[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)

[12] 发明专利说明书

专利号 ZL 200610142390.1

[45] 授权公告日 2009 年 12 月 16 日

[11] 授权公告号 CN 100570564C

[22] 申请日 2006.10.11
[21] 申请号 200610142390.1
[30] 优先权
[32] 2005.12.1 [33] US [31] 11/291,600
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 宋正中 柯文正
丹尼尔·A.·赫弗利
[56] 参考文献
CN1627264A 2005.6.15
WO2005/022386A2 2005.3.10
US6401155B1 2002.6.4
CN1592890A 2005.3.9
US5944816A 1999.8.31
WO2004/036354A2 2004.4.29
审查员 刘宇儒
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 付建军

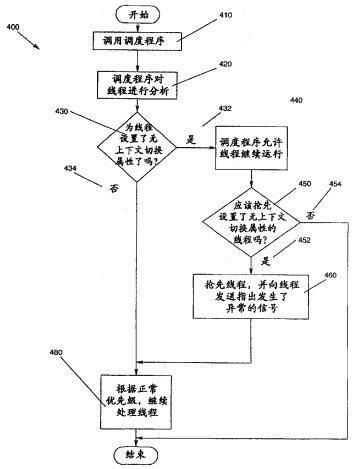
权利要求书 2 页 说明书 10 页 附图 5 页

[54] 发明名称

将用户模式线程配置为接近禁止中断优先级的方法和系统

[57] 摘要

说明了用于在多线程处理器系统中将用户模式线程配置为接近禁止中断优先级的方法和相应计算机系统，其中提供了无上下文开关属性，该属性允许用户模式线程变为接近禁止中断优先级。 数据包括无上下文切换属性。 基于无上下文切换属性的线程的控制比实时优先级更有效，因为无上下文切换属性回避了调度的开销。 此外，无上下文切换属性可以用来检测线程是否执行可能使线程在关键部分时被挂起的任何不希望的操作。 无上下文切换属性可配置为指示线程的执行是否可被挂起。



1. 一种计算机系统，包括

一个或多个处理器，存储设备，总线，总线接口，和输入/输出设备；

其中所述处理器在多线程进程内执行线程；

用于向所述计算机系统分配系统资源的操作系统驻留在所述存储设备内，该操作系统执行应用程序所需的进程；

由所述操作系统控制的调度程序用于协调线程的执行并管理对系统资源的访问；以及

所述线程提供可执行的指令，其中所述线程包括无上下文切换属性，无上下文切换属性可配置为指出该线程的执行是否可被调度程序挂起。

2. 根据权利要求1所述的计算机系统，其中，当设置了无上下文切换属性时，线程被允许继续而不会被基于优先级调度而挂起。

3. 根据权利要求1所述的计算机系统，其中，无上下文切换属性是通过系统调用设置的。

4. 根据权利要求1所述的计算机系统，其中，调度程序确保了线程在相对于其指定的优先级和类的时间内获得处理器的访问，并且防止线程对处理器的访问发生饥饿。

5. 根据权利要求1所述的计算机系统，其中，调度程序抢先地删除线程，使其不被处理。

6. 根据权利要求5所述的计算机系统，其中，调度程序在分配给线程的时间量届满时抢先地从处理器中删除线程。

7. 根据权利要求1所述的计算机系统，其中，调度程序选择执行线程的顺序。

8. 根据权利要求1所述的计算机系统，其中，当调用调度程序时，调度程序检查无上下文切换属性。

9. 根据权利要求8所述的计算机系统，其中，当设置了无上下

文切换属性时，调度程序绕过所有调度进程，并允许线程继续。

10. 根据权利要求1所述的计算机系统，其中，当线程被迫挂起并且为线程设置了无上下文切换属性时，调度程序清除无上下文切换属性，并向线程发送信号以指出发生了异常。

11. 根据权利要求10所述的计算机系统，其中，当为线程设置了无上下文切换属性时，只有中断处理程序才迫使线程挂起。

12. 根据权利要求11所述的计算机系统，其中，当中断处理程序退出时，控制返回到被中断的线程。

13. 一种用于在多线程处理器系统中将用户模式线程配置为接近禁止中断优先级的方法，包括以下操作：

提供用于提供可由操作系统执行的指令的线程；以及

为线程提供无上下文切换属性，无上下文切换属性可配置为指出该线程的执行是否可被调度程序挂起。

14. 根据权利要求13所述的方法，进一步包括使用系统调用为线程设置无上下文切换属性。

15. 根据权利要求13所述的方法，进一步包括，当设置了无上下文切换属性时，对无上下文切换属性进行分析，以判断是否允许该线程继续而不会被基于优先级调度而挂起，并允许线程继续。

16. 根据权利要求13所述的方法，进一步包括，当线程被迫挂起并且为线程设置了无上下文切换属性时，清除无上下文切换属性，并向线程发送信号以指示发生了异常。

将用户模式线程配置为接近禁止中断优先级的方法和系统

技术领域

一般而言，本发明涉及计算机操作系统，具体来说，涉及用于在多线程处理器系统中将用户模式线程配置为接近禁止中断优先级的方法和相应计算机系统。

背景技术

在计算机技术中，操作系统 (OS) 是负责对硬件和基本系统操作进行直接控制和管理的系统软件。另外，它还提供了诸如文字处理程序和 Web 浏览器之类的应用程序软件运行所依赖的基础。许多现代操作系统都支持多重任务处理，通过该功能，多个线程共享诸如处理器之类的共同的处理资源。计算机科学中的线程是执行的线程或指令序列的简称。线程基本上是在同一个存储器上下文中运行的进程。多个线程可以并行地在许多计算机系统上执行。这种多线程处理一般是通过时间分片进行的（其中，单个处理器在不同的线程之间切换）或通过多重处理进行的（其中，线程在单独的处理器上执行）。线程类似于进程，但在它们共享资源的方式方面不同。

在计算机具有单一处理器的情况下，如果说只有一个线程在任何时间点正在运行，就是说，处理器正在积极地执行该线程的指令。多任务处理通过安排哪一个线程可以在任何给定时间运行以及何时该轮到另一个正在等待的线程来解决问题。将处理器从一个线程重新分配到另一个线程的动作叫做“上下文切换”。每当一个进程从对处理器的访问离开时，必须存储足够的有关其当前操作状态的信息，以便当它被再次安排到在处理器上运行时，它可以从相同的位置恢复其操作。此操作状态数据被称为“其上下文”，将执行的进程的线程从处理器中删除的动作（以及用另一个替换它）被称为“进程切换”或上下文切换。当足够频繁地发生上下文切换时，会出现并行性的错觉。甚至

在具有一个以上的处理器的计算机上，多任务处理也允许比处理器的数量更多的任务运行。

操作系统对系统进行管理，并为该系统运行第三方应用程序软件。如此，通常的理解是，操作系统不仅包括直接与硬件进行交互的低级别的“内核”，而且还包括应用程序以及基本程序对文件进行操纵和对系统进行配置所需库。操作系统内核允许程序员通过系统调用接口对线程进行操纵。

操作系统可以采用许多不同调度策略中的一个，一般属于下列类别中的一个类别。在多程序系统中，运行的任务一直运行，直到它执行需要等待外部事件（例如从磁带读取）的操作。多程序系统能够最大化处理器使用率，在时间共享系统中，正在运行的任务需要放弃处理器，要么是自愿地，要么通过诸如硬件中断之类的外部事件。时间共享系统能够允许多个程序显然同时执行。在实时系统中，可保证当发生外部事件时，给予某些正在等待的任务处理器。

操作系统的体系结构包括两个主要层：用户模式和内核模式。用户模式下的程序和子系统在它们可以访问哪些系统资源方面受到限制，而内核模式的对系统存储器和外部设备的访问没有限制。用户模式是非特惠的状态，其中，硬件禁止正在执行的代码执行某些可能会使系统不稳定或造成安全漏洞的操作（如写入到没有分配给它的存储器）。

当今的操作系统，例如，**Linux**、**Windows**、**Unix**，支持在用户模式下运行的程序和在内核模式下运行的程序。例如，所有的 **Windows 2000** 子系统和应用程序都在用户模式下运行，而每一个子系统和应用程序都在其自己的受保护的地址空间中运行。如此，一个进程始终要么在用户模式下运行，要么在内核模式下运行。用户程序的主体在用户模式下执行，而系统调用在内核模式下执行。内核模式包括用于给受保护的存储器模式提供完全权限的代码。用户模式包括访问其自己的存储区的权限。用户应用程序和环境子系统在此模式下执行。此外，用户模式是某些操作系统（例如，**Linux**、**Windows** 等

等）中的让用户在更大的操作系统内安装操作系统的单独的实例的机制。如此，用户模式让用户在计算机内创建虚拟机，从而提供了创建测试情况的能力，无需冒险在主系统内产生问题。

在用户模式下操作有多个优点。在用户模式下操作可使工程师无需单独的测试机器即可对代码进行调试，安装软件内核（主存储器和软件的处理区域）的不同版本，培训新用户。用户模式还可以隔离在其内部运行的进程，以便可以防止病毒或其他恶意代码感染更大的机器，从而节省未计划的重新安装和重建的时间。用户模式还有助于测试新网络配置和创建灾难恢复操练，因为它将主要安装与非故意的或故意的损坏隔离开来。

在 1998 年 2 月成立了开放源代码倡议 (OSI)。OSI 希望使用“开放源代码”这一标签将消除歧义，特别是对于感觉“自由软件”是反商业的那些人。如此，OSI 试图使较高的 profile 产生自由地可用的源代码的实际好处，并希望使主要的软件企业及其他高科技行业加入到开放源代码倡议中来。开放源代码软件将计算机软件和其源代码的可用性称为根据研究、改变，并改进其设计的开放源代码许可证的开放源代码。如此，开放源代码是指其中的源代码自由地可供其他人查看、修改和改编的软件。通常，它是由跨机构和国界的开发人员的团队创建和维护的。如此，开放源代码软件不能被一个大型专有的供应商据为己有。

Linux 是自由软件和开放源代码开发的最著名的示例之一：与诸如 Windows 和 Mac OS 之类的专有的操作系统不同，所有其基础源代码都对公众可用，任何人都可以自由地使用、修改和重新分发它。从狭义上来讲，术语 Linux 是指 Linux 内核，但它通常用于描述基于 Linux 内核的，并与来自 GNU 项目的库和工具及其他源代码相结合的全部类似于 Unix 操作系统（也称为 GNU/Linux）。从广义上来讲，Linux 分发将大量的应用程序软件与核心系统捆绑起来，并提供对用户更友好的安装和升级。但是 Linux（或更准确地说 GNU/Linux）只是冰山一角。当今有各式各样的开放源代码软件可用，

一直有新项目启动。

随着开放源代码内核（例如 **Linux**）越来越流行，越来越多的控制器程序正在移向用户模式。然而，控制器程序不断地以实时的方式对硬件进行访问，因此需要允许用户模式线程在接近禁止中断优先级运行。某些实时操作系统提供特殊的实时优先级。然而，为提供灵活性，它们仍可能会在相同的实时优先级与其他线程共享优先级。

可以看出，需要用于提供允许用于提供无上下文切换属性（该属性允许用户模式线程变为接近禁止中断优先级）的方法、设备和程序存储设备。

发明内容

为克服上文所描述的现有技术中的限制，并克服在阅读和理解本说明书之后将变得显而易见的其他限制，本发明说明了用于在多线程处理器系统中将用户模式线程配置为接近禁止中断优先级的方法和相应计算机系统，其中提供了无上下文开关属性（该属性允许用户模式线程变为接近禁止中断优先级）。

本发明通过提供无上下文切换属性来解决上文所描述的问题，该属性比实时优先级更有效，因为无上下文切换属性回避了调度的开销。此外，无上下文切换属性可以用来检测线程是否执行可能使线程在关键部分时被挂起的任何不希望的操作。

在本发明的一个实施例中，提供了计算机系统。该计算机系统包括一个或多个处理器，存储设备，总线，总线接口，和输入/输出设备；其中所述处理器在多线程进程内执行线程；用于向所述计算机系统分配系统资源的操作系统驻留在所述存储设备内，该操作系统执行应用程序所需的进程；

由所述操作系统控制的调度程序用于协调线程的执行并管理对系统资源的访问；以及

所述线程提供可执行的指令，其中所述线程包括无上下文切换属性，无上下文切换属性可配置为指出该线程的执行是否可被调度程序挂起。

在本发明的另一个实施例中，还提供了一种用于在多线程处理器系统中将用户模式线程配置为接近禁止中断优先级的方法。该方法包括以下操作：

提供用于提供可由操作系统执行的指令的线程；以及

为线程提供无上下文切换属性，无上下文切换属性可配置为指出该线程的执行是否可被调度程序挂起。

为了更好地理解本发明及其在使用中实现的优点和目的，应该结合同样作为本申请文件一部分的附图以及关于附图描述性内容，在这部分描述性内容中，说明并描述了根据本发明的设备的特定实施例。

附图说明

现在请参看附图，在附图中，类似的附图标记表示对应的部件：

图 1 显示了根据本发明的实施例的计算机系统的方框图；

图 2 显示了根据本发明的实施例的执行多线程处理的计算机系统；

图 3 显示了根据本发明的实施例的操作系统的运转；

图 4 是根据本发明的实施例的使用具有无上下文切换属性的线程的方法的流程图；以及

图 5 显示了根据本发明的实施例的系统。

具体实施方式

在下面对实施例进行的描述中，参考了构成了本发明的组成部分的附图，在附图中，通过例图，显示了其中可以实施本发明的特定实施例。应该理解，也可以利用其他实施例，因为在不偏离本发明的范围的情况下，可以进行结构更改。

本发明提供了用于提供无上下文切换的属性的方法、设备和程序存储设备，该属性允许用户模式线程变为接近禁止中断优先级。无上下文切换属性比实时优先级更有效，因为无上下文切换属性回避了调度的开销。此外，无上下文切换属性可以用来检测线程是否执行可能

使线程在关键部分时被挂起的任何不希望的操作。

图 1 显示了根据本发明的实施例的计算机系统 100 的方框图。在图 1 中，计算机系统 100 包括一个或多个处理器 130，每一个处理器都能够在许多并行多线程进程中的一个进程内执行线程。如在多重任务数据处理系统中的典型情况，可以给每一个用户进程分配其自己的虚拟存储器空间，该空间可以由存储器管理器 136 部分地映射到高速主存储器 132，并部分地映射到较低速度辅助存储器 134。

计算机系统 100 和系统资源向计算机系统 100 的分配是由操作系统 138 进行控制的。为便于本讨论，假设操作系统 138 驻留在主存储器 132 内，虽然那些精通相关技术的人将认识到，操作系统 138 的某些不经常使用的段可以由存储器管理器 136 转储到辅助存储器 134。操作系统 138 包括内核 140，该内核包括操作系统 138 的最直接与计算机系统 100 进行交互的最低层。内核 140 将内核线程调度到处理器 130 供其执行，向与计算机系统 100 内的硬件连接的设备驱动程序提供服务，并实现由计算机系统 100 使用的系统服务、存储器管理、网络访问以及文件系统。除了内核 140 之外，主存储器 132 还存储了应用程序软件 142 的频繁使用的段。如那些精通本技术的人所知道的，应用程序软件 142 通过应用程序编程接口 (API) 与操作系统 138 进行通信。

计算机系统 100 还包括总线接口 144，通过该总线接口，多个节点可以连接到计算机系统 100 内可用的系统资源。如那些精通本技术的人所理解的，计算机系统 100 还可以包括连接到系统总线 146 的对于理解本发明不需要的额外的硬件，相应地，为简单起见，省略这些硬件。

图 2 显示了根据本发明的实施例的执行多线程处理的计算机系统 200。在图 2 中，总线 218 将一个处理器 212 连接到各种 I/O 设备 214 和存储设备 216。存储设备 216 包括线程集合 220，每一个线程与特定进程关联。每一个线程都包括数据，例如，执行堆栈 224 和无上下文切换属性。当由处理器 212 执行线程时，线程值被加载

到处理器寄存器中。当线程被挂起时，值被保存回存储器 216 中。存储设备 216 进一步包括由进程所使用的数据和指令的所有逻辑地址，包括各种线程的堆栈。在创建线程之后并在终止之前，线程将最有可能利用系统资源来访问进程上下文 222。通过进程上下文 222，进程线程可以共享数据，并以简便易行和直接的方式彼此进行通信。

线程调度是实现线程的一个重要方面。在第一种类别中，有协作线程，它们不依赖调度程序并在它们本身之间合作，以共享处理器。由于编制这样的线程的复杂性，并且因为它们不能快速对外部事件作出反应，协作线程在当今的技术中使用得不太多。第二种类别是抢先式线程。这样的线程依赖于调度程序，该调度程序可以在执行过程中的任一点决定将处理器从一个线程切换到另一个线程。抢先式线程快速对外部事件作出反应，因为在需要时另一个线程可用比当前正在运行的线程先抢先处理器用接管来处理紧急情况。与协作线程不同，抢先式调度免除了程序员在应用程序内实现调度机制的负担。

正在运行的线程取得优先权一般可以在程序执行过程中的任一点发生。通常，当允许调度程序干涉并在线程之间切换处理器的计时器届满时，发生抢先的情况。在技术界，这被称为在线程之间对处理器进行“时间分片”，每一个线程在“时间片”内运行。 这种干涉形式允许调度程序实现各种调度机制，包括循环复用、优先级调度等等。另外，还可以响应可能需要对某些线程进行及时处理的外部事件发生抢先。

如上文所描述的，开放源代码内核（例如 Linux）越来越流行，导致越来越多的控制器程序正在移向用户模式。然而，控制器程序不断地以实时的方式对硬件进行访问，因此需要允许用户模式线程在接近禁止中断优先级运行。某些实时操作系统提供特殊的实时优先级。然而，为提供灵活性，它们仍可能会在相同的实时优先级与其他线程共享优先级。

图 3 显示了根据本发明的实施例的操作系统 300 的运转。在图 3 中，操作系统包括调度程序 320。操作系统 300 执行线程 310、

312、314 以执行用户模式应用程序所需的进程。操作系统提供了调度程序 320，以协调线程的执行并管理它们的对系统资源的访问。调度程序 320 必须确保线程在相对于其指定的优先级和类的时间内能够访问处理器，并且没有线程对处理器的访问发生饥饿，不管它是否是可用的最低优先级任务。调度程序 320 可以抢先地从处理器中删除线程，例如，在实现了时间分片调度时，在给它分配的时间量届满时。调度程序 320 选择哪一个是接下来运行的最合适的进程。

根据本发明的实施例，给线程 310、312、314 提供了无上下文切换属性 330、332 334 以控制是否要抢先线程。提供了系统调用 340，以允许设置和/或清除无上下文切换线程属性 330、332、334。无上下文切换属性 330、332 334 是每个线程都有的，即，每一个线程 310、312、314 都包括无上下文切换属性 330、332、334。调度程序 320 在调用了调度程序 320 的任何时间都检查无上下文切换属性 330、332、334。当设置了无上下文切换属性 330、332、334 时，调度程序 320 将绕过所有调度进程，只允许线程继续。如果在设置了上下文切换属性 330、332、334 的情况下线程 310、312、314 中的某一个被迫挂起，则调度程序 320 将清除无上下文切换属性 330、332、334，并向线程 310、312、314 发送信号，以指出发生了异常。当无上下文切换属性 330、332、334 是设置的属性时，线程 310、312、314 将继续运行，不管优先级和时间片如何。只有中断处理程序 350 才能抢先 352 设置了其无上下文切换属性 330、332、334 的线程 310、312、314。当中断处理程序 350 退出时，控制立刻回到设置了无上下文切换属性 330、332、334 的被中断的线程 310、312、314。

图 4 是根据本发明的实施例的使用具有无上下文切换属性的线程的方法的流程图。在图 4 中，调用了调度程序 410。调度程序对线程进行分析 420。就是否为线程设置了无上下文切换属性作出判断 430。如果设置了无上下文切换属性 432 ，则调度程序允许线程继续运行 440。判断是否要抢先设置了无上下文切换属性的线程 450。如

果是 452，则抢先线程，并向线程发送指出发生了异常的信号 460。被抢先的线程开始再次执行 480。如果将不抢先线程 454，则退出调度程序，不进行上下文切换。如果没有设置无上下文切换属性 434，即，清除，则调度程序根据普通优先级对线程进行处理 480。

图 5 显示了根据本发明的实施例的系统 500。本发明的实施例可以呈现完全是硬件、完全是软件、或结合了软件和硬件元件的组合的形式。在优选实施例中，本发明是以软件实现的，包括，但不仅限于固件、常驻软件、微代码等等。此外，本发明的实施例可以呈现可以从计算机可使用的或计算机可读的介质 568 访问的计算机程序产品 590 的形式，提供供计算机或任何指令执行系统使用的或与它们一起使用的程序代码。

对于此描述，计算机可使用的或计算机可读介质 568 可以是任何设备，可以包含、存储、通信、传播或输送供指令执行系统、设备使用的，或与它们一起使用的程序。介质 568 可以是电子、磁性、光学、电磁、红外线或半导体系统（或设备）或传播介质。计算机可读的介质的示例包括半导体或固态存储器、磁带、可移动计算机磁盘、随机存取存储器 (RAM)、只读存储器 (ROM)、硬磁盘和光盘。光盘的当前示例包括光盘-只读存储器 (CD-ROM)、光盘-读取/写入 (CD-R/W) 和 DVD。

适用于存储和/或执行程序代码的系统将包括通过系统总线 520 直接或间接地连接到存储器元件 592 的至少一个处理器 596。存储器元件 592 可以包括在程序代码的实际执行过程中使用的本地存储器、大容量存储器、高速缓存存储器，这种高速缓存存储器提供用于存储至少某些程序代码的临时存储器，以便减少在执行过程中必须从大容量存储器中检索代码的次数。

输入/输出或 I/O 设备 540（包括但不仅限于：键盘、显示器、指示设备等等）可以直接或者通过干涉 I/O 控制器连接到系统。

也可以将网络适配器 550 连接到系统，以使系统通过干涉专用或公共网络 560 连接到其他数据处理系统 552，远程打印机 554 或

存储设备 556。调制解调器、电缆调制解调器和以太网卡是现行的几种网络适配器。

相应地，计算机程序 590 包括指令，当由图 5 的系统 500 读取和执行指令时，使系统 500 执行为了执行本发明的步骤或元素所需的步骤。

本发明的实施例的上述描述只是为了说明和描述。它没有穷尽一切，也不将本发明限制到所述准确的形式。根据上文的讲述，许多修改和变化也是可以的。本发明的范围不是由此详细说明进行限制，而是由所附的权利要求进行限制。

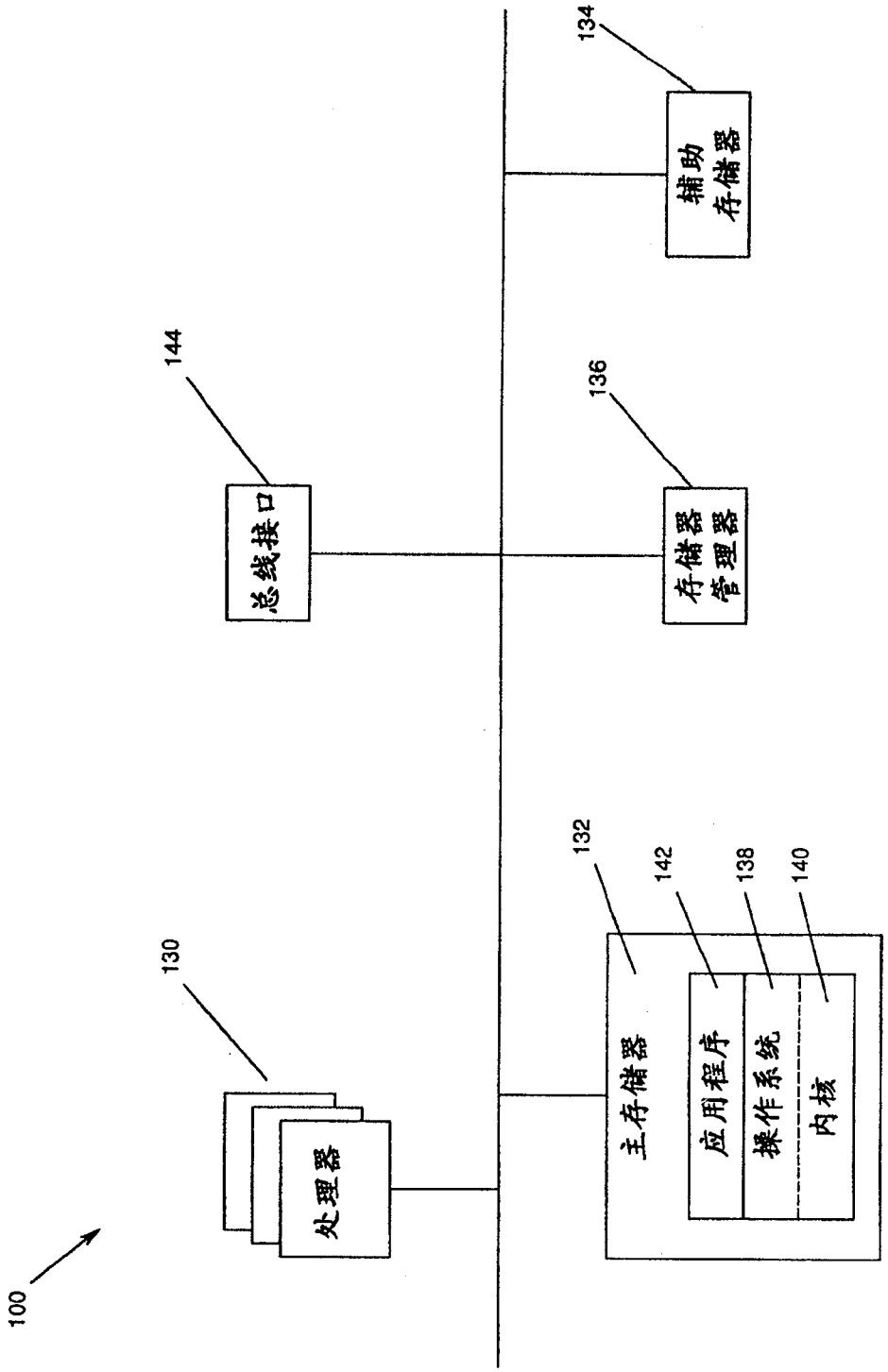

100
处理器
130
主存储器
132
应用程序
142
操作系统
138
内核
140
总线接口
144
存储器管理器
136
辅助存储器
134

图 1

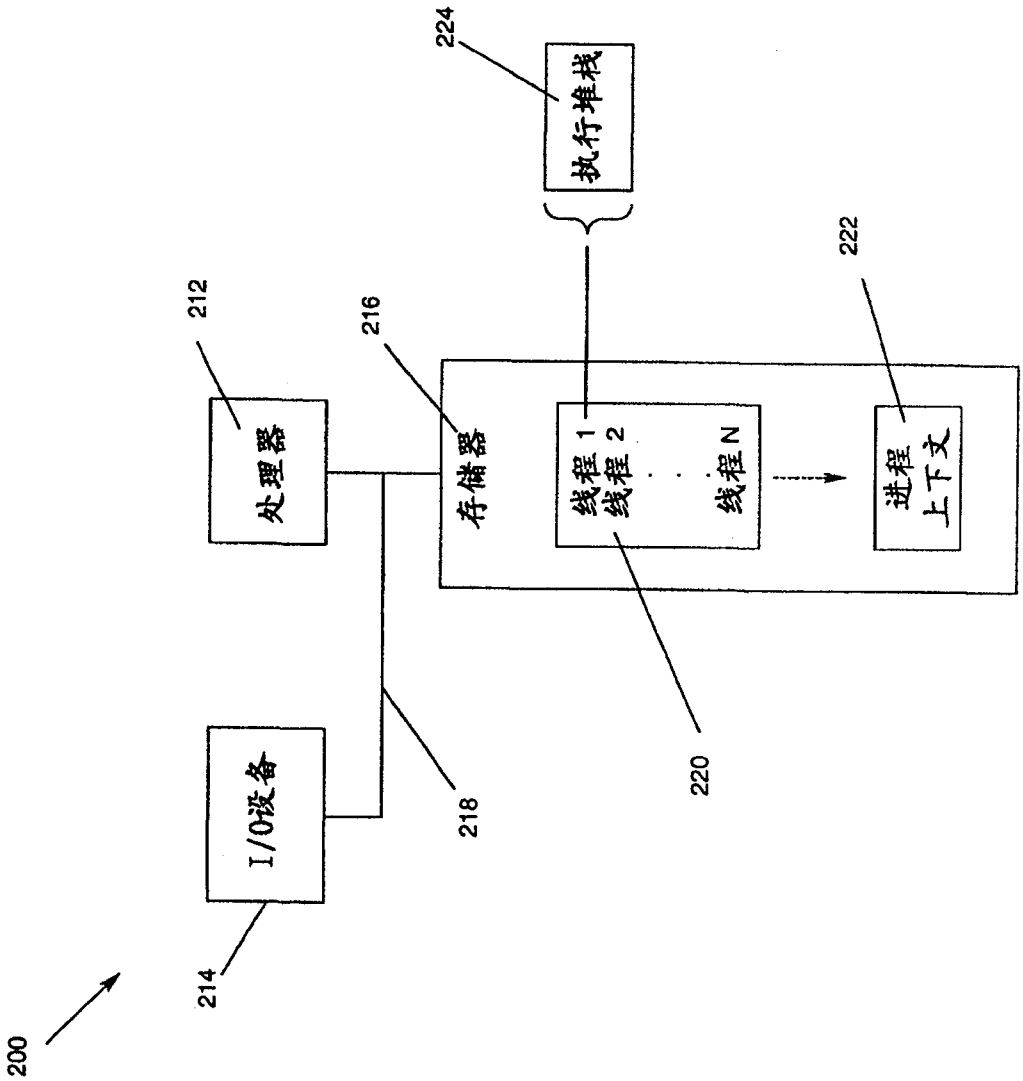

200
214
I/O设备
218
处理器
212
216
存储器
线程 1
线程 2
线程 N
220
进程
上下文
222
执行堆栈
224

图 2

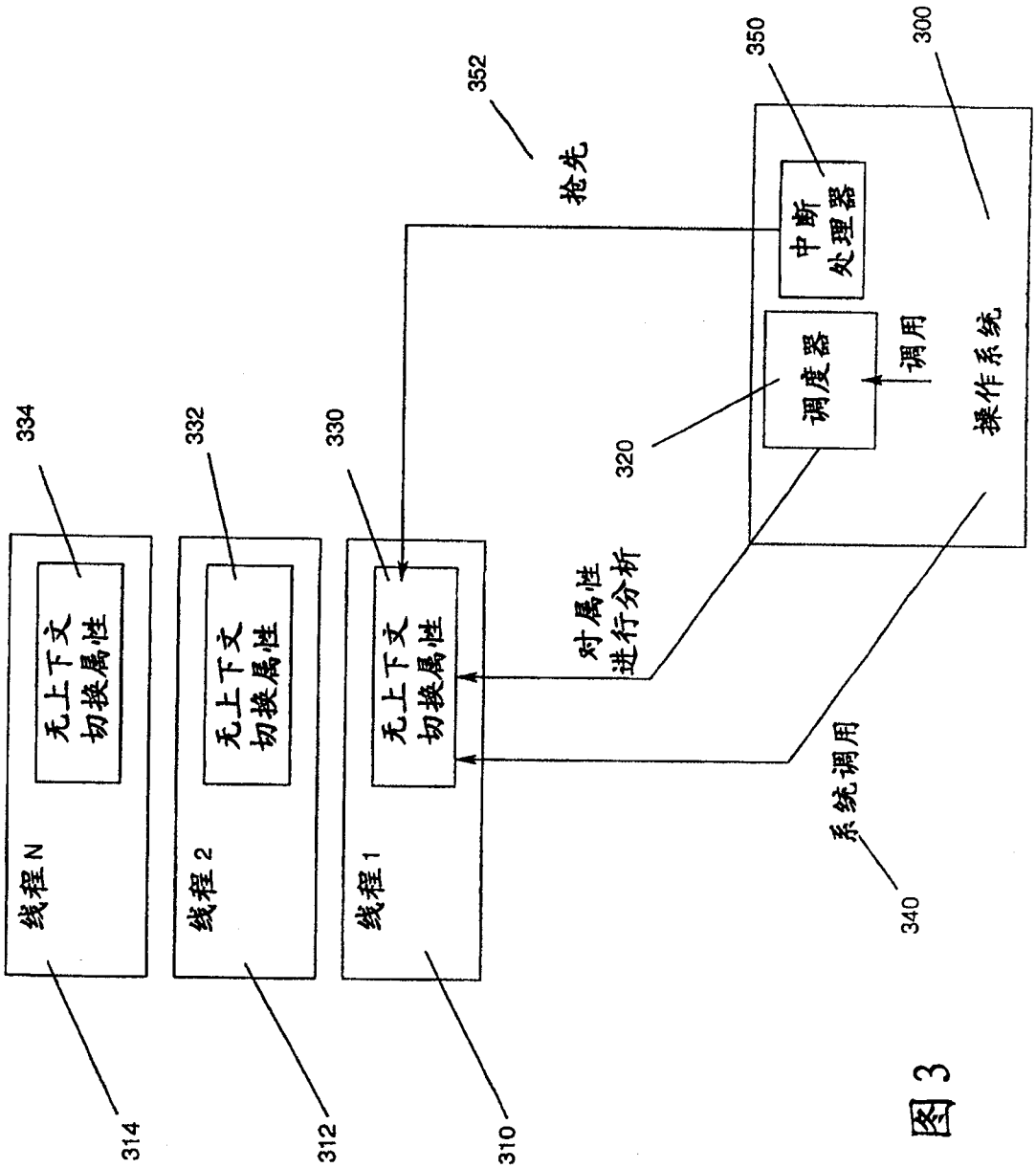

线程 N
无上下文切换属性
线程 2
无上下文切换属性
线程 1
无上下文切换属性
334
332
330
314
312
310
对属性进行分析
352
抢先
350
中断处理器
300
调度器
320
调用
操作系统
系统调用
340

图 3

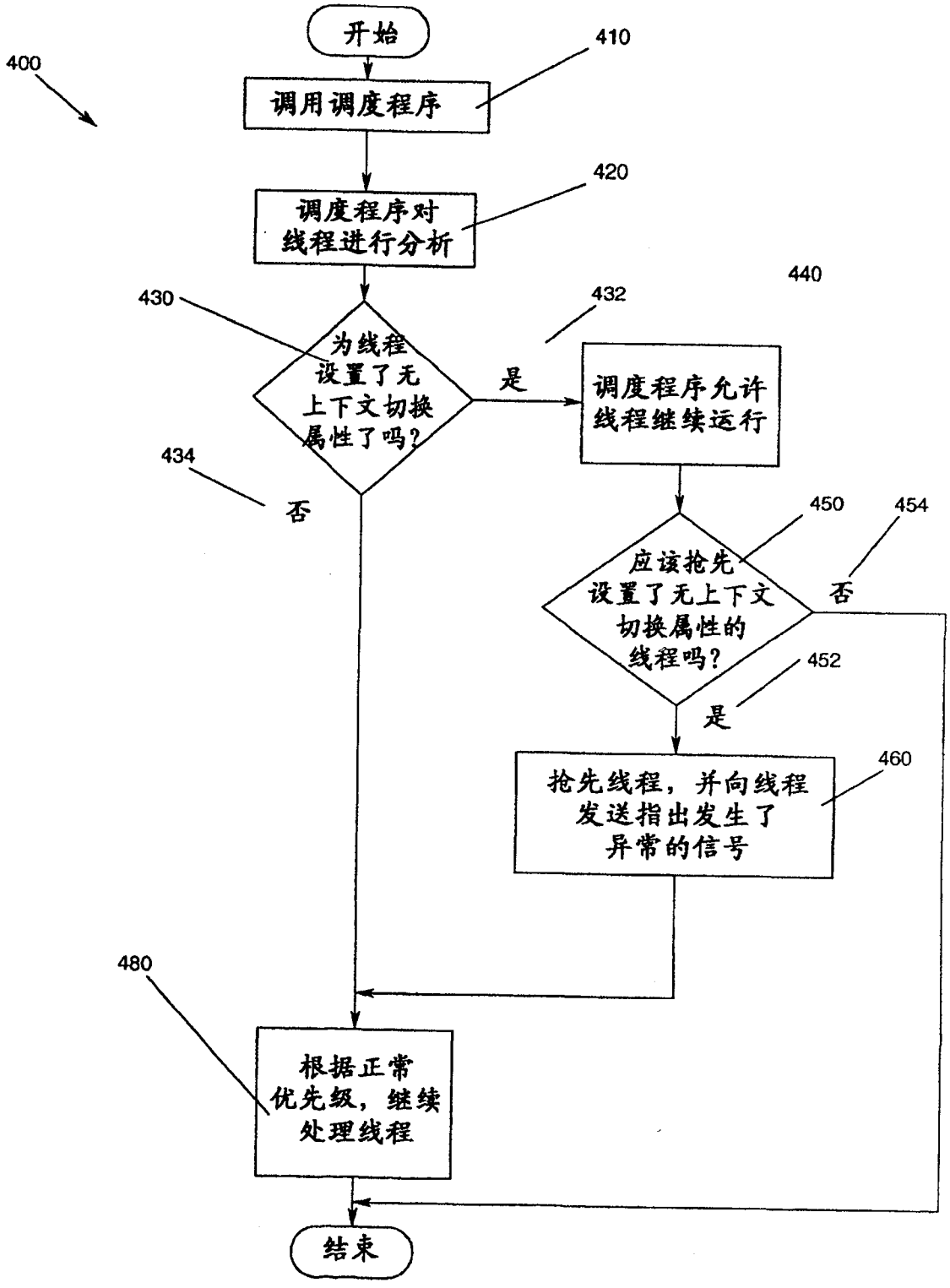

开始
400
410
调用调度程序
420
调度程序对
线程进行分析
430
为线程
设置了无
上下文切换
属性了吗?
是
432
440
调度程序允许
线程继续运行
434
否
450
454
应该抢先
设置了无上下文
切换属性的
线程吗?
否
452
是
460
抢先线程，并向线程
发送指出发生了
异常的信号
480
根据正常
优先级，继续
处理线程
结束

图4

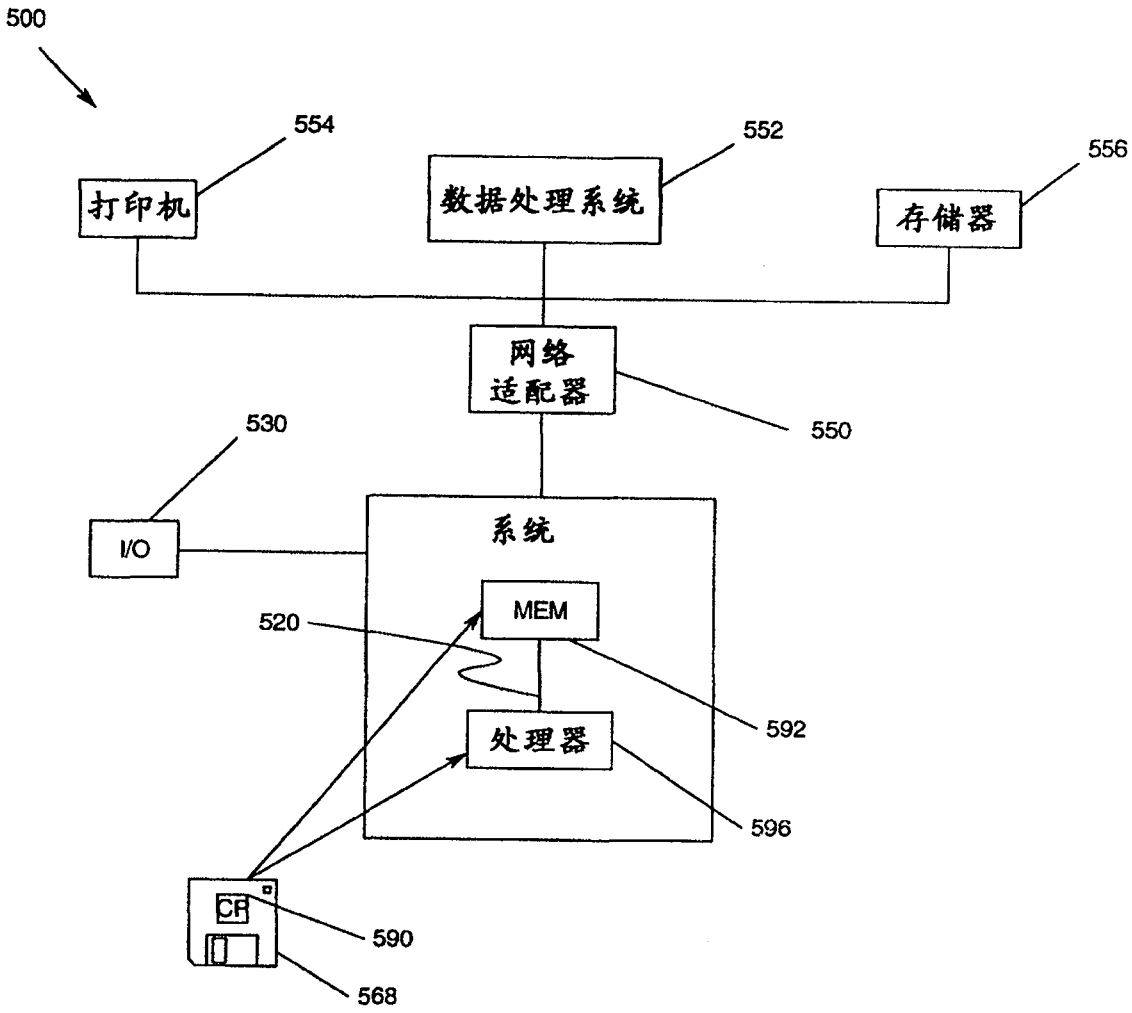

500
554
打印机
552
数据处理系统
556
存储器
网络
适配器
550
530
I/O
系统
MEM
520
592
处理器
596
CR
590
568

图5