[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/48 (2006.01)
G06F 11/36 (2006.01)

[12] 发明专利说明书

专利号 ZL 200710137019.0

[45] 授权公告日 2009 年 6 月 24 日

[11] 授权公告号 CN 100504792C

[22] 申请日 2007.7.19
[21] 申请号 200710137019.0
[30] 优先权
[32] 2006.10.6 [33] EP [31] 06301024.3
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 马克·沃特斯
[56] 参考文献
US6047124A 2000.4.4
US5274811A 1993.12.28
审查员 田 冰

[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

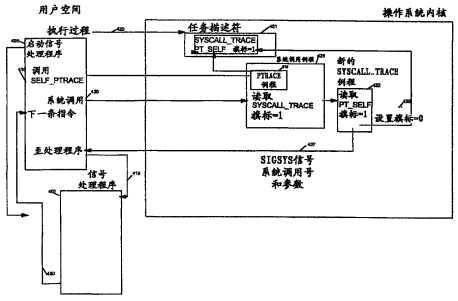
权利要求书 3 页 说明书 11 页 附图 4 页

[54] 发明名称

在用户空间中进行系统调用截取的方法和系统

[57] 摘要

一种从用户空间的过程进行的截取系统调用以及执行与截取的系统调用有关的处理的方法和系统。 所述方法包括第一步骤，用于在执行过程中启动信号处理程序，在收到特定信号后执行与执行过程中系统调用的截取有关的处理。 执行过程以新的请求执行 ptrace 系统调用——自 ptrace，它在执行过程的任务描述符中设置现有系统跟踪 ptrace 请求对应系统跟踪旗标，并且设置新的旗标——pt_self 旗标。 在执行过程中进行系统调用时，执行新的系统调用跟踪，测试 pt_self 旗标是否置位。 如果 pt_self 旗标已经置位，就将特定信号发送到执行过程本身并且在用户空间中执行已经启动的处理程序。



1. 一种在用户空间中执行过程中截取系统调用并且在用户空间中执行与截取的系统调用有关的处理的方法，所述方法在计算机上执行，其中，操作系统内核支持包括系统跟踪的 ptrace 系统调用，所述方法进一步包括:

在执行过程中启动信号处理程序，所述信号处理程序包含执行与将要截取的系统调用有关的操作的代码;

在执行过程中，在执行将要截取的系统调用之前，进行在内核中设置系统跟踪旗标和自跟踪旗标的自 ptrace 请求;

在执行系统调用之后，内核验证已设置系统跟踪旗标和自跟踪旗标;

内核使系统跟踪旗标和自跟踪旗标复位;

内核保存系统调用信息并发送与信号处理程序对应的、传递系统调用信息的信号;

执行过程收到信号后，开始信号处理程序的执行;

在信号处理程序执行结束时，在执行过程中，在所截取系统调用后跟随的下一条指令重新开始执行，同时提供系统调用返回信息。

2. 根据权利要求 1 的方法，其中执行自 ptrace 请求的步骤包括:

在执行过程的任务描述符中保存系统跟踪旗标和自跟踪旗标。

3. 根据权利要求 1 的方法，其中执行自 ptrace 请求的步骤进一步包括:

在执行过程的任务描述符中保存自跟踪旗标，在执行过程的线程描述符中保存系统跟踪旗标。

4. 根据权利要求 1 至 3 中任何一条的方法，其中，内核发送传递系统调用信息的信号的步骤进一步包括:

内核发送传递系统调用标识符和参数的信号。

5. 根据权利要求 1 至 3 中任何一条的方法，其中，内核发送与信号处理程序对应的信号的步骤进一步包括:

所述信号为操作系统已经使用的一个信号，并且在执行先前步骤时从其初始用途进行了修改；

如果没有执行所述方法的先前步骤，所述信号由内核不经修改地使用。

6. 根据权利要求 1 至 3 中任何一条的方法，其中，所述信号处理程序包含用于进行预处理或系统调用仿真或后处理的代码。

7. 根据权利要求 1 至 3 中任何一条的方法，其中，所述信号处理程序包含用于虚拟或调试所述执行过程的代码。

8. 一种用于在用户空间中执行过程中截取系统调用并且在用户空间中执行与截取的系统调用有关的处理的系统，其中，操作系统内核支持包括系统跟踪的 ptrace 系统调用，所述系统包括：

用于在执行过程中启动信号处理程序的装置，所述信号处理程序包含执行与将要截取的系统调用有关的操作的代码；

用于在执行过程中，在执行将要截取的系统调用之前，进行在内核中设置系统跟踪旗标和自跟踪旗标的自 ptrace 请求的装置；

用于在执行系统调用之后，内核验证已设置系统跟踪旗标和自跟踪旗标的装置；

用于令内核使系统跟踪旗标和自跟踪旗标复位的装置；

用于令内核保存系统调用信息并发送与信号处理程序对应的、传递系统调用信息的信号的装置；

用于在执行过程收到信号后，开始信号处理程序的执行的装置；

用于在信号处理程序执行结束时，在执行过程中，在所截取系统调用后跟随的下一条指令重新开始执行，同时提供系统调用返回信息的装置。

9. 根据权利要求 8 的系统，其中执行自 ptrace 请求的装置包括：

用于在执行过程的任务描述符中保存系统跟踪旗标和自跟踪旗标的装置。

10. 根据权利要求 8 的系统，其中执行自 ptrace 请求的装置包括：

用于在执行过程的任务描述符中保存自跟踪旗标，在执行过程的

线程描述符中保存系统跟踪旗标的装置。

11. 根据权利要求 **8** 至 **10** 中任何一条的系统，其中，令内核发送传递系统调用信息的信号的装置进一步包括:

用于令内核发送传递系统调用标识符和参数的信号的装置。

12. 根据权利要求 **8** 至 **10** 中任何一条的系统，其中，所述信号处理程序包含用于进行预处理或系统调用仿真或后处理的代码。

13. 根据权利要求 **8** 至 **10** 中任何一条的系统，其中，所述信号处理程序包含用于虚拟或调试所述执行过程的代码。

在用户空间中进行系统调用截取的方法和系统

技术领域

一般来说，本发明涉及截取系统调用的方法，更确切地说，这种方法允许从用户空间截取系统调用。

背景技术

应用程序级的若干虚拟化系统将应用程序与底层物理硬件隔离，其目的是为了保护（容错）、（利用在 Linux 上运行的 IBM 的 MetaCluster，MetaCluster 是 IBM 公司在某些国家中的商标）通过检查点和重新开始的流动性（应用程序再定位）、确定性重放或者仅仅是资源隔离，如 Linux（Linux 是 Linux Torvalds 在美国、其他国家或兼而有之的注册商标）的 Vserver（Vserver 是 Linux Torvalds 在某些国家中的商标）、Virtuozzo（Virtuozzo 是 Swsoft 在某些国家中的商标）的 OpenVZ，它们都需要截取和改变现有系统调用的原始语义。

做到这一点的一种方法是在内核中改变系统调用例程，以引进语义的系统调用截取和修改。在操作系统内部执行必需的改变难以实行，危及整个系统的稳定性和安全性，并且用户和维护者通常不太愿意接受，因为它增加了内核的复杂性并可能危及系统的完整性和支持它的能力。

存在着某些方法将代码插入到程序中以分析其行为，例如通过收集分析数据。这种修改程序使其分析自己的技术被称为“探测方法”。探测方法可以用于检测系统调用，有可能以这种方式从用户空间修改它们。不过，现有探测方法虽然对于调试足以胜任，但是不能满足高性能要求，如容错系统的要求。

检测可执行代码的“ptrace”方法——如 Linux strace 工具使用的——需要外部控制器过程，当信号通知它时，它在每次系统调用发生

时都停止、内观，然后再重新开始。尽管这种方法是一般的方法，但是形成的运行开销巨大。

LD_PRELOAD 方法——也是检测可执行代码——执行动态链接符号插入，以截取和替换以动态符号形式存在的系统调用。这种方法仅限于动态可执行文件，不适用于系统调用内联在库中的情况（因为存在着关联符号）。现在内联系统调用在最近的 **Linux** 标准库中越来越常见，使得这种方法遭到反对。

机器码重写法是另一种可执行码的探测方法：可执行机器码被静态地或动态地重写，并且当遇到系统调用时，能够插入某种附加代码以提供附加值。这种方法不支持自修改的可执行码，并且运行开销也可能非常可观。实例是数字设备公司工作站上过去可用的 **ATOM** 产品。**ATOM** 在编译时向要被分析的程序内插入代码。

因此需要一种新方法，在程序执行期间截取所有类型的系统调用，并且从用户空间修改它们的行为，同时避免（对容错系统无法接受的）运行开销，因为内核码以特权模式执行，并且不能由程序修改。

发明内容

本发明的目的是提供截取系统调用并从用户空间修改它们行为的探测方法，它适用于任何类型的可执行码，同时保持良好的性能水平。

根据本发明的一个方面，达到这个目的是利用了在用户空间中执行过程中截取系统调用并且在用户空间中执行与截取的系统调用有关的若干操作的方法，所述方法在计算机上执行，其中，操作系统内核支持包括系统跟踪的 **ptrace** 系统调用，所述方法进一步包括：

— 在执行过程中启动信号处理程序，所述信号处理程序包含执行与将要截取的系统调用有关的操作的代码；

— 在执行过程中，在执行将要截取的系统调用之前，进行在内核中设置系统跟踪旗标和自跟踪旗标的自 **ptrace** 请求；

— 在执行系统调用之后，内核验证已设置系统跟踪旗标和自跟踪旗标；

— 内核使系统跟踪旗标和自跟踪旗标复位;

— 内核保存系统调用信息并发送与信号处理程序对应的、传递系统调用信息的信号;

— 执行过程收到信号后，开始信号处理程序的执行;

— 在信号处理程序执行结束时，在执行过程中，在所截取系统调用后跟随的下一条指令重新开始执行，同时提供系统调用返回信息。

优选地，其中执行自 ptrace 请求的步骤包括:

— 在执行过程的任务描述符中保存系统跟踪旗标和自跟踪旗标。

优选地，其中执行自 ptrace 请求的步骤进一步包括:

— 在执行过程的任务描述符中保存自跟踪旗标，在执行过程的线程描述符中保存系统跟踪旗标。

优选地，其中，内核发送传递系统调用信息的信号的步骤进一步包括:

— 内核发送传递系统调用标识符和参数的信号。

优选地，其中，内核发送与信号处理程序对应的信号的步骤进一步包括:

— 所述信号为操作系统已经使用的一个信号，并且在执行先前步骤时从其初始用途进行了修改;

— 如果没有执行所述方法的先前步骤，所述信号不由内核修改地使用。

优选地，其中，启动信号处理程序的步骤进一步包括:

— 所述信号处理程序包含进行预处理或系统调用仿真或后处理的代码。

优选地，其中，启动信号处理程序的步骤进一步包括:

— 所述信号处理程序包含虚拟或调试所述执行过程的代码。

根据本发明的另一方面，涉及一种系统，用于在用户空间中执行过程中截取系统调用并且在用户空间中执行与截取的系统调用有关的处理，其中，操作系统内核支持包括系统跟踪的 ptrace 系统调用，所述系统包括: 用于在执行过程中启动信号处理程序的装置，所述信号处理程序包含执行与将要截取的系统调用有关的操作的代码; 用于在

执行过程中，在执行将要截取的系统调用之前，进行在内核中设置系统跟踪旗标和自跟踪旗标的自 ptrace 请求的装置；用于在执行系统调用之后，内核验证已设置系统跟踪旗标和自跟踪旗标的装置；用于令内核使系统跟踪旗标和自跟踪旗标复位的装置；用于令内核保存系统调用信息并发送与信号处理程序对应的、传递系统调用信息的信号的装置；用于在执行过程收到信号后，开始信号处理程序的执行的装置；用于在信号处理程序执行结束时，在执行过程中，在所截取系统调用后跟随的下一条指令重新开始执行，同时提供系统调用返回信息的装置。

利用本发明，在执行的程序进行系统调用时，内核的系统调用通知机构产生信号。如果在执行的程序中已经执行了本发明特定的新 PTRACE 系统调用，这个信号就发送到执行的程序本身，它可以提供用户空间信号处理程序以执行新的代码。

应当指出，即使本发明对 Linux 系统参考手册中描述的现有 PTRACE 系统调用实施了新的请求，但是除了 Linux 以外，所有 UNIX 系统都已经提供了 'ptrace 式' 特点并具有现有的系统调用通知机构（由 strace 或 truss 工具的存在所表明）。这暗示了在本发明之上开发的服务的可移植性。一旦在新的 UNIX 系统中实施了本发明，先前 UNIX 系统上的用户空间中已经开发的服务马上变为对新系统可用。

具有调试器——它取得过程的控制并强制逐步的执行——的所有操作系统都以新的 PTRACE 请求实施例提供 ptrace 系统调用或类似手段——可以按建议进行修改。根据本发明的解决方案，如果 PTRACE 系统调用不存在，一个其他实施例应当在内核中创建特定的服务，它将仅仅执行系统调用通知的激活。

本发明的解决方案适用于任何种类的可执行码（静态的或动态的），支持内联系统调用以及由自修改的代码动态产生的系统调用。

通知的运行开销极低。它对应于信号中断成本的一半。因为在检测中不包括附加任务，所以在任务之间没有附加的上下文切换，也没有附加的过程间通讯。这些特征确保了良好的性能水平。

本解决方案的其他优点可以列举如下：

— 为了支持系统调用的整个集合的截取和虚拟，除了唯一的简单小补

丁以外，内核中不需要进一步的修改。

— 这种机制也允许广义的概念，将本发明的方法应用于不存在的或不支持的系统调用，只要它们能够被截取和仿真。

— 安全性：在执行过程的情况下，在这种截取机制之上实施的全部服务都驻留在用户空间中，完全如同常规的动态库。它们不会危及操作系统或外部应用程序。不存在使内核崩溃的风险。

— 开发速度：虚拟、检查点/重新开始、记录/重放或其他服务现在都能够在用户空间中开发，不必篡改系统内部，也不必重新开机。灵活性多得多，并且可能在用户级接入各式各样的现有服务。

— 保障：保持了现有过程的保障模型。内核或其他过程的内部都不允许所述过程访问。同样，所述过程也不向其他实体公开任何信息。进行保障评估极为简单，因为内核修改仅仅包含在信号的自通知中。

附图说明

图 1 介绍了根据优选实施例以 SELF_PTRACE 请求执行 PTRACE 系统调用的结果；

图 2 是根据优选实施例的方法修改的系统调用入口的流程图；

图 3 是根据优选实施例的方法在用户空间中执行信号处理程序的流程图；

图 4 展示了根据本发明优选实施例的用户空间和操作系统中的多种组件。

具体实施方式

实施优选实施例的方法需要使用操作系统内核的系统调用跟踪例程，以 ptrace 实施系统调用的跟踪。正如本文档马上要论述的，Linux 可以被视为这样的操作系统的实例，不过今日的大多数操作系统，包括嵌入式操作系统，都在其内核中提供了 ptrace 系统调用，以及以 ptrace 跟踪系统调用。在下文中，即使 Linux 语法用于展示优选实施例的解决方案，不过所介绍的 ptrace 系统调用和系统调用的处理在其

他操作系统中实施时也是类似的。

Linux 和其他 Unix 以 ptrace 系统调用检测用户/系统边界，它提供了用户空间控制器视界以及对另一个（受控）过程的控制。无论何时，只要受控过程要接收信号、进入系统调用或从其退出时，都通知所述控制器过程。恰好在所述事件之前停止所述控制器过程，给予所述控制器机会捕捉并转发信号，以检查或修改受控过程的存储器和寄存器，最后令受控过程恢复所述事件的正常处理。这用于程序调试目的或跟踪系统调用。

将要被跟踪的过程中使用的现有 ptrace 语法如下:

```
#include  <sys/ptrace.h>
long int ptrace(enum _ptrace_request request, pid_t pid, void * addr,
void * data)
```

其中 'request' 定义了将要进行的行为。例如，ptrace 系统调用执行的请求 PTRACE_TRACEME 告诉内核，进行调用的过程希望被跟踪。其他请求使用的 'pid' 参数是将要被跟踪的活动过程的 id; 'addr' 是控制器过程将读取的、被跟踪过程的用户区域中的地址; 'data' 是来自跟踪的过程区域的数据，它将取代被跟踪过程在地址 addr 的信息。使用 PTRACE_SYSCALL 请求时无需参数，它使将要被跟踪的过程在下一次系统调用之后停止。

现有的 PTRACE_SYSCALL 请求使受控任务的内核上下文中的 SYSCALL_TRACE 旗标置位，表明系统调用跟踪受到请求。然后，在被跟踪的过程执行下一次系统调用时，内核的系统调用例程检测 SYSCALL_TRACE 是否置位。如果该旗标没有置位，那么就执行系统调用。如果该旗标已置位，就执行系统调用例程: 停止该过程并向控制器过程发送信号，它将能够从内核访问被跟踪过程进行的系统调用有关的信息。

根据优选实施例，通过修改现有的系统调用跟踪例程，在 Linux 内核中实施新的 ptrace 请求 SELF_PTRACE，在控制器过程已经进行了 PTRACE_SYSCALL 请求时，在内核中执行它。控制器过程使用

现有的 PTRACE_SYSCALL 请求截取受控过程中的系统调用。它允许例如从控制器过程查看系统调用参数，并且可能改变它们。在优选实施例中，提议了系统调用跟踪例程的修改，意在以信号自通知某过程，以允许该过程从用户空间将自身置于其自己的系统调用上。新的 ptrace 请求 SELF_PTRACE 加入内核中。它具有能力通过信号通知该过程本身，无论何时只要它执行系统调用。然后，系统调用的截取和例如虚拟可以在用户空间中的信号处理程序中执行。为了从信号处理程序透明地恢复以及将（真实的或仿真的）系统调用返回值传送到进行调用的代码，提供了特定的过程（本文档后面图 3 中的 320）。总之，通过在内核中加入简单补丁实施的方法，以及用户空间中的信号处理程序，允许从用户空间置于系统调用上。

这种请求的 ptrace 系统调用的语法如下；它不使用 ptrace 系统调用的任何参数:

```
#include   <sys/ptrace.h>
long int ptrace(enum _ptrace_request SELF_PTRACE)
```

图 1 介绍了根据优选实施例以 SELF_PTRACE 请求执行 PTRACE 系统调用的结果。在需要从用户空间中截取和修改系统调用的用户空间中执行过程中，执行 PTRACE 系统调用。在 Linux 环境中介绍这幅流程图。正如以 SYSCALL_TRACE 请求，SYSCALL_TRACE 旗标（100）在该过程的上下文中置位，保存在内核中。所谓的 PT_SELF 旗标（110）是新的旗标，也在该过程的上下文中置位，以表明为该过程配备了自跟踪机构。由于 Linux 中性能的缘故，优选情况下在该过程对应的当前任务的内核任务描述符（120）的 ptrace 字段中设置 PT_SELF 旗标，在该任务描述符的线程信息结构（130）中设置 SYSCALL_TRACE 旗标。

图 2 是根据优选实施例的方法修改的系统调用入口的流程图。对于需要在用户空间中截取和修改系统调用的过程，通过在所述过程中执行 SELF_PTRACE 请求 ptrace 系统调用而配备了自跟踪机构。根据参考上图介绍的优选实施例，这意味着在执行 ptrace 系统调用的

SELF_PTRACE 请求时，在内核空间中保存的过程描述符中已经设置了 SYSCALL_TRACE 和 PT_SELF 旗标。在调用系统调用之前，该过程正常执行。在所有操作系统中，它照例调用软件中断（200）到内核，内核在检查系统调用的参数后将执行系统调用。在执行将要截取的系统调用之前，内核在任务描述符中查看 SYSCALL_TRACE 旗标是否置位（210）。如果答案为否（SYSCALL_TRACE 旗标等于 0），就执行系统调用并将结果返回到该过程，它在用户空间中恢复执行（270）。如果答案为是（SYSCALL_TRACE 旗标等于 1），内核就执行 SYSCALL_TRACE 例程（220）。根据优选实施例，已经通过加入 PT_SELF 旗标的测试（230）而修改了 SYSCALL_TRACE 例程。如果 PT_SELF 旗标没有置位(它等于 0)，就执行 SYSCALL_TRACE 例程：将信号发送到先前已经执行了等待系统调用的控制器过程（280）。这个信号传递了 SYSCALL_TRACE 结果，以便控制器过程对该结果执行操作。内核等待控制器过程完成（290）。当控制器过程请求重新开始执行被跟踪的过程时，就为被跟踪的过程执行系统调用并将结果送回被跟踪的过程，它在用户空间中恢复执行（270）。

根据本优选实施例，如果 PT_SELF 旗标已置位，这意味着在执行过程中已经要求了自 ptrace 并且第一步骤包括为了执行过程中下一个到来的系统调用而解除自 ptrace 过程。这将避免递归。在执行过程的描述符中由此将 SYSCALL_TRACE 和 PT_SELF 旗标复位至 0（240）。然后，由内核向该过程本身发送 SIGSYS 信号（250）。与 PTRACE 正常过程相比，这是很大的改变，它不警告执行过程本身而是执行过程的控制器过程（280）。

选择 SIGSYS 信号是因为 Linux 中的 SIGSYS 信号按惯例用于传送 SYSCALL ptrace 系统调用不利结束的通知信息。在 SIGSYS 信号的‘不利系统调用’使用的情况下，该信号传送的信息是已经产生该不利系统调用的地址。在本优选实施例中，SIGSYS 信号传送将由信号处理程序执行的处理所截取系统调用的执行期间所需的信息（系统调用号和参数），正如本文档后面关于图 3 的介绍。然后，内核向该

过程转交控制（255）。

应当指出，操作系统中可用的任何软件中断信号（例如在 Linux 操作系统中有 64 种可能的信号）都可以用于实施本发明。不过，使用信号的方法却必须容许该信号的常规用途。例如，在实施本发明所用的信号是 SIGSYS 信号的情况下，无论何时执行 SYSCALL_TRACE 例程，该信号都向控制器过程传送已经产生该不利系统调用的地址。

图 3 是根据优选实施例的信号处理程序的流程图。在用户空间中执行与被截取的系统调用有关的处理，作为该过程的信号处理程序。在该过程的执行期间执行将要截取的系统调用前已经启动了信号处理程序。所以，当执行过程已经收到了 SIGSYS 信号并且内核向该过程转交了控制时，就执行信号处理程序。在信号处理程序中执行以下步骤：

— 提取系统调用号和参数（300）：它们存储在所保存的寄存器中作为信号上下文的一部分，根据本优选实施例由内核在发送该信号前自动设置。正如从现有的信号处理程序的处理已知，根据 Linux 中信号处理程序的标准处理（可以参考 Linux 手册 sigaction 页面），信号上下文地址通过堆栈传递到处理程序作为第三参数。正如在信号处理程序的标准处理中，信号处理程序将使用信号上下文指令计数器在执行后返回执行过程中所截取系统调用随后的下一条指令。

— 按要求使用识别该系统调用的系统调用号和系统调用参数进行任何预处理、系统调用仿真、后处理（310），例如通过虚拟或调试服务。由于在发送信号前在内核中已经清除了 SYSCALL_TRACE 旗标和 PT_SELF 旗标，应用程序进行的所有后续系统调用将不被截取。这避免了递归问题。

— 使用收到的上下文数据（Linux 中的 sigcontext）恢复执行过程的上下文（320），具有两种改变：强制所希望的系统调用返回专用寄存器（Intel 处理器上的 eax 寄存器，Intel 是 Intel 公司或其子公司在美国和其他国家的商标或注册商标）中的数值，直接跳转到调用代码中系统调用中断随后的的下一条指令(在 Intel 处理器上指令指针寄存器

值+2）。

因此，在需要时通过在 ptrace 系统调用服务中加入新的 ptrace 请求，以及（通过增加步骤 230、240、250、260）修改操作系统内核中的 SYSTEM_TRACE ptrace 请求，在该过程并且在用户空间中的信号处理程序中执行以下操作：

— 截取一个执行过程中的系统调用，

— 改变到这个系统调用，以及

— 增加如果操作，比如调试、记载或虚拟所述执行过程所属的应用程序。

这种相同的方法可以应用于从用户空间控制任何系统调用的执行。

图 4 展示了根据本发明优选实施例的用户空间和操作系统中的多种组件。在用户空间中执行过程需要具有在用户空间中截取和修改的系统调用。承担任务管理的、操作系统的内核保存（420）执行过程的上下文，作为内核空间中的任务描述符（421）。根据本优选实施例，该过程的代码执行信号处理程序的启动（400）。处理程序的代码包含与需要被截取的系统调用有关的、需要完成的全部处理：这种处理可以涉及调试或虚拟操作。执行过程的代码以 SELF_PTRACE 请求执行 ptrace 系统调用（410）。内核中的系统调用例程（425）将系统调用路由至 ptrace 服务例程（415），对这个请求在内核中执行它：在任务描述符中设置了 SYSCALL_TRACE 和 PT_SELF 旗标两个旗标。当执行过程进行系统调用（430）时，系统调用例程（425）检验 SYSCALL_TRACE 旗标已置位并将该调用路由至新的 SYSCALL_TRACE 例程（432），它将任务描述符的这两个旗标设置为 0（435），并在一个实施例中以系统调用的信息发送 SIGSYS 信号（437）。内核发布了这个信号后，重新开始用户空间中执行过程。由于信号处理程序已经启动，信号处理程序就开始执行用户空间中的信号处理程序（419）。信号处理程序执行与系统调用有关的处理。在信号处理程序执行结束时，执行过程在系统调用随后的下一条指令处重新开始（450）。在图 4 中，点线框表示现有组件，新的或修改的组件

以实线框表示。从图 4 可以看出，通过在设置任务描述符中若干旗标的 ptrace 服务例程中创建新的 SYSCALL_TRACE 例程和新的请求，在内核中实施了本发明的方法。通过在执行过程和信号处理程序中增加信号处理程序和自 ptrace 系统调用的启动，在用户空间中实施了其余部分。

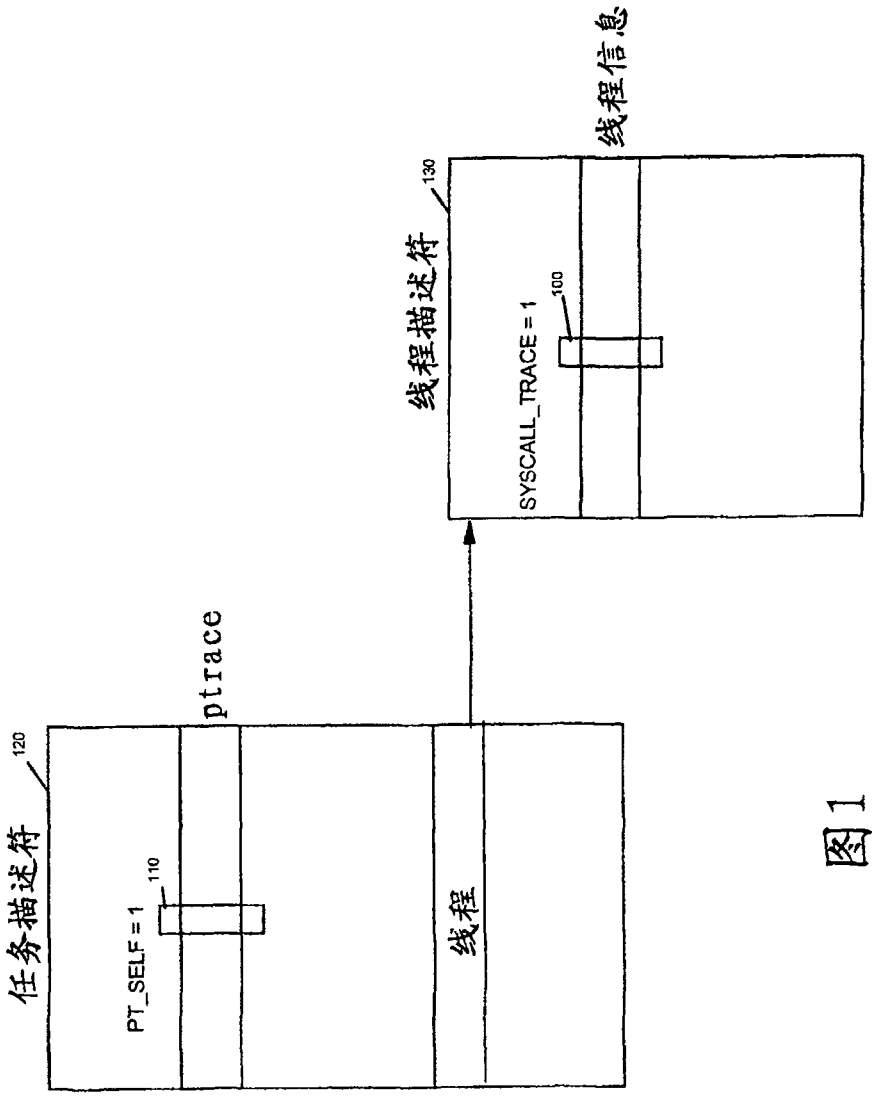

线程信息
线程描述符
130
100
SYSCALL_TRACE = 1
ptrace
任务描述符
120
110
PT_SELF = 1
线程

图 1

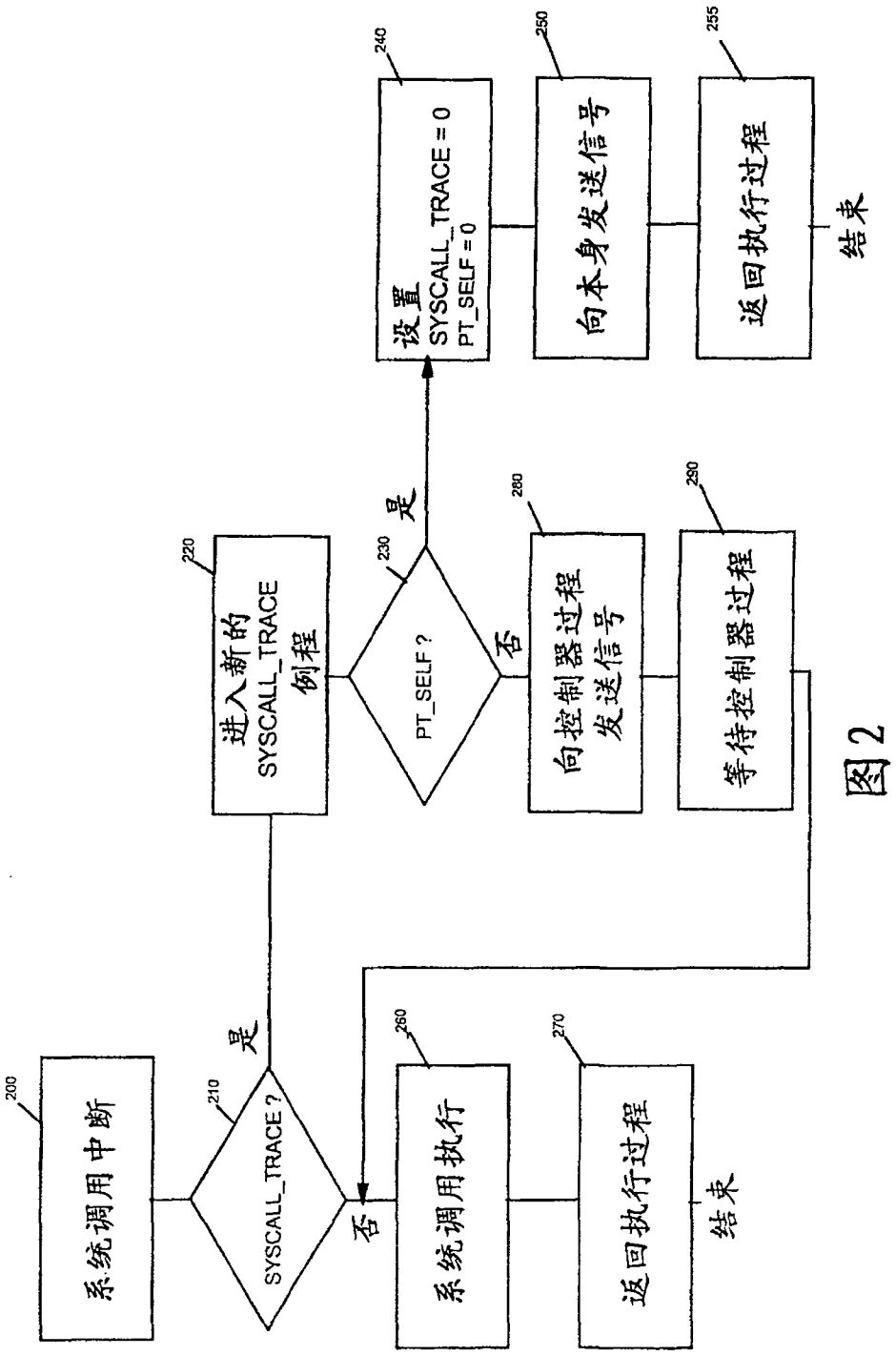

200
系统调用中断
210
SYSCALL_TRACE ?
是
否
220
进入新的 SYSCALL_TRACE 例程
230
PT_SELF ?
是
否
240
设置 SYSCALL_TRACE = 0 PT_SELF = 0
250
向本身发送信号
255
返回执行过程
结束
260
系统调用执行
270
返回执行过程
结束
280
向控制器过程发送信号
290
等待控制器过程

图 2

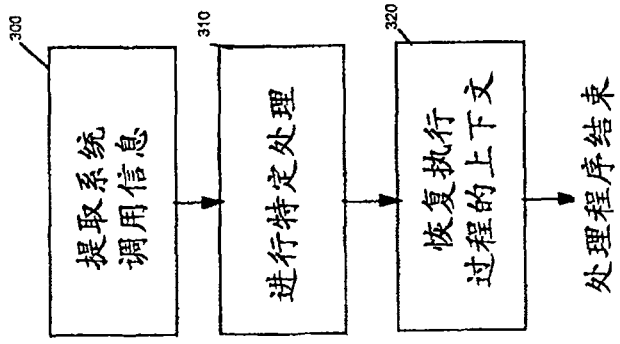


图3

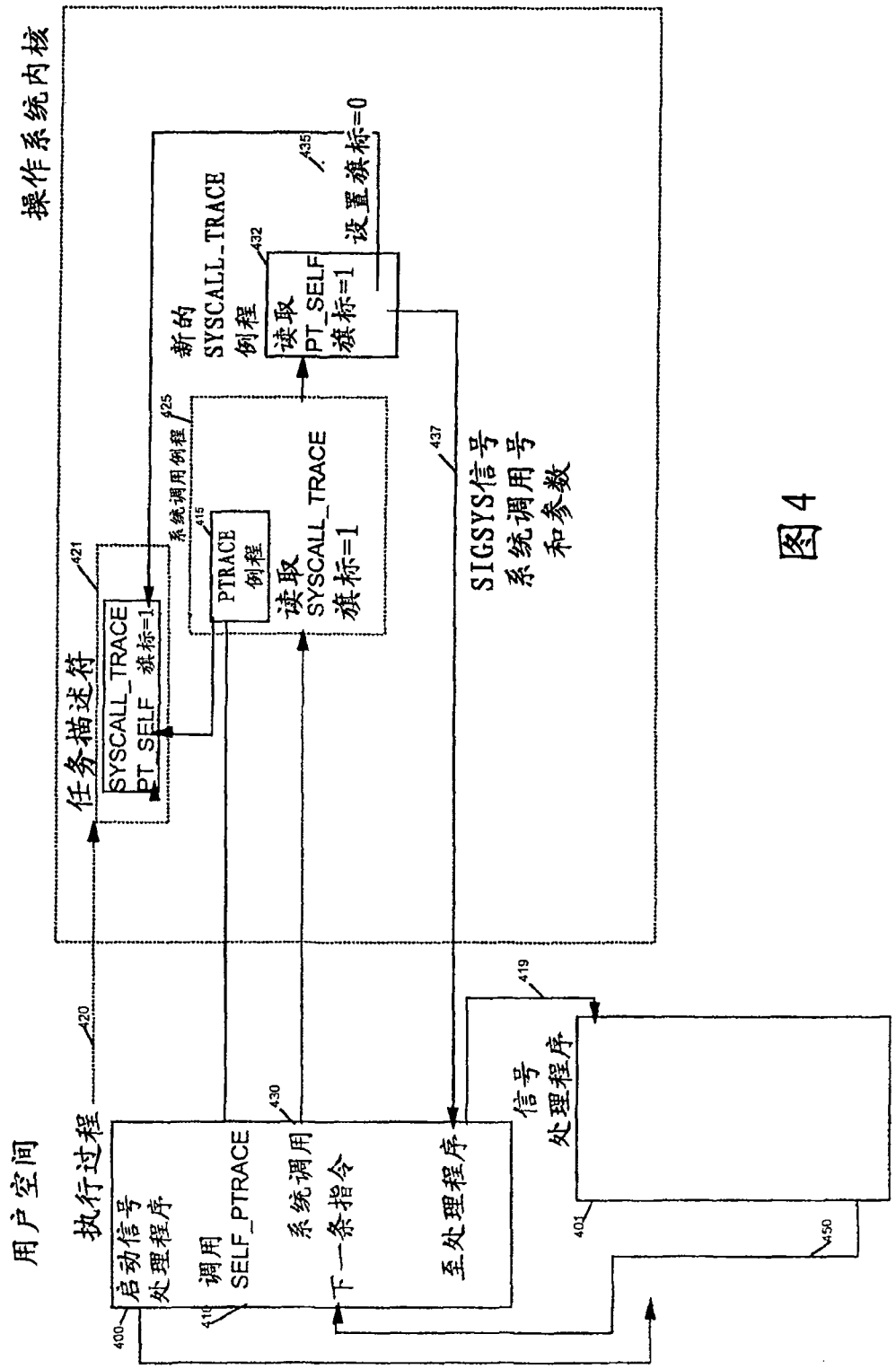

用户空间
操作系统内核
执行过程
420
启动信号
处理程序
调用
SELF_PTRACE
410
400
系统调用
下一条指令
430
至处理程序
信号
处理程序
419
401
450
任务描述符
421
SYSCALL_TRACE
PT_SELF 旗标=1
系统调用例程
425
PTRACE
例程
415
读取
SYSCALL_TRACE
旗标=1
新的
SYSCALL_TRACE
例程
读取
PT_SELF
旗标=1
432
设置旗标=0
435
437
SIGSYS信号
系统调用号
和参数

图 4