

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
C06F 9/48 (2006.01)



[12] 发明专利申请公开说明书

[21] 申请号 200610049137.1

[43] 公开日 2006 年 7 月 12 日

[11] 公开号 CN 1801101A

[22] 申请日 2006.1.17

[74] 专利代理机构 杭州求是专利事务所有限公司
代理人 林怀禹

[21] 申请号 200610049137.1

[71] 申请人 浙江大学

地址 310027 浙江省杭州市西湖区浙大路 38
号

[72] 发明人 陈天洲 戴鸿君 黄彧

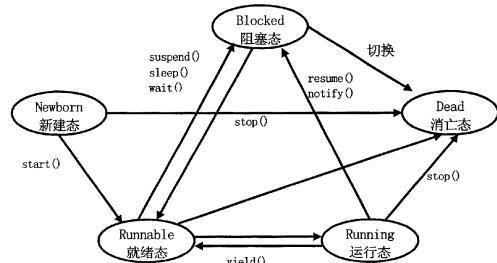
权利要求书 2 页 说明书 6 页 附图 1 页

[54] 发明名称

Java 操作系统中线程的实现和线程状态切换
的方法

[57] 摘要

本发明公开了一种 Java 操作系统中线程的实现和线程状态切换的方法。在 Java 操作系统中的线程是进程中的一条执行路径，是系统进行处理器调度的基本单位，同一个进程中的所有线程共享进程获得的地址空间和资源，线程具有 5 个不同的状态并且可以在状态间进行高效切换，这可以使系统性能获得很大提高。本发明对嵌入式系统环境，尤其是面向嵌入式系统的 Java 操作系统，有重大意义。



1、一种 Java 操作系统中线程的实现和线程状态切换的方法，其特征在于：

每个进程内允许包含多个并行执行的路径就是多线程，Java 操作系统中的线程是系统进行处理器调度的基本单位，同一个进程中的所有线程共享进程获得的地址空间和资源；一个线程具有：

- 一个线程执行状态，运行、就绪等状态
- 有一个受保护的线程上下文，当线程不运行时，用于存储现场信息
- 一个独立的程序指令计数器
- 一个执行堆栈
- 一个容纳局部变量的静态存储器

多线程进程的组成包括线程和空间，空间是完成一个程序的运行所需占有和管理的内存空间，它封装了对进程的管理，包括对指令代码、全局数据和 I/O 状态数据等共享部分的管理，Java 操作系统中的线程封装了并发，包括对 CPU 寄存器、执行栈和局部变量、过程调用参数、返回值等线程私有部分的管理，线程主动地访问空间；

Java 操作系统中的线程状态和状态切换，Java 操作系统中的线程从产生到消失，可分 5 个状态：

1) 新建态

线程在已被创建但尚未执行这段时间内，处于一个特殊的新建状态。这时，线程对象已被分配内存空间，其私有数据已被初始化，但该线程还未被调度。此时线程对象可通过 start () 方法调度，或者利用 stop () 方法杀死。新创建的线程一旦被调度，就将切换到就绪状态；

2) 就绪态

线程的就绪状态，表示线程正等待处理器资源，随时可被调用执行，处于就绪状态的线程事实上已被调度，已经被放到某一队列等待执行。处于就绪状态的线程何时可真正执行，取决于线程优先级以及队列的当前状况，线程的优先级如果相同，将遵循“先来先服务”的调度原则；

线程依据自身优先级进入等待对列的相应位置，某些系统线程具有最高优先级、这些最高优先级线程一旦进入就绪状态，将抢占当前正在执行的线程的处理器资源，当前线程只能重新在等待队列寻找自己的位置。某些具有最高优先级的线程执行完自己的任务之后，将调用 sleep ()、wait ()、suspend () 方

法，睡眠一段时间，等待被某一事件唤醒。一旦被唤醒，这些线程就又开始抢占处理器资源。这些最高优先级线程通常被用来执行一些关键性任务，如屏幕显示；

低优先级线程需等待更长的时间才能有机会运行，由于系统本身无法中止高优先级线程的执行，因此用到了优先级较高的线程对象，那么最好不时让这些线程放弃对处理器资源的控制权，以使其他线程能够有机会运行。

3) 运行态

线程已经拥有了对处理器的控制权，其代码目前正在运行。这个线程将一直运行直到运行完毕，除非运行过程的控制权被一优先级更高的线程强占，线程在如下3种情形之下将释放对处理器的控制权：

- 主动或被动地释放对处理器资源的控制权，该线程必须再次进入等待队列，等待其他优先级高或相等线程执行完毕
- 睡眠一段确定的时间，不进入等待队列，这段确定的时间到期后，重新开始运行
- 等待某一事件唤醒自己；

4) 阻塞态

暂时这个线程将无法进入就绪队列，处于堵塞状态的线程通常必须由某些事件才能唤醒。至于是何种事件，则取决于堵塞发生的原因，处于睡眠中的线程必须被堵塞一段固定的时间；被挂起、或处于消息等待状态的线程则必须由一外来事件唤醒；

5) 消亡态

表示线程已退出运行状态，并且不再进入就绪队列，其中原因可能是线程已执行完毕且正常结束，也可能是该线程被另一线程所强行中断。

Java 操作系统中线程的实现和线程状态切换的方法

技术领域

本发明涉及 Java 操作系统，尤其涉及一种 Java 操作系统中线程的实现和线程状态切换的方法。

背景技术

在传统的操作系统中，往往采用多进程并发程序设计来解决并行技术、网络技术和软件技术发展带来的要求，即创建并执行多个进程，按一定策略来调度和执行各个进程，以最大限度地利用计算机系统中的各种各样的资源。这一方式当然是可行的，但关键在于并行和并发的效率问题，采用这一方式来实现复杂的并发系统时，会出现以下的缺点：

- 进程切换的开销大，频繁的进程调度将耗费大量时间。
- 进程之间通信的代价大，每次通信均要涉及通信进程之间以及通信进程与操作系统之间的切换。
- 进程之间的并发性粒度较粗，并发度不高，过多的进程切换和通信使得细粒度的并发得不偿失。
- 不适合并行计算和分布并行计算的要求。对于多处理器和分布式的计算环境来说，进程之间大量频繁的通信和切换过程，会大大降低并行度。
- 不适合客户/服务器计算的要求。对于 C/S 结构来说，那些需要频繁输入输出并同时大量计算的服务器进程很难体现效率。

这要求新一代的操作系统改进进程结构，提供新的机制，使得很多应用能够按照需求，在同一进程中设计出多条的控制流，多控制流之间可以并行执行，切换不需通过进程调度；多控制流之间还可以通过内存区直接通信，降低通信开销。这就是多线程（结构）进程（Multiple Threaded Process）。

目前，很多著名的操作系统都支持多线程（结构）进程，如：Solaris 2.x、Mach 2.6、OS/2、Window NT、Windows 2000、Chorus 等。

发明内容

本发明的目的在于提供一种 Java 操作系统中线程的实现和线程状态切换的方法。

本发明解决其技术问题采用的技术方案如下：

每个进程内允许包含多个并行执行的路径就是多线程，Java 操作系统中的

线程是系统进行处理器调度的基本单位，同一个进程中的所有线程共享进程获得的地址空间和资源；一个线程具有：

- 一个线程执行状态，运行、就绪等状态
- 有一个受保护的线程上下文，当线程不运行时，用于存储现场信息
- 一个独立的程序指令计数器
- 一个执行堆栈
- 一个容纳局部变量的静态存储器

多线程进程的组成包括线程和空间。空间是完成一个程序的运行所需占有和管理的内存空间，它封装了对进程的管理，包括对指令代码、全局数据和 I/O 状态数据等共享部分的管理。Java 操作系统中的线程封装了并发，包括对 CPU 寄存器、执行栈和局部变量、过程调用参数、返回值等线程私有部分的管理。线程主动地访问空间。

Java 操作系统中的线程状态和状态切换，Java 操作系统中的线程从产生到消失，可分 5 个状态：

1) 新建态

线程在已被创建但尚未执行这段时间内，处于一个特殊的新建状态。这时，线程对象已被分配内存空间，其私有数据已被初始化，但该线程还未被调度。此时线程对象可通过 start() 方法调度，或者利用 stop() 方法杀死。新创建的线程一旦被调度，就将切换到就绪状态；

2) 就绪态

线程的就绪状态，表示线程正等待处理器资源，随时可被调用执行，处于就绪状态的线程事实上已被调度，已经被放到某一队列等待执行。处于就绪状态的线程何时可真正执行，取决于线程优先级以及队列的当前状况，线程的优先级如果相同，将遵循“先来先服务”的调度原则；

线程依据自身优先级进入等待对列的相应位置，某些系统线程具有最高优先级、这些最高优先级线程一旦进入就绪状态，将抢占当前正在执行的线程的处理器资源，当前线程只能重新在等待队列寻找自己的位置。某些具有最高优先级的线程执行完自己的任务之后，将调用 sleep()、wait()、suspend() 方法，睡眠一段时间，等待被某一事件唤醒。一旦被唤醒，这些线程就又开始抢占处理器资源。这些最高优先级线程通常被用来执行一些关键性任务，如屏幕显示；

低优先级线程需等待更长的时间才能有机会运行，由于系统本身无法中止

高优先级线程的执行，因此用到了优先级较高的线程对象，那么最好不时让这些线程放弃对处理器资源的控制权，以使其他线程能够有机会运行。

3) 运行态

线程的运行状态。该线程已经拥有了对处理器的控制权，其代码目前正在运行。这个线程将一直运行直到运行完毕，除非运行过程的控制权被一优先级更高的线程强占，线程在如下3种情形之下将释放对处理器的控制权：

- 主动或被动地释放对处理器资源的控制权，该线程必须再次进入等待队列，等待其他优先级高或相等线程执行完毕
- 睡眠一段确定的时间，不进入等待队列，这段确定的时间到期后，重新开始运行
- 等待某一事件唤醒自己；

4) 阻塞态

暂时这个线程将无法进入就绪队列，处于堵塞状态的线程通常必须由某些事件才能唤醒。至于是何种事件，则取决于堵塞发生的原因，处于睡眠中的线程必须被堵塞一段固定的时间；被挂起、或处于消息等待状态的线程则必须由一外来事件唤醒；

5) 消亡态

表示线程已退出运行状态，并且不再进入就绪队列，其中原因可能是线程已执行完毕且正常结束，也可能是该线程被另一线程所强行中断。

本发明具有的有益效果是：

(1) 快速线程切换。进程具有独立的虚地址空间，以进程为单位进行任务调度，系统必须交换地址空间，切换时间长，而在同一进程中的多线程共享同一地址空间，因而，能使线程快速切换。

(2) 减少系统管理开销。对多个进程的管理（创建、调度、终止等）系统开销大，如响应客户请求建立一个新的服务进程的服务器应用中，创建的开销比较显著。面对创建、终止线程，虽也有系统开销，但要比进程小得多。

(3) 线程通信易于实现。为了实现协作，进程或线程间需要交换数据。对于自动共享同一地址空间的各线程来说，所有全局数据均可自由访问，不需什么特殊设施就能实现数据共享。而进程通信则相当复杂，必须借助诸如通信机制、消息缓冲、管道机制等设施，而且必须调用内核功能，才能实现。

(4) 并行程度提高。许多多任务操作系统限制用户能拥有的最多进程数目，这对许多并发应用来说是不方便的。而对多线程技术来说，不存在线程数目的

限制。

(5) 节省内存空间。多线程合用进程地址空间，而不同进程独占地址空间，使用不经济。由于队列管理和处理器调度是以线程为单位的，在一个进程中的所有线程共享同一地址空间，多线程可以合用进程地址空间，因此可以节省内存空间。

附图说明

图1是线程的状态切换图；

图2是线程的构成示意图。

具体实施方法

每个进程中允许包含多个并行执行的路径就是多线程，如图 2 所示，Java 操作系统中的线程是系统进行处理器调度的基本单位，同一个进程中的所有线程共享进程获得的地址空间和资源；一个线程具有：

- 一个线程执行状态，运行、就绪等状态
- 有一个受保护的线程上下文，当线程不运行时，用于存储现场信息
- 一个独立的程序指令计数器
- 一个执行堆栈
- 一个容纳局部变量的静态存储器

多线程进程的组成包括线程和空间。空间是完成一个程序的运行所需占有和管理的内存空间，它封装了对进程的管理，包括对指令代码、全局数据和 I/O 状态数据等共享部分的管理。Java 操作系统中的线程封装了并发，包括对 CPU 寄存器、执行栈和局部变量、过程调用参数、返回值等线程私有部分的管理。线程主动地访问空间。

Java 操作系统中的线程状态和状态切换，Java 操作系统中的线程从产生到消失，可分 5 个状态，如图 1 所示：

1) 新建态

线程在已被创建但尚未执行这段时间内，处于一个特殊的新建状态。这时，线程对象已被分配内存空间，其私有数据已被初始化，但该线程还未被调度。此时线程对象可通过 start() 方法调度，或者利用 stop() 方法杀死。新创建的线程一旦被调度，就将切换到就绪状态；

2) 就绪态

线程的就绪状态，表示线程正等待处理器资源，随时可被调用执行，处于就绪状态的线程事实上已被调度，已经被放到某一队列等待执行。处于就绪状

态的线程何时可真正执行，取决于线程优先级以及队列的当前状况，线程的优先级如果相同，将遵循“先来先服务”的调度原则；

线程依据自身优先级进入等待对列的相应位置，某些系统线程具有最高优先级、这些最高优先级线程一旦进入就绪状态，将抢占当前正在执行的线程的处理器资源，当前线程只能重新在等待队列寻找自己的位置。某些具有最高优先级的线程执行完自己的任务之后，将调用 `sleep()`、`wait()`、`suspend()` 方法，睡眠一段时间，等待被某一事件唤醒。一旦被唤醒，这些线程就又开始抢占处理器资源。这些最高优先级线程通常被用来执行一些关键性任务，如屏幕显示；

低优先级线程需等待更长的时间才能有机会运行，由于系统本身无法中止高优先级线程的执行，因此用到了优先级较高的线程对象，那么最好不时让这些线程放弃对处理器资源的控制权，以使其他线程能够有机会运行。

3) 运行态

线程的运行状态。该线程已经拥有了对处理器的控制权，其代码目前正在运行。这个线程将一直运行直到运行完毕，除非运行过程的控制权被一优先级更高的线程强占，线程在如下 3 种情形之下将释放对处理器的控制权：

- 主动或被动地释放对处理器资源的控制权，该线程必须再次进入等待队列，等待其他优先级高或相等线程执行完毕
- 睡眠一段确定的时间，不进入等待队列，这段确定的时间到期后，重新开始运行
- 等待某一事件唤醒自己；

4) 阻塞态

暂时这个线程将无法进入就绪队列，处于堵塞状态的线程通常必须由某些事件才能唤醒。至于是何种事件，则取决于堵塞发生的原因，处于睡眠中的线程必须被堵塞一段固定的时间；被挂起、或处于消息等待状态的线程则必须由一外来事件唤醒；

5) 消亡态

表示线程已退出运行状态，并且不再进入就绪队列，其中原因可能是线程已执行完毕且正常结束，也可能是该线程被另一线程所强行中断。

JAVA 操作系统提供一个多线程系统，每个线程均被赋予一个优先级，优先级决定了线程获得 CPU 被调度执行的优先程度。优先级高的线程可在更短的时间段内获得更多的执行时间，优先级低的线程则正好相反。线程的优先级如果

完全相等，将被依据“先来先服务”的原则进行调度。

线程运行通过一个“调度程序”(scheduler)来进行调度。调度实际上就是分配处理器资源，处理器资源是按时间片分配的，获得处理器资源的线程只能在规定的时间片内执行，一旦时间片使用完毕，就必须把处理器让给另一处于等待状态的线程。

JAVA 操作系统使用“抢占式”(preemptive) 调度方式。为使低优先级线程能够有机会运行，较高优先级线程不时进入“睡眠”(sleep) 状态。进入睡眠状态的线程必须在被唤醒之后才能继续运行。

JAVA 操作系统的线程是从相同内存空间开始运行，共享类变量和方法的访问权，产生了同步问题。为了避免多个线程访问同一个资源，JAVA 操作系统提供了一种类似信号量的东西，即 monitor 来控制对象的访问同步。同步执行的代码段要访问某个共享对象，必须拥有该对象的 monitor。

本 Java 操作系统软件已经向中华人民共和国版权局申请计算机软件著作权登记。

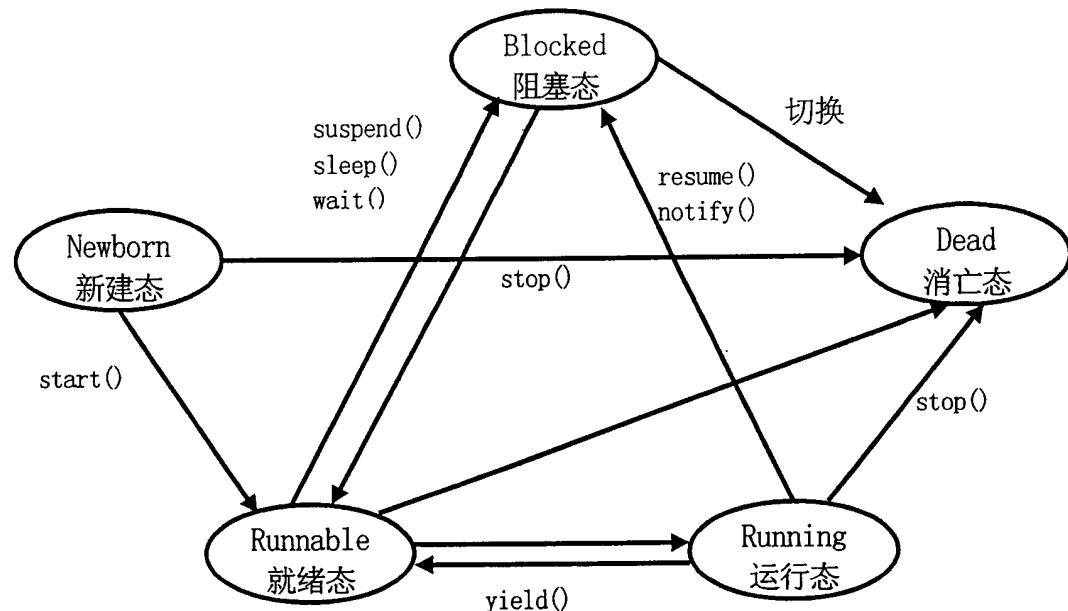


图 1

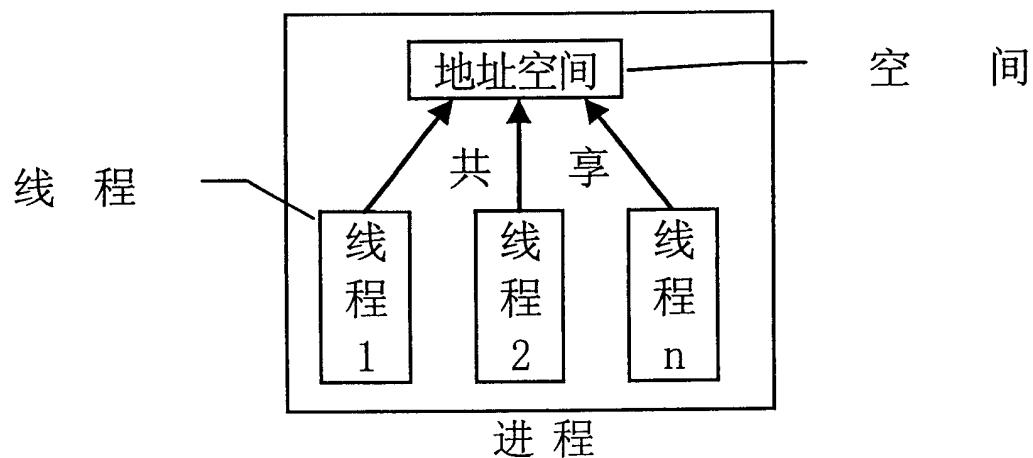


图 2