线程的调度。该操作对外提供相应的元语。

2、线程的激活

线程创建后处于初始状态，需要事件触发方可真正的工作，这种初始的触发称为线程的激活，线程在激活之前，不处理接收到的事件。线程的激活可以在线程创建的同时完成，也可以在线程创建后由调用者显式的进行。线程的激活是一次线程状态的切换操作（从初始状态切换至其他状态）。

3、线程的运行

线程的运行是由事件触发的，事件的种类可以是多种多样的，主要包括线程之间的消息，外界输入等，上面提到的线程激活也是一种事件。当有事件触发后，线程转入运行状态，并与系统线程绑定，对当前的事件处理结束后，重新转入空闲状态，并与系统线程分离。

4、线程的调度

线程的调度是本方法的主要部分，线程的调度主要完成如下几个功能：线程状态的切换，线程就绪线程队列的管理，线程运行线程队列的管理以及相应的入队出队操作。本方法提出的线程调度并不是由一个独立的实体完成，线程系统中所有的运行线程共同完成整个线程机制的调度工作。具体工作如下：

- (1) 线程系统维护数据结构：就绪线程队列数据结构和运行线程队列数据结构；

这两个数据结构的维护完全是数据的操作，包括向队列或池中增加一个数据，

或从队列和池中取出一个数据，其中队列维护进入队列数据的次序，采用先进先出的原则，而池则不需要维护这样的次序。

- (2) 每一个线程可由主线程（进程内固有的系统线程）和其他线程创建。线程的创建是通过线程创建元语实现的。

(3) 某个空闲线程有事件触发时，事件的发出方在发送事件（将事件放入接收线程的事件接收区）的同时，将接收线程从空闲状态转入就绪状态，并将该线程放入就绪线程队列。

(4) 系统线程阻塞等待在就绪线程队列上，一旦有线程进入就绪线程队列，

则系统线程逐个按次序绑定到就绪线程上，并将该线程转入运行线程队列，线程状态转换为运行状态，随后控制权转入线程入口，同时输入触发此次运行的事件，线程运行（对事件的处理）。

(5) 当线程对当前事件处理完成后，查看其事件接收区是否有其他事件存在，如果有，则该线程从运行线程队列转入就绪线程队列；如果没有，该线程退出运行线程队列，其状态转换成空闲状态。

(6) 在每次线程状态的切换时，都伴随着系统线程数量的重新计算，根据当前线程总数，就绪线程数，运行线程数，空闲线程数，系统线程数，计算出是否需要增加或是减少系统线程的数量。

5、线程的退出

线程的退出完成线程数据的清除，线程标识的释放，并调整系统线程数量。

如图 1 所示，显示了线程控制块的结构，包括线程标识，线程入口地址，线程入口参数，事件接收区地址，线程状态，线程属性以及其他线程数据；各部分说明如下：

线程标识用于唯一的确定一个线程，在线程入队操作可以代替线程实体，在线程间消息发送时可以用于寻址。线程标识必须在进程内唯一，可以通过统一的分配机制实现，线程退出后，标识释放允许重复使用。

线程入口地址用于在线程运行时控制权的传入，入口的形式可以自行定义。

线程入口参数用于传入线程在创建时指定的参数。

事件接收区地址用于存放触发线程运行的事件，该接收区维护事件到达的先后次序，保证先达事件先处理。

线程状态用于指定线程生命周期的不同阶段的行为，状态是可循环的，可切换的，具体描述见图 2 的说明。

线程的属性用于描述线程的一些固有性质，由线程创建时指定，在线程运行期间可以修改部分属性。

线程的其他数据，这部分可以根据具体的实现自行处理。

图 2 描述了线程在整个生命期拥有的各种状态及其转换。在线程的整个生命期中，线程的状态包括：初始状态，空闲状态，就绪状态，运行状态，阻塞状态，关闭状态。

初始状态：线程创建后即处于该状态，处于初始状态时，线程不被运行，只有线程激活操作才可以使线程离开初始状态。当激活时，线程的事件接收区内没有事件，状态转换为空闲状态；否则，当激活，并且线程的事件接收区有事件时，转换为就绪状态。

空闲状态：该状态表示线程的事件接收区内没有事件，该状态在有事件触发下转换为就绪状态；在有外部线程终止操作时，转换为关闭状态。

就绪状态：该状态表示线程在激活后，有事件触发，线程进入就绪线程队列等待运行。当线程到达队列头，并有系统线程绑定，则状态转换为运行状态；当在等待运行期间有外部线程终止操作，则转入关闭状态。

运行状态：当线程被系统线程绑定后，则进入运行状态。在运行结束后，如果没有事件处理，则转入空闲状态；否则转入就绪状态。

阻塞状态：当线程在运行期间调用阻塞操作时，将状态变换成阻塞状态，阻塞结束后，会继续进入运行状态，除非线程自行退出或是有外部终止，这将会使线程切换至关闭状态。

关闭状态：当线程在生命期内被终止或是退出，则切换至关闭状态。

图 3 显示了线程的调度和运行过程。具体如下：

- 线程创建时，处于初始状态，在激活后线程进入调度流程。如果当前没有事件触发，线程始终处于空闲状态（如图 3 中 a），与系统线程相分离（如图 3 中 e）。
- 当有事件触发某个空闲线程时，该空闲线程被挂入就绪线程队列（如图 3 中 b），排队等待绑定。
- 有一个或多个系统线程阻塞在就绪线程队列上（如图 3 中 d），当就绪线程队列中存在线程时，唤醒一个系统线程与其绑定，将其挂入运行线程队列，线程运行（如图 3 中 c）。
- 当线程处理完当前事件后返回，将其从运行线程队列中清除，根据其是否存在后续事件，决定切换其状态至空闲还是就绪，并与系统线程分离。
- 如果运行中的线程进行了阻塞调用，则其将被标记为阻塞，该标记在其被外部终止等操作中使用。
- 如果运行中的线程要求退出或是被外部终止，与其绑定的系统线程将根据线程的运行情况，实施分离和终止操作。

通过线程调度如图 3 所述，可以清楚的看到线程是如何运行的。首先，线程的调度是在每一个状态切换时进行，在线程运行期间是不会进行调度的，因此线程的调度是在用户级别进行的，不涉及到系统线程的切换，因此这种调度很安全。第二，线程的调度与事件的触发有着密切的关系，可以说在线程的调度是通过事件驱动的。第三，线程的调度涉及到线程与系统线程的绑定与分离，这个过程是动态进行的。

线程标识
线程入口地址
线程入口参数
事件接收区地址
线程状态
线程属性
线程其他数据

图 1

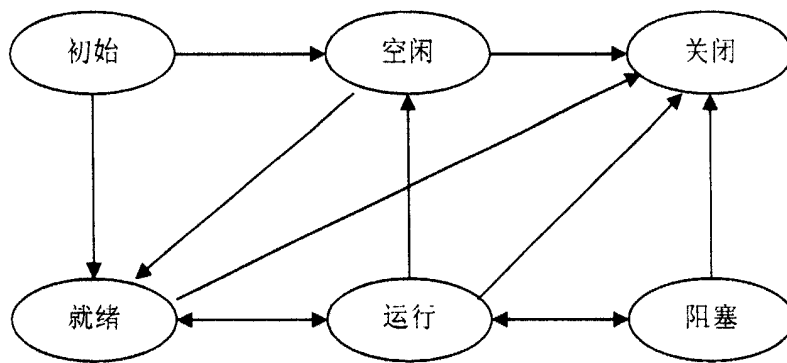
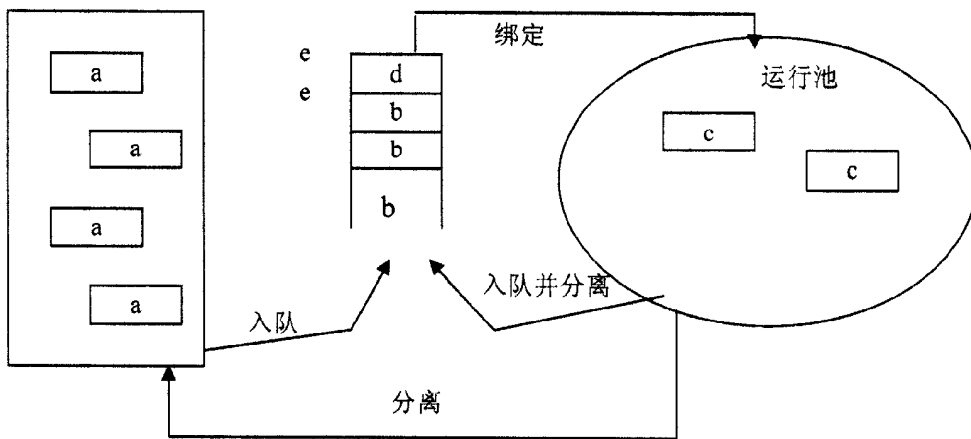


图 2



- 图中标号：
- a-- 空闲线程
 - b-- 就绪队列
 - c-- 运行线程
 - d-- 被阻塞的线程
 - e-- 系统线程

图 3

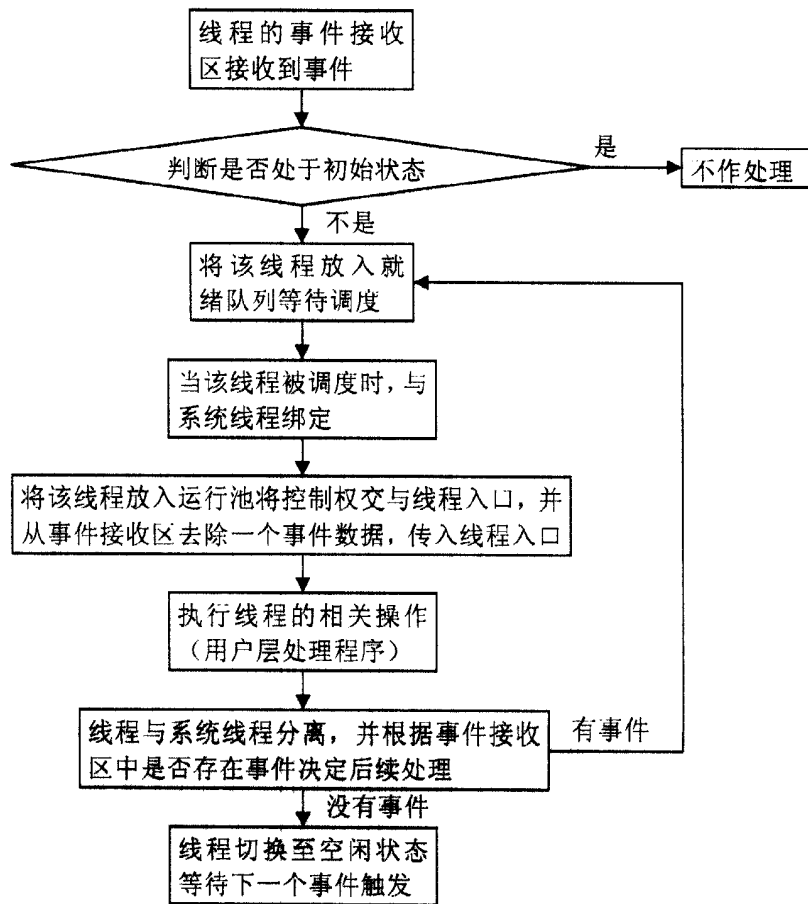


图 4