(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101192238 B
(45)授权公告日 2012.09.05

(21)申请号 200710186961.6
(22)申请日 2007.11.15
(30)优先权数据
0623916.4 2006.11.30 GB
(73)专利权人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 菲利普·G·威洛比
(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 黄小临
(51)Int.Cl.
G06F 17/30(2006.01)
(56)对比文件
US 5313646 A,1994.05.17,说明书第2栏第34-62行,第3栏第20-40行,第4栏第55-60行,第5栏第5-10行,第30-65行,第6栏第37-58行,第7栏第54-64行,第8栏第27-30行.
审查员 李劲娴

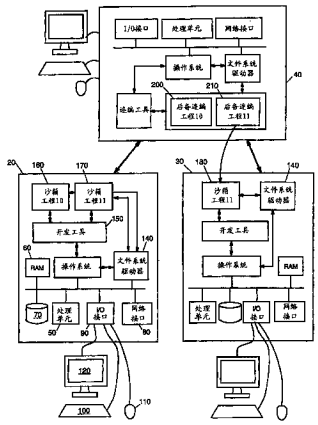
权利要求书 2 页 说明书 9 页 附图 6 页

(54)发明名称

测试文件系统的方法和使用该方法的文件系统存取控制器和系统

(57)摘要

提供用于在数据处理系统中的改变管理、更具体地用于模拟从系统移除文件的效果的方法、设备和计算机程序。在被覆盖在一组计算机程序文件上的文件系统层内创建反文件。该反文件隐藏了对应的第一文件,以便在这组计算机程序上覆盖反文件层模拟了从这组程序文件中移除第一文件的效果。



CN 101192238 B

1. 一种用于测试文件系统的文件系统存取控制器，包括：

用于在作为连编处理部分的测试期间标识在新逻辑文件系统中的一组逻辑上区别的层的装置，其中新逻辑文件系统包含反文件层和后备层，其中所述后备层包括第一组文件系统资源，且其中所述反文件层包括反文件，该反文件指定了后备层中的第一文件，该第一文件将被文件系统存取控制器可逆地掩盖来测试新逻辑文件系统，以便确定在实际上没有移除所述第一文件的情况下所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用；

用于标识所述后备层内第一文件与所述反文件层内反文件之间的匹配以及将所述反文件层内反文件解释为一个用于对任何试图存取第一文件的程序可逆掩盖所述第一文件的存在的指令的装置；以及

用于响应于标识匹配而在没有从所述后备层移除所述第一文件的情况下通过可逆地向至少一个请求者掩盖所述第一文件的存在而为了测试目的而模拟移除第一文件的装置；

用于响应于可逆地掩盖所述第一文件的存在，在所述第一文件的存在被可逆地掩盖的同时测试所述新逻辑文件系统的装置，以便为测试目的确定所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用。

2. 根据权利要求 1 的文件系统存取控制器，进一步包括用于标识前层的装置，其中所述前层包括覆盖所述反文件层和所述后备层的附加的一组文件系统资源。

3. 根据权利要求 1 或权利要求 2 的文件系统存取控制器，其中所述用于标识一组逻辑上区别的层的装置包括用于识别对应于所述后备层的目录的目录名的装置，且其中所述后备层的第一组文件系统资源被分配给所述后备层目录。

4. 根据前述权利要求 3 的文件系统存取控制器，其中，所述用于标识一组逻辑上区别的层的装置包括用于识别所述反文件层的该组文件引用被分配给的指名反文件目录的目录名的装置。

5. 根据权利要求 3 的文件系统存取控制器，其中所述用于标识一组逻辑上区别的层的装置包括用于识别包含所述反文件层的该组文件引用在内的指 名文件的文件名的装置。

6. 根据前述权利要求 5 的文件系统存取控制器，其中，所述用于标识匹配的装置包括用于检查反文件层以寻找所述第一文件的文件引用的装置。

7. 根据权利要求 6 的文件系统存取控制器，其中，所述用于标识匹配的装置还包括用于检查反文件层以寻找所述第一文件的父母的引用的装置。

8. 根据前述权利要求 7 的文件系统存取控制器，其中，所述用于掩盖所述第一文件的存在的装置包括响应于来自第一请求者对于存取所述第一文件的请求、生成文件未找到错误消息并将该错误消息返回所述第一请求者的装置。

9. 根据前述权利要求 8 的文件系统存取控制器，其中所述至少一个请求者是请求执行关于所述第一文件的操作的计算机程序，该操作是从包括下列项的组中选择的：读、编辑、删除、创建、执行和改变许可。

10. 一种用于测试文件系统的方法，包括步骤：

在作为连编处理部分的测试期间标识在新逻辑文件系统中的一组逻辑上区别的层，其中新逻辑文件系统包含反文件层和后备层，其中所述后备层包括第一组文件系统资源，且

其中所述反文件层包括反文件，该反文件指定了后备层中的第一文件，该第一文件将被文件系统存取控制器可逆地掩盖来测试新逻辑文件系统，以便确定在实际上没有移除所述第一文件的情况下所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用；

标识所述后备层内第一文件与所述反文件层内反文件之间的匹配以及将所述反文件层内反文件解释为一个用于对任何试图存取第一文件的程序可逆掩盖所述第一文件的存在的指令；以及

响应于标识匹配而在没有从所述后备层移除所述第一文件的情况下通过可逆地向至少一个请求者掩盖所述第一文件的存在而为了测试目的而模拟移除第一文件；

响应于可逆地掩盖所述第一文件的存在，在所述第一文件的存在被可逆地掩盖的同时测试所述新逻辑文件系统，以便为测试目的确定所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用。

11. 一种数据处理系统，包括：

数据处理单元；

数据存储单元；

包括所述数据存储单元内的数据的逻辑排列的文件系统，所述文件系统包括第一文件系统层内的第一文件；

用于在所述文件系统的反文件层中创建所述第一文件的反文件的装置，所述反文件包括对所述第一文件的引用；以及

文件系统存取控制器，用于对于至少一个请求者管理文件系统存取，包括：

用于在作为连编处理部分的测试期间标识在新逻辑文件系统中的一组逻辑上区别的层的装置，其中新逻辑文件系统包含反文件层和后备层，其中所述后备层包括第一组文件系统资源，且其中所述反文件层包括反文件，该反文件指定了后备层中的第一文件，该第一文件将被文件系统存取控制器可逆地掩盖来测试新逻辑文件系统，以便确定在实际上没有移除所述第一文件的情况下所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用；

用于标识所述后备层内第一文件与所述反文件层内反文件之间的匹配以及将所述反文件层内反文件解释为一个用于对任何试图存取第一文件的程序可逆掩盖所述第一文件的存在的指令的装置；以及

用于响应于标识匹配而在没有从所述后备层移除所述第一文件的情况下通过可逆地向至少一个请求者掩盖所述第一文件的存在而为了测试目的而模拟移除第一文件的装置；

用于响应于可逆地掩盖所述第一文件的存在，在所述第一文件的存在被可逆地掩盖的同时测试所述新逻辑文件系统的装置，以便为测试目的确定所述新逻辑文件系统在所述第一文件被有效移除的情况下是否能正常起作用。

12. 根据权利要求 11 的数据处理系统，其中，所述用于创建反文件的装置包括用于将所创建的文件引用分配给反文件目录的装置。

13. 根据权利要求 11 的数据处理系统，其中，所述用于创建反文件的装置包括用于向数据库表内的反文件引用列表追加所创建的文件引用的装置。

测试文件系统的方法和使用该方法的文件系统存取控制器和系统

技术领域

[0001] 本发明提供用于在数据处理环境中改变管理的解决方案。具体地，本发明使得在软件开发和测试系统或运行时数据处理环境中、能够评估从一组计算机程序文件中移除文件的效果。

背景技术

[0002] 对于日益增加的软件开发来说已知的技术是在逻辑地覆盖预存在的源代码树的文件系统目录中放入需要测试的改变。然后可以从得到的组合中创建可执行代码。该技术的优点是，可以在不需要重建整个源代码树的情况下测试并校验许多改变。

[0003] 现有技术提供对于测试添加文件或改变现有文件的效果的足够的支持，但很少有助于测试文件移除的效果。为了确定移除文件的效果，需要软件开发者实际地从由连编系统(build system)使用的源代码树中移除文件，并重建整个程序。除非该文件被实际地移除，否则文件移除效果的综合测试需要考虑对该文件的每个显式或暗含的引用。将被用于连编处理中的工具的暗含的引用是测试员难以手动标识的。

[0004] 例如在使用“沙箱”的开发环境中，产生上述问题。“沙箱”是开发者的私人程序开发环境——与其他沙箱隔离，且与可以同时改变其他代码模块的其他开发者隔离。可以在开发者的隔离的沙箱中开发代码，然后将其与预存在的一组计算机程序文件组合。然后可以对该源代码元素的组合进行连编处理，以生成可执行代码。在沙箱中工作的每个开发者可以在对于其他开发者施加这些更新之前想要测试添加、改变和移除文件的效果。

[0005] 在沙箱内编码的添加文件和文件改变被覆盖在一组后备文件上以使能测试，但比较难以测试移除文件的效果。虽然，开发者可以标识和列出它们想要移除的文件，但是可能出现持续暗中引用名义上“不存在”的文件的大量工具（被执行来建立程序的编译器、汇编器、链接器等等）。使这些工具适合于确定哪个文件应该被视为不存在的连编系统专有(specific)的方式并不现 实。因此，仍然使用物理地移除文件然后重建整个程序的传统方法。

[0006] 虽然该问题降低了软件开发的效率，但是更需要在运行时数据处理环境中充分测试移除文件的效果。一组计算机程序的用户可能正在对其更新成为可获得的应用程序基础设施产品上运行商业重要应用程序。用户需要能够在继续下去之前测试更新他们的基础设施的效果，来最小化丢失他们重要的应用程序的可用性的风险。因为难以标识暗含的文件引用，因此目前难以测试文件移除的效果，直到安装了新的程序文件且从运行时环境中实际地移除老的文件为止。

发明内容

[0007] 本发明的第一方面提供一种文件系统存取控制器，用于对于至少一个请求者管理文件系统存取，包括：

[0008] 用于标识包含至少反文件层和后备层的、在逻辑文件系统中的一组逻辑上区别的层的装置，其中所述后备层包括第一组文件系统资源，且其中所述反文件层包括一组文件引用；

[0009] 用于标识所述后备层内第一文件与所述反文件层内第一文件引用之间的匹配的装置；以及

[0010] 用于响应于标识匹配而向所述至少一个请求者掩盖所述第一文件的存在的装置。

[0011] 本发明的另一方面提供一种由数据处理系统内的文件系统存取控制器实现用于对于至少一个请求者管理文件系统存取的方法。该方法包括：标识包含至少反文件层和后备层的、在逻辑文件系统中的一组逻辑上区别的层，其中所述后备层包括第一组文件系统资源，且其中所述反文件层包括一组文件引用；标识所述后备层内第一文件与所述反文件层内第一文件引用之间的匹配；以及响应于标识匹配而向所述至少一个请求者掩盖所述第一文件的存在。

[0012] 所述至少一个请求者可以是经由文件系统存取控制器来请求存取逻辑文件系统内的文件的任何计算机程序或数据处理系统。

[0013] 匹配后备层内的第一文件的、反文件层内的文件引用在此被称为第一文件的“反文件”。文件系统存取控制器响应于反文件，以便反文件层内的匹配文件引用有效地使文件系统存取控制器掩盖第一文件的存在。即，第一文件显得好像不在尝试经由文件系统存取控制器来存取文件的任何程序或数据处 理设备的视野里。

[0014] 在本发明的一个实施例中，反文件层是包含文件名或符号链接（即，仅包含无数据内容的文件引用）的文件系统或文件系统目录。反文件层可由文件系统存取控制器通过例如反文件层的文件系统名或目录名标识为反文件层。在其他实施例中，反文件层可以简单的是包含反文件名列表的文件（诸如包括代表一组反文件的一组串的数据库表）。后备层也可以是文件系统目录的文件系统。在反文件层内的文件引用可以是与后备层内的文件名相同的文件名。

[0015] 优选地，把反文件层和后备层与前层组合以创建多层文件系统。在优选实施例中，反文件有效地掩盖后备层内的匹配文件，而在没有掩盖前层中的文件的情况下，前层覆盖反文件层。可以在逻辑文件系统内把不同的层实施为一组预定一的指名 (named) 目录（例如目录“/back”、“/anti”和“/front”）。文件系统存取控制器响应于目录名来确定哪个文件名是反文件名，并确定哪个匹配的指名文件（即在“/back”目录内的那些文件）应该被掩盖。

[0016] 在另一实施例中，反文件层包括在与前层相同的文件系统目录内的一组一个或多个文件名。在该实施例中，每个反文件层的文件名都有代表反文件的预定的特殊字符位于前面，由文件系统存取控制器标识这种字符以使能把反文件层内的文件名与后备层内的文件名匹配。例如，反文件层和后备层中的文件名除了预定义的字符以外可以相同。

[0017] 前层和反文件层每个都可以在沙箱内实施，且覆盖于文件系统的后备层之上以使得测试移除、改变和添加文件的效果。

[0018] 本发明的一个实施例提供一种文件系统驱动器，其响应于反文件以掩盖具有在多层文件系统的高层内匹配的反文件的任何文件。更普遍地，本发明的实施例提供“反文件得知”文件传统存取控制器，其可以被实施作为数据处理系统的操作系统的完整组件，或作为

分离的文件系统驱动器。依赖于用于文件存取的文件系统存取控制器的其他程序和系统不能检测到被掩盖的文件，以至于当进行大部分的数据处理操作时不考虑被掩盖的文件。在一些实施例中，被掩盖的文件不可被除了“反文件得知”文件系统存取控制器以外的任何系统实体检测。在一些实施例中，反文件也仅可被反文件得知文件系统存取控制器检测。

[0019] 在可替换实施例中，通过修改操纵文件的 C 库调用的实施来实施反文件， 以便被掩盖的文件完全不被考虑，或被移动到受保护不被其他程序和系统存取的特殊的存储位置。反文件可以是标识将被掩盖的文件的任何逻辑实体——诸如被存储在数据库表中作为串的文件名，或适用于在文件系统中存储文件名的任何其他机制。

[0020] 在本发明的一个实施例中，在创建其中反文件层覆盖第一组文件（后备层）且其本身被（前层）覆盖的多层文件系统之后，嵌套层的第二级可以被用于创建该多层文件系统以及另一反文件层和可选的另一前层的逻辑组合。即，使用本发明，任何多层文件系统可以被视为后备层，对其可施加添加的层（反文件层和 / 或前层）。可以将此过程重复所需的次数，使用嵌套的层以研究文件和反文件的各种逻辑组合。

[0021] 不同于先前的测试文件移除效果的方法，每个上述层仍然可以被（反文件得知程序，或通过移除反文件）检查以标识具体文件系统层内的各种文件。可以通过移除反文件来相对容易地消除掩盖的效果。

[0022] 在本发明的一个实施例中，创建其中第一文件与其反文件组合以掩盖第一文件的多层文件系统的处理包括将沙箱覆盖到“后备连编 (backing build)”的元素上的步骤。“后备连编”包括包含了第一文件的一组稳定的源代码组件。具体地，根据本实施例的后备连编（后备文件系统层）包括一组程序资源、相关头文件和支持沙箱的库。由管理员和 / 或锁机制控制对该后备层的写访问，以实现源代码的稳定性——即防止多个开发者造成的不一致的同时更新。一组连编工具（编译器、解译器等等）可以处理后备文件系统层和一个或多个沙箱的多层组合以生成这组程序文件的新的可执行版本。来自多个开发者的多个沙箱可以被应用于单个后备连编，然后被处理以创建可执行代码。由于被掩盖的文件和反文件对连编工具来说不可见，因此新的可执行版本忽略被样的文件和反文件。

[0023] 本发明的掩盖机制具有类似于可逆文件删除机制的效果，使能测试移除第一文件的效果。通过在中央服务器系统保持的一组程序文件上覆盖包含反文件的沙箱，程序开发者可以测试移除文件的效果，而无需进行复杂的文件移除及在中央服务器重建处理。通过使得开发者能够从中央服务器取出仅他们的开发所需的那些文件并测试，可以在不需要每个开发者的本地数据处理系统上的大量磁盘空间的情况下实施本发明。

[0024] 本发明的另一实施例提供一种用于运行时间系统管理环境的文件系统存 取控制器。如在上述用于在开发环境中使用的实施例中，该文件系统存取控制器响应于反文件以便匹配后备层内的第一文件的、反文件层内的文件引用有效地使文件系统存取控制器掩盖第一文件的存在。

[0025] 本发明的另一方面提供一种数据处理系统，包括 ：

[0026] 数据处理单元 ；

[0027] 数据存储单元 ；

[0028] 包括所述数据存储单元内的数据的逻辑排列的文件系统，所述文件系统包括第一文件系统层内的第一文件 ；

[0029] 用于在所述文件系统的反文件层中创建所述第一文件的反文件的装置，所述反文件包括对所述第一文件的引用；以及

[0030] 第一系统存取控制器，用于对于至少一个请求者管理文件系统存取，包括：用于标识包含至少该反文件层和该后备层的、在逻辑文件系统中的一组逻辑上区别的层的装置；用于标识所述第一文件系统层内第一文件与所述反文件层内匹配的反文件之间的匹配的装置；以及用于响应于标识匹配而向所述至少一个请求者掩盖所述第一文件的存在的装置。

[0031] 本发明的另一方面提供一种用于模拟从数据处理系统内的文件系统移除第一文件的效果的方法，该数据处理系统包括上述的文件系统存取控制器，该方法包括步骤：

[0032] 在逻辑文件系统中创建包含至少后备文件系统层和反文件层的一组逻辑上区别的层；

[0033] 给后备文件系统层分配第一组文件；

[0034] 在所述反文件层中创建对所述第一组文件的第一文件的引用；以及

[0035] 响应于存取所述第一文件的请求，执行所述文件系统存取控制器以控制文件系统存取。

[0036] 可以在计算机程序代码中实施本发明的方面，并可以作为包括被记录在计算机可读记录介质上的程序代码的计算机程序产品而可得到本发明的方面。可以用于经由数据传输介质诸如经由因特网而传输得到该程序代码。

附图说明

[0037] 作为示例，下面参考附图更具体地描述本发明的实施例，其中：

[0038] 图 1 是本发明可以在其中实施的分布式软件开发环境的示意表示；

[0039] 图 2 是根据本发明的实施例，响应于尝试读取文件而由文件系统驱动器实施的方法的步骤的示意流程图表示；

[0040] 图 3 示出响应于编辑文件的尝试而由文件系统驱动器实施的方法的步骤；

[0041] 图 4 示出响应于创建文件的请求而由文件系统驱动器实施的方法的步骤；

[0042] 图 5 是根据本发明的实施例，用于列出目录中的文件的方法的示意表示；以及

[0043] 图 6 示出响应于删除文件的尝试而由文件系统驱动器实施的方法的步骤。

具体实施方式

[0044] 图 1 示出包括一些数据处理系统 20、30、40 的分布式软件开发环境 10。系统 20、30、40 经由通信网络（未示出）而被连接。在网络中可以有任何数量的数据处理系统，且受制于这些系统具有足够的存储器和处理功率来适应于程序代码开发的需求，这些系统不局限于任何具体类型的计算机硬件。该网络可以使 LAN 或 WAN，或因特网，不局限于具体的网络配置或通信协议。该网络可以包括有线和 / 或无线连接。

[0045] 图 1 所示的具体的数据处理网络实现了客户端 - 服务器结构，其中多个客户端系统 20、30 每个经由网络与服务器系统 40 通信。每个客户端系统包括处理单元 50、诸如 RAM 60 的系统存储器、非易失性存储器 70、网络接口 80 和使能用户经由键盘 100、鼠标 110 和显示屏幕 120 交互的输入 / 输出接口 90，。每个系统包括操作系统 130，其包括或接口于

(interface) 文件系统存取控制器 140。该文件系统存取控制器使得操作系统能够对文件系统进行读和写。操作系统提供对于被存储在文件系统中的多个应用程序和“中间件程序”的支持。中间件程序是应用程序支持的基础设施产品，其当被提供在应用程序和操作系统之间时能够简化应用程序编程。

[0046] 在图 1 的示范实施例中，数据处理系统 20、30、40 形成分布式软件开发环境的部分，且应用程序包括开发工具 150，诸如应用程序模拟器和代码生成器、源编辑器、源调试器和包括编译器或解译器的编译工具。上述特征是用于开发环境中的传统数据处理系统的公知特征。在第一示范实施例中，正 被开发的程序是复杂的中间件程序。

[0047] 图 1 的具体的分布式软件开发环境支持使用一个或多个“后备连编”200、210 和“沙箱”160、170、180 的开发。如上所述，后备连编是只有管理员批准或在锁管理控制下才可以更新以保证代码稳定性的一组程序代码资源。如果大组开发者每个都能够对代码库同时地或独立地进行改变，则将不可能保证所有开发者正工作于每个程序组件的最近版本，且这可能导致不能通过简单的同步来解决的不一致。使用后备连编避免了这个问题。如上所述，沙箱是与其他沙箱隔离且与其他开发者隔离的私人程序开发环境。可以在开发者的隔离的沙箱内开发代码，然后将其与预存在的一组计算机程序资源组合。

[0048] 当在系统 30 上的沙箱 180 内工作时，开发者能够从服务器系统 40 的盘式存储器中下载感兴趣的工程（例如，“工程 11”）所需的后备连编 210 的元素。典型地，正工作在诸如应用程序服务器或消息中间件产品的、大型应用程序支持的基础设施产品上的开发者将仅希望下载整个程序的小部分。该开发者写新的代码，包括在沙箱 180 内创建新文件和改变现有文件。开发工具 150 包括编译器和其他连编工具，因此开发者能够在沙箱 180 内编译新代码与下载的后备连编的元素的组合。如果连编成功且相对于该连编进行的测试给出令人满意的结果，在沙箱中开发的程序代码可以被上传到服务器系统，在该服务器系统中，管理员能够进行所需的查验并创建合并该新代码的更新的后备连编。这种分布式开发解决方案是本领域公知的，且是用于测试添加和改变程序文件的效果的令人满意的解决方案。

[0049] 然而，使用传统的解决方案，当试图评估移除文件的效果时出现了问题。这是由于难以确定任一各种开发工具或其他程序是否暗中引用了开发者想要视为已经被移除的文件。仅列出开发者想要视为“不存在”的文件不是令人满意的解决方案，因为开发者可能无法标识一些引用了名义上不存在的文件的程序。（如果仅当需要对其引用时才建造所引用的文件名，这种引用文件的标识可能尤其难）。修改每个各种开发工具和其他应用程序以无视所列（名义上不存在）文件是不现实的。考虑这些问题，目前开发者需要在实际移除程序文件之后进行全部程序的重建。由于对于大型的应用程序支持基础设施产品该重建可能花费数小时来完成，因此需要更好的解决方案。

[0050] 根据本发明的第一实施例，开发者也能够评估移除文件的效果。这是通过在多层文件系统内创建区别的 (distinct) 层——至少包括后备层 (back layer) 和反文件层 (anti-file layer)——以及在反文件层内创建对应于后备层的第一文件的反文件来实现的。文件系统存取控制器能够标识区别的层，并标识在第一文件及其反文件之间的匹配。该文件系统存取控制器解译反文件作为掩盖 (mask) 第一文件存在的指令。即，文件系统存取控制器能够响应于尝试经由文件系统存取控制器来存取第一文件的任何程序或数据处理系统而否定第一文件的存在。当文件系统存取控制器掩盖第一文件的存在时，能够对得到

的文件逻辑组合测试各种操作。

[0051] 根据本发明的第一实施例的在客户端系统上创建的新的逻辑文件系统包括下列程序资源：

[0052] ●所保存在目录“/back”中的、一组下载的程序文件（即，从包括预存在的资源代码文件、头文件和相关资源的预存在的“后备连编”中选择的一组文件，以下称为“后台(background)资源”）；

[0053] ●包括对所选的一组文件的第一文件的引用的至少一个反文件，反文件被存储在目录“/anti”内；以及

[0054] ●被保存在目录“/front”内的一组添加的文件和文件改变（以下称为“新资源”）。

[0055] 为当前文件系统的文件系统驱动器给出指向这三组资源的一组指针，以及这些资源应该被保存在“/back”、“/anti”和“/front”目录中的哪一个的一种标识。这组指针有效地布满(populate)了该新的逻辑文件系统。在该第一个实施例中，在没有复制任何文件的情况下如此创建了新的逻辑文件系统，且这三组资源被保存在三个独立的逻辑目录中，而下层(underlying)文件所有都可以被保存在物理数据存储器的相同区域中。

[0056] 当连编工具或某其他的应用程序想要存取这一组新的程序代码组件时，文件系统驱动器检查对应于三组资源的三组逻辑目录的这一组指针。由文件系统驱动器实现的具体的处理序列取决于所请求的操作，且下面参考图2至6描述一些示范操作。对于每个示范操作，标识新资源为存在于逻辑文件系统中，而没有任何来自反文件的掩盖效果。由文件系统存取控制器标识该反文件，并将其与后台资源比较。匹配所标识的反文件的、后台资源内的任何文件被文件系统存取控制器声明为不存在于逻辑文件系统。不与反文件匹配的、后台资源内的任何文件被标识为存在于新的逻辑文件系统。以此方式，文件系统驱动器（或另一文件系统存取控制器）通过考虑一个或多个反文件 并解译这些反文件为掩盖在后备层中对应文件存在的指令，来控制对于逻辑文件系统内的文件的存取。

[0057] 在一个实施例中，反文件仅仅是被保存在新的逻辑文件系统的专用“/anti”目录中的传统文件名。后台资源保存在‘/back’目录中以及新资源保存在‘/front’目录中。文件系统存取控制器解译该目录名为逻辑层，其中前层(front layer)覆盖反文件层和后备层的组合，且其中反文件层覆盖后备层。在本实施例中，反文件匹配在后备层中对应的文件的文件名。反文件的文件名可以与后备层中的文件的名称一样，或者可以用特殊的字符（例如，诸如“-”字符）置于反文件的名称前面，以简化它们与传统文件的不同。

[0058] 在后一实施例的第一个示例中，有四个文件组成第一组程序文件。文件名是“test.c”、“newlibrarycode.c”、“oldlibrarycode.c” 和“librarycode.h”。 系统用户或管理员想要确认不再需要文件“oldlibrarycode.c”，而尚没有从代码库中移除该文件。解决方案是创建新的逻辑文件系统，包括文件：“test.c”、“newlibrarycode.c”、“librarycode.h”和反“oldlibrarycode.c”，其中，用被放于目录“/anti”中的文件“oldlibrarycode.c”（或仅仅文件引用）代表文件反“oldlibrarycode.c”。其他文件以其原始名称被放于目录“/back”中。

[0059] 文件系统驱动器解译反“oldlibrarycode.c”作为掩盖“oldlibrarycode.c”的指令，因此由文件系统驱动器向其他程序声明的新文件系统的内容包括三个文件：“test.

c”、“newlibrarycode.c”和“librarycode.h”。先前的文件“oldlibrarycode.c”仍然存在于下层文件系统中，但使用普通目录存取 API 的程序将不会检测到它或已经由其反文件掩盖的任何其他文件。

[0060] 可以对于该新的逻辑文件系统运行连编处理，且文件系统驱动器将持续掩盖所要求的文件。如果连编不成功，则开发者通过简单地移除掩盖其他文件的反文件，能够容易地返回到在明显的文件移除之前的代码库。

[0061] 图 2 示出根据本发明的实施例，响应于计算机程序尝试读文件而由文件系统驱动器实施的方法的步骤。计算机程序可以是使用文件系统驱动器的文件存取 API 来请求存取新的多层逻辑文件系统内文件的任何程序。当文件系统驱动器接收 300 读请求时，由驱动器进行 310 关于所请求的文件是否在逻辑文件系统的前层（在本实施例中，“/front”目录）中的第一检查。如果文件是在前层中，则不需要检查掩盖的反文件，且从前层读 320 文件。

[0062] 然而，如果所请求的文件不是在前层中，进行 320 关于在反文件层（“/anti” 目录或文件）中是否存在所请求的文件的名称或所请求的文件的父母 (parent) 的第二检查。该语境中的父母仅仅是逻辑文件系统的层次中的更高层节点。例如，如果所请求的文件被标识为“/ancestor1/ancestor2/child”，则在此的“ancestor1”和“ancestor2”都被成为文件“child”的父母。如果在反文件层中找到所请求的文件的名称或父母，则文件系统驱动器生成 340 “文件未找到”错误消息，并将该消息送回请求的程序。即，文件系统驱动器声明，所请求的文件不存在于逻辑文件系统内。文件系统驱动器能够标识所请求的文件的存在，但文件系统驱动器向请求的程序掩盖所请求的文件的存在。

[0063] 如果所请求的文件不具有反文件层内的匹配的文件名（其本身的名称或父母），则进行 350 关于所请求的文件是否在后备层中的第三检查。该后备层是被反文件层和前层覆盖以形成多层逻辑文件系统的文件系统或目录。如果在该阶段确定 350 所请求的文件在后备层中，则从后备层读取 360 文件。如果该确定 350 得到否定的结果，则文件系统驱动器生成 340 “文件未找到”错误消息，并将该消息返回到请求的程序。

[0064] 文件系统驱动器响应于检查文件上的许可的请求（而不是对于文件内数据的读访问的请求），实施与图 2 所示的流程相同的流程。如果程序想要检查许可的文件是在多层逻辑文件系统的前层中的，则文件系统驱动器便利检查前层的许可。如果文件或文件的父母不在前层中而被引用到反文件层中，则文件系统驱动器声明“文件未找到”。如果文件不在前层中，且在反文件层中没有相关的引用，则进行关于文件是否在后备层中的检查，然后，如果是，检验后备层的许可。

[0065] 图 3 是响应于计算机程序尝试编辑多层逻辑文件系统内的文件而由文件系统驱动器进行的方法的步骤的示意表示。文件系统驱动器接收 400 编辑请求，并确定 410 在逻辑文件系统的前层中是否可得到所请求的文件。如果该文件存在于前层中，则文件系统驱动器许可 420 请求的程序编辑前层内的该文件。

[0066] 如果所请求的文件不在前层，则文件系统驱动器确定 430 反文件层是否包含对所请求的文件的引用，或是否包含对所请求的文件的父母的引用。如果在反文件层中找到任何这种引用，则文件系统驱动器生成 440 “文件未找到”错误并将其送回请求的程序。

[0067] 如果所请求的文件不在前层也没有被引用于反文件层中，则文件系统驱 动器确定 450 该文件是否在多层逻辑文件系统的后备层中。如果该确定是否定的，则该文件不存

在于逻辑文件系统中，因此文件系统驱动器生成 440 “文件未找到”错误。如果该文件存在于后备层中，文件系统驱动器从后备层中复制 460 所请求的文件到前层中，然后许可 420 在前层内编辑。

[0068] 响应于程序对于改变文件许可的尝试，文件系统驱动器实施图 3 所示的等同流程。如果相关文件在多层逻辑文件系统的前层中，文件系统驱动器许可在前文件系统层内改变文件的许可。注意，在本实施例中，在沙箱内正进行随后可能或可能不被应用于后备联编的任何这种更新。因此，不必须在此时检查程序或正请求改变的用户的权限级 (authority level)。如果相关文件不在前层中，进行关于文件是否被引用于反文件层中的检查，且这确定是否通过生成“文件未找到”错误来掩盖文件的存在。如果文件既不在前层中也没有被引用在反文件层中，则确定文件是否存在于后备层中。如果在后备层中，文件被复制到前层，然后在前层中改变许可。

[0069] 响应于对于改变目录许可的请求，也实施图 3 所示的等同流程。如果该目录既不在前层中也没被引用在反文件层中，但存在于后备层中，则在前层中创建目录，然后可以在前层中改变目录的许可。

[0070] 图 4 示出响应于创建文件或目录的请求而由文件系统驱动器进行的方法的步骤。当文件系统驱动器接收 500 创建文件或目录的请求时该处理开始。由文件系统驱动器进行 510 关于将被创建的文件的紧接的父母是否已经在多层逻辑文件系统的前层中的检查。如果父母已经存在于前层中，则文件系统驱动器许可 520 在前层中创建文件或目录。

[0071] 如果父母不在前层中，则进行 530 关于是否有任何文件或目录的父母被引用在反文件层中的检查。如果父母被引用在反文件层中，文件系统驱动器将如上所述掩盖父母的存在，并生成“文件未找到”错误 560。这阻止程序在逻辑上不存在的目录中创建文件或子女。

[0072] 如果反文件层的检查 530 未标识任何对将被创建的文件或目录的父母的引用，则进行 550 关于文件或目录的紧接的父母是否存在于后备层中的检查。如果紧接的父母存在于后备层中，则父母被复制 540 到前层，且然后可以在前层中创建 520 文件或目录。如果紧接的父母不在前层或后备层中，文件系统驱动器将生成“文件未找到”错误 560。这阻止了在不存在的目录中创建文件或目录。

[0073] 图 5 示出用于使用根据本发明的反文件得知 (aware) 文件系统驱动器、列出目录中的文件的方法的步骤序列。当文件系统驱动器接收 600 列出在多层逻辑文件系统的目录中的文件的请求时处理开始。文件系统驱动器初始地从逻辑文件系统的后备层中取出 610（如果有的话）所指名的目录的相关文件的列表。然后文件系统驱动器取出 620 在反文件层中的文件和目录的列表，并检查 630 对文件或目录在反文件层和后备层之间的任何匹配的引用。然后，该文件系统驱动器从初始的列表中移除 640 任何在反文件层中具有相关条目（匹配的文件名或父母名）的文件或目录。这创建了编辑后的列表。然后文件系统驱动器取出 650 在前层中的文件和目录列表，然后将这些添加到列表中以创建倒数第二个列表。注意，不考虑在反文件层中是否有匹配的条目，就添加在前层中的条目；前层优先，并覆盖反文件层，而反文件层优先于下面的后备层。然后文件系统驱动器从倒数第二个列表中移除 660 任何重复的条目以创建最终列表。向请求者返回 670 最终的列表。

[0074] 图 6 示出响应于删除文件的请求而由文件系统驱动器实施的方法的步骤序列。当

文件系统驱动器接收700删除请求时处理开始。文件系统驱动器确定710文件是否存在于前层中。如果存在于前层中，文件系统驱动器许可从前层删除720文件，然后检查770文件或父母是否被引用于反文件层中。如果文件或父母被引用于反文件层中，处理结束780。如果文件或父母没被引用于反文件层，则向反文件层添加760对该文件的引用，并结束处理。如果文件不在前层中，则进行730关于文件或文件的父母是否被引用于反文件层中的检查。如果已经存在这种引用，该处理生成“文件未找到”错误740，因为该文件已经被掩盖了。如果文件及其父母没被引用于反文件层中，则进行750关于文件是否存在于后备层中的检查。如果该确定是肯定的，则向反文件层添加760对该文件的引用。然后，这将产生对尝试存取该文件的其他程序掩盖所引用的文件的效果。如果确定750文件不存在于多层逻辑文件系统的任何层中，处理生成“文件未找到”错误740。

[0075] 可以响应于删除目录的请求实施与图6所示的流程类似的流程。然而，当处理删除目录的请求时，可以移除多余的文件引用以简化反文件层。这是通过在向反文件层添加对目录本身的引用之前从反文件层中移除对该目录的子女(child)的所有引用来实现的。

[0076] 如前所述，本发明的文件掩盖机制不局限于在分布式开发环境中的沙箱 和后备连编。本发明等同地可用于诸如系统管理员想要测试从运行时数据处理环境中移除一个或多个文件的效果的其他环境。当一组计算机程序文件的计划的更新需要测试移除至少一个文件的效果时，可以使用本发明。

[0077] 为了有助于理解本发明，下面突出本发明与先前出版物的一些区别。虽然反文件不是广泛使用的编程概念，但至少一个出版物提出了使用“反文件”目录条目来使删除数据文件用于同步一组指名的目标文件与一组指名的源文件的目的。

[0078] 转让给国际商业机器公司的US 6473767公开了支持每个都维护作为多个目录条目的数据文件列表的多个目录的文件系统。该系统包括用于创建“反文件”目录条目的装置，该“反文件”目录条目每个都具有指示没有对应于该条目的真实的数据文件的属性。在US 6473767中，当在同一目录中存在相同名称的“反文件”和数据文件时，删除真实的数据文件。这种自动的删除被用作同步源和目标之间的文件组的机制，其中永久的删除是适当的。US6473767要求在目录之间移除数据文件或“反文件”以控制是否及何时删除任何数据文件——因为根据US 6473767的“反文件”将永久地消除对应的数据文件，如果它们都存在于相同的目录中的话。

[0079] 本发明不同于US 6473767之处在于，在多层逻辑文件系统中实现第一文件的反文件，其中，反文件被用于掩盖该第一文件的存在。该反文件不永久地删除第一文件，而是文件系统存取控制器解译该反文件作为向尝试经由文件系统存取控制器来存取第一文件的任何程序可逆地掩盖第一文件的存在的指令。例如，在沙箱覆盖源代码树的开发环境中，由反文件掩盖的文件并不永久地从原始源代码树中移除。而是，在没有不可逆的文件删除的情况下，该掩盖使能更简单的连编处理，且因此使能有效地测试移除文件的效果。

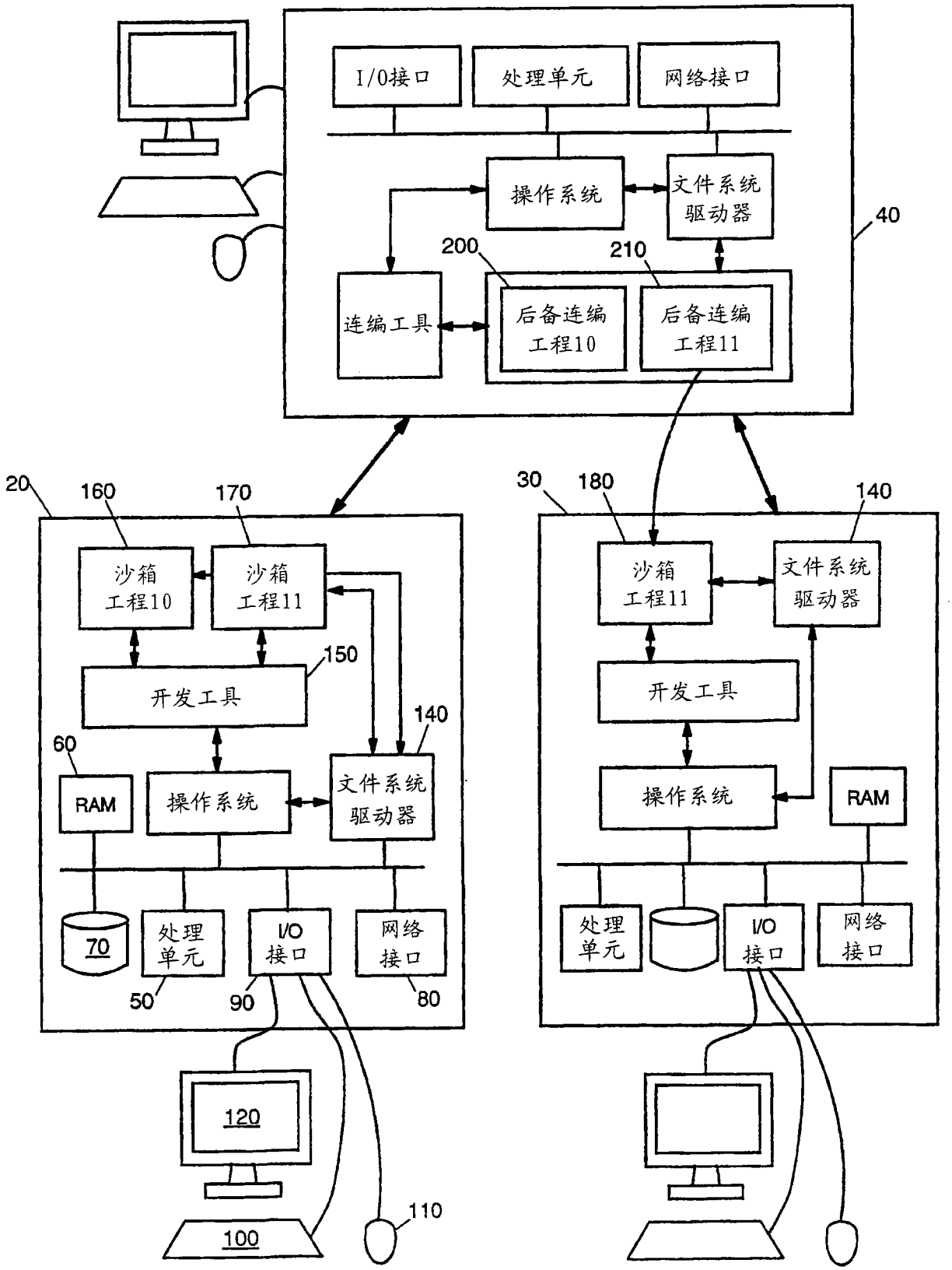

I/O接口
处理单元
网络接口
操作系统
文件系统驱动器
40
200
210
连编工具
后备连编工程10
后备连编工程11
20
160
170
沙箱工程10
沙箱工程11
150
开发工具
140
60
RAM
操作系统
文件系统驱动器
70
处理单元
I/O接口
网络接口
50
90
80
120
100
110
30
180
140
沙箱工程11
文件系统驱动器
开发工具
操作系统
RAM
处理单元
I/O接口
网络接口

图 1

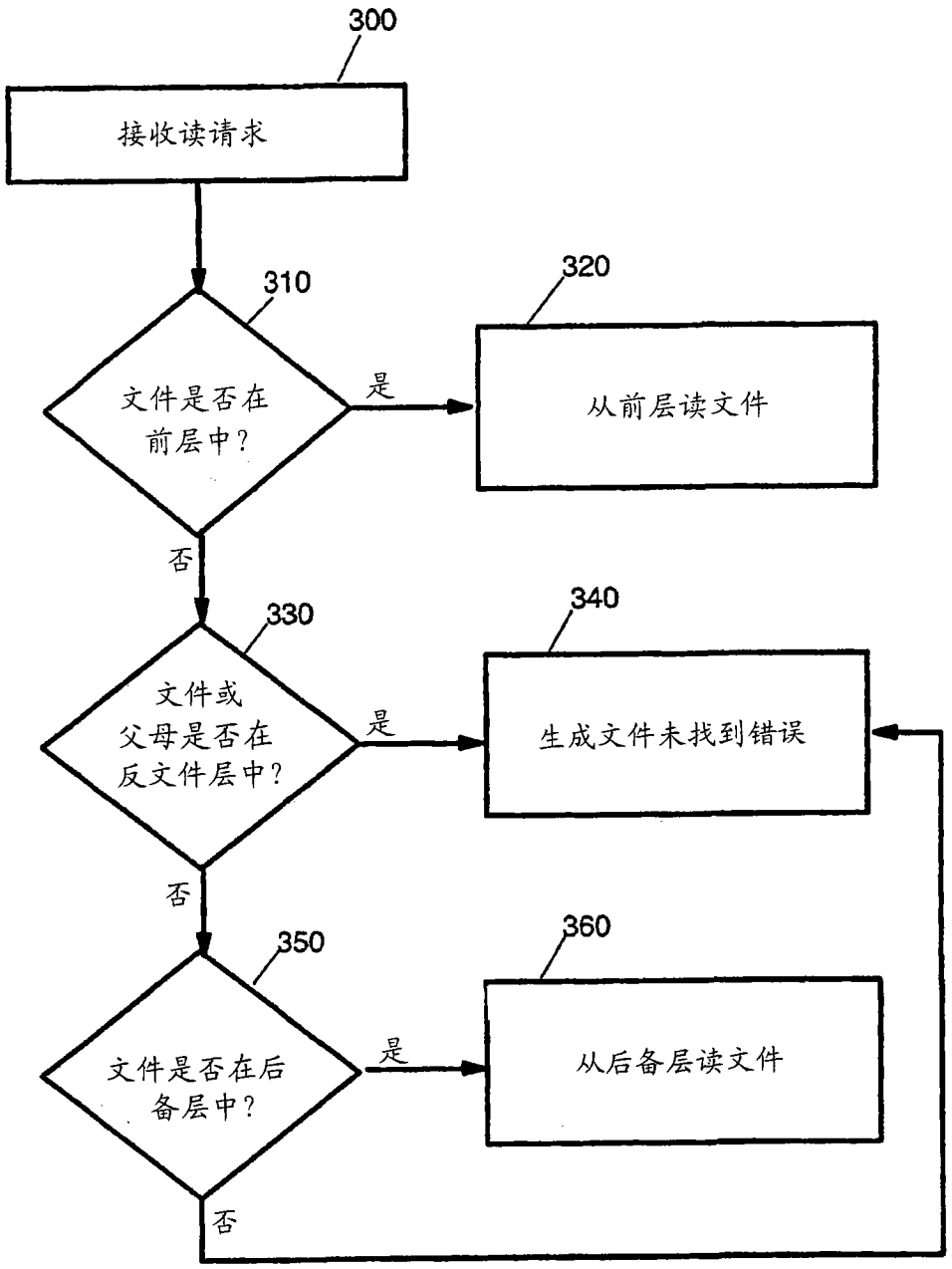

300
接收读请求
310
文件是否在前层中？
是
320
从前层读文件
否
330
文件或父母是否在反文件层中？
是
340
生成文件未找到错误
否
350
文件是否在后备层中？
是
360
从后备层读文件
否

图 2

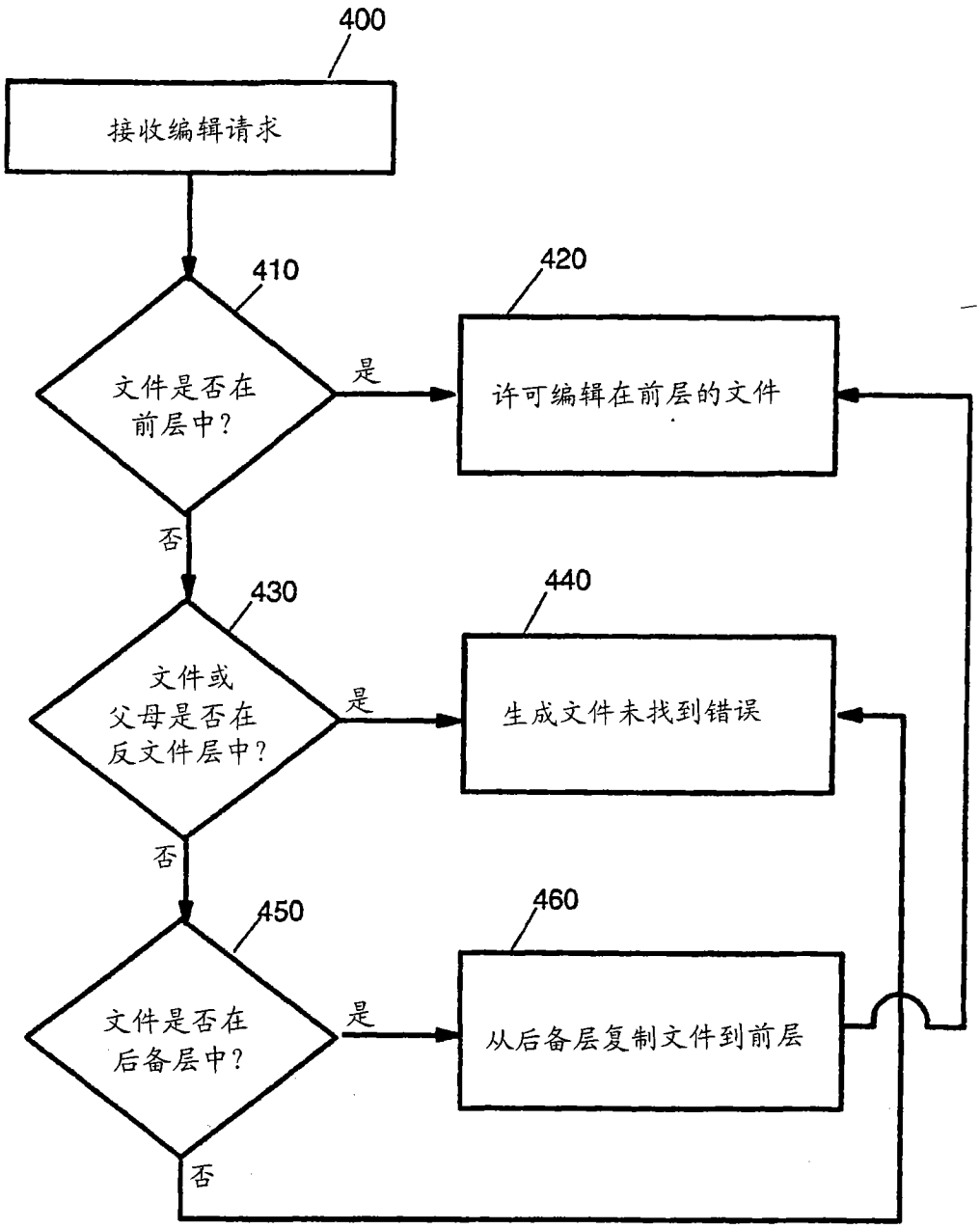

400
接收编辑请求
410
文件是否在前层中？
是
420
许可编辑在前层的文件
否
430
文件或父母是否在反文件层中？
是
440
生成文件未找到错误
否
450
文件是否在后备层中？
是
460
从后备层复制文件到前层
否

图 3

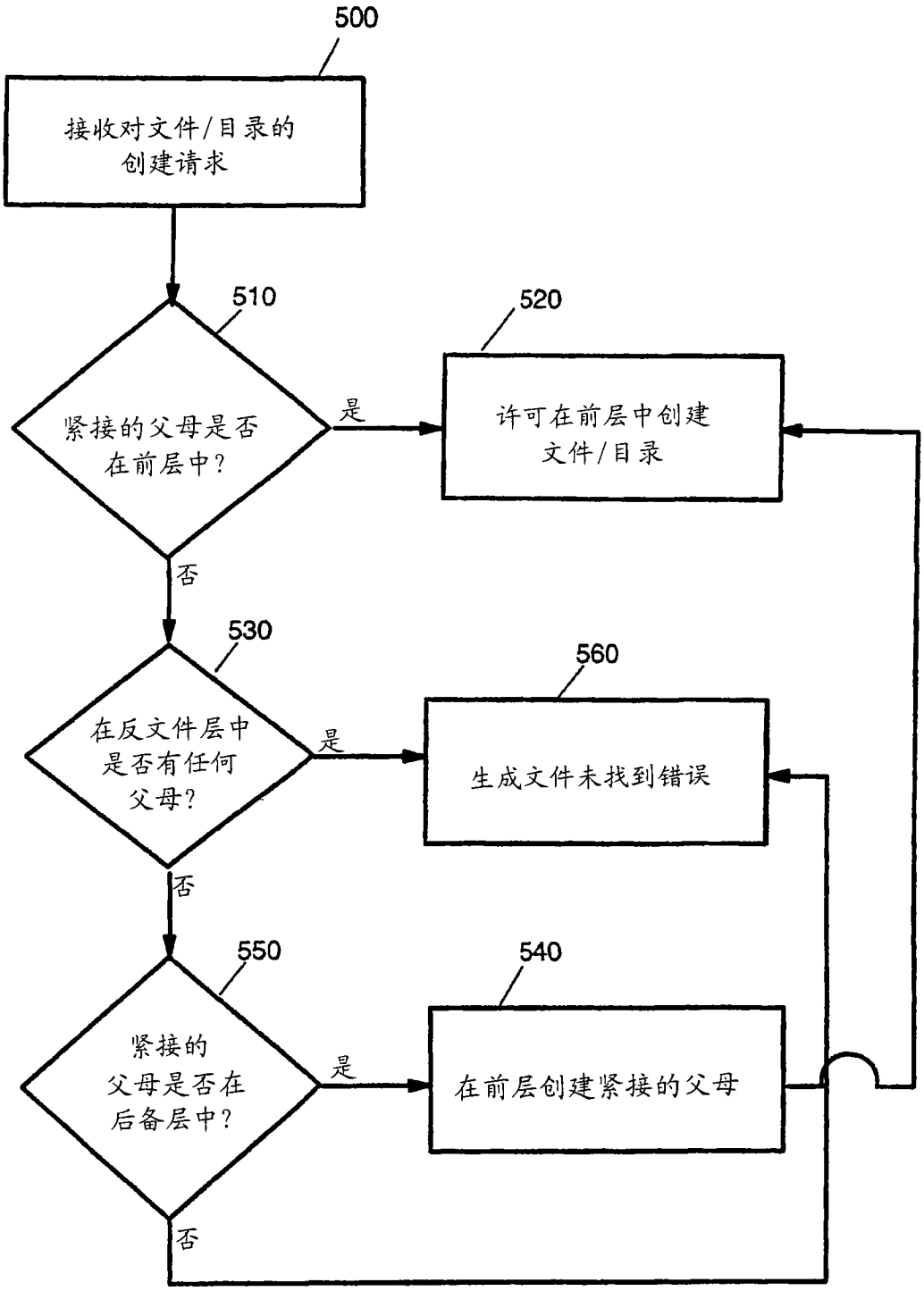

500
接收对文件/目录的创建请求
510
紧接的父母是否在前层中?
是
520
许可在前层中创建文件/目录
否
530
在反文件层中是否有任何父母?
是
560
生成文件未找到错误
否
550
紧接的父母是否在后备层中?
是
540
在前层创建紧接的父母
否

图 4

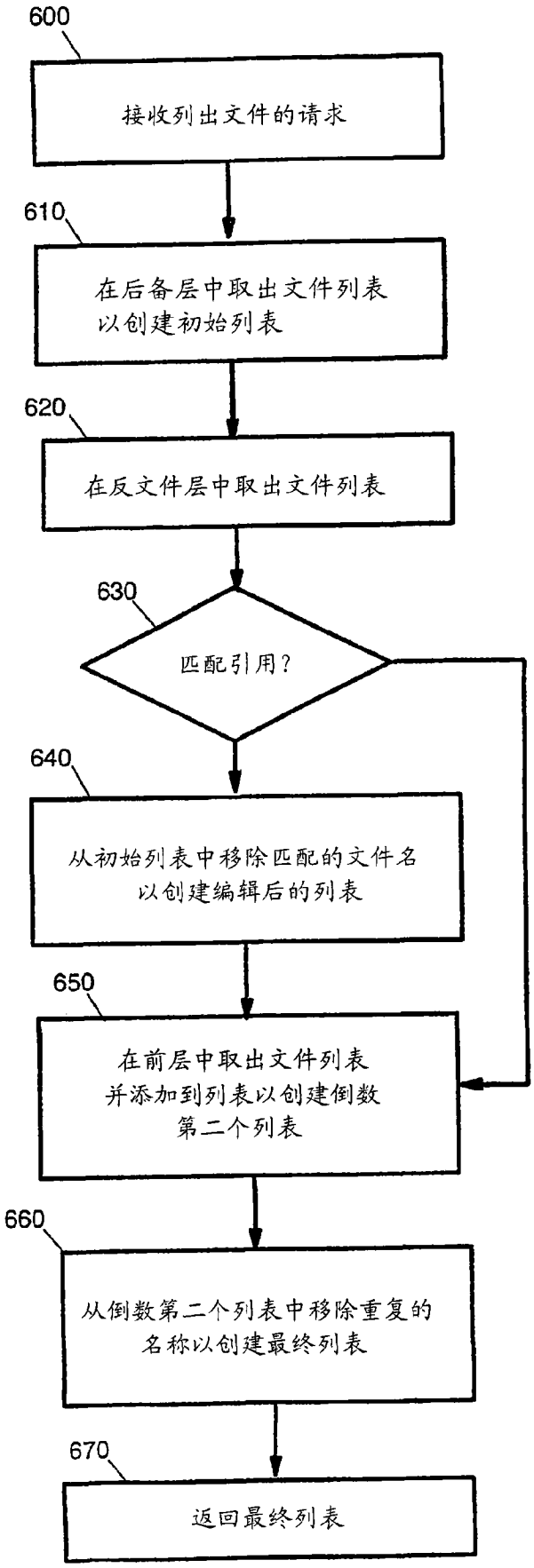

600
接收列出文件的请求
610
在后备层中取出文件列表
以创建初始列表
620
在反文件层中取出文件列表
630
匹配引用？
640
从初始列表中移除匹配的文件名
以创建编辑后的列表
650
在前层中取出文件列表
并添加到列表以创建倒数
第二个列表
660
从倒数第二个列表中移除重复的
名称以创建最终列表
670
返回最终列表

图 5

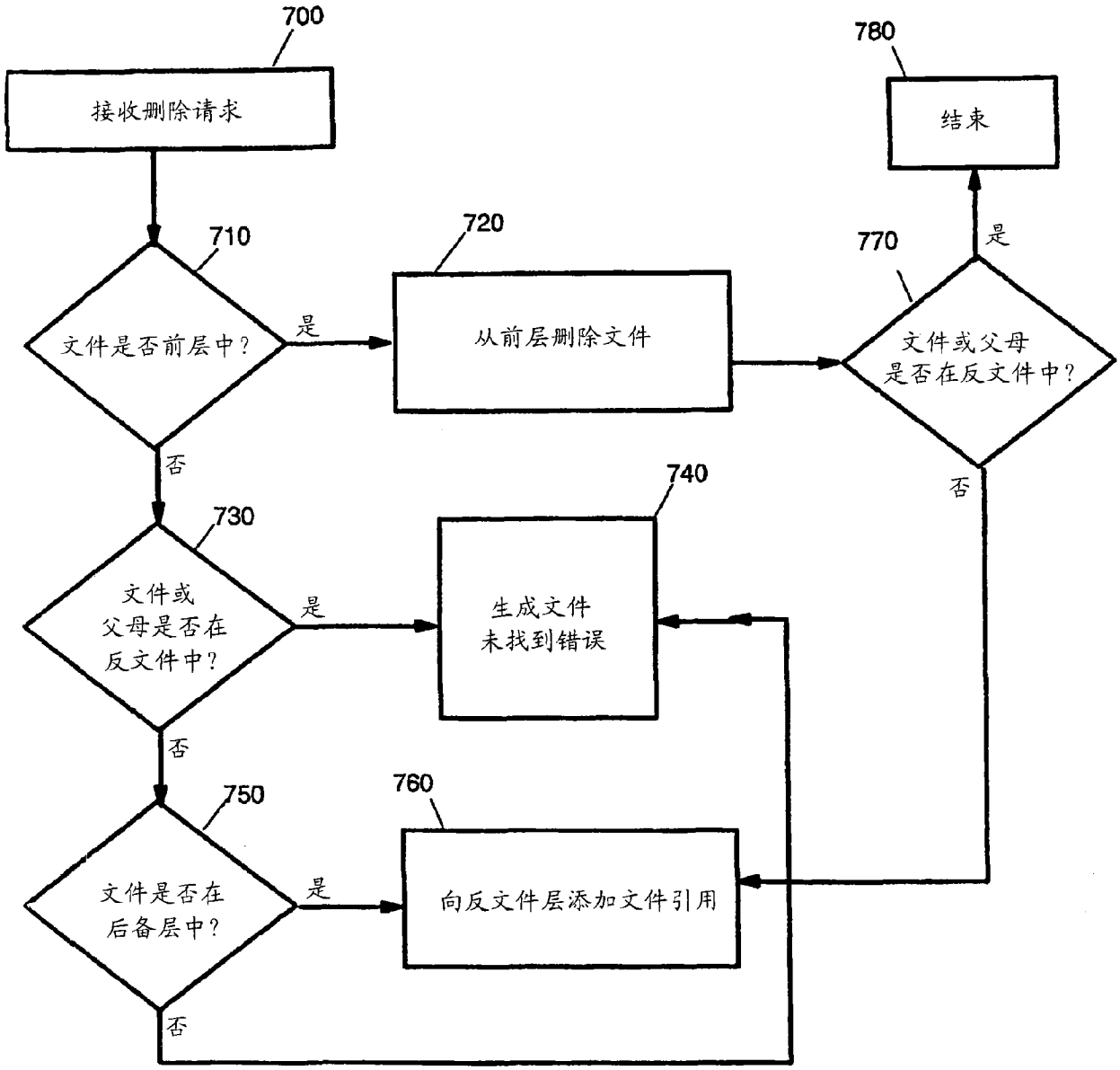

700
接收删除请求
710
文件是否前层中？
是
720
从前层删除文件
770
文件或父母
是否在反文件中？
是
780
结束
否
730
文件或
父母是否在
反文件中？
是
740
生成文件
未找到错误
否
750
文件是否在
后备层中？
是
760
向反文件层添加文件引用
否

图 6