(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 December 2010 (16.12.2010)

PCT

(10) International Publication Number
WO 2010/142537 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/EP2010/057384

(22) International Filing Date:
28 May 2010 (28.05.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/482,935 11 June 2009 (11.06.2009) US

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, North Harbour, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VINING, Robert, Guy** [US/US]; IBM Corporation, Mail Drop 9032-2 253, 9000 S Rita Rd, Tucson, Arizona 85744-0002 (US). **GOLDSMITH, Kevin, Scott** [US/US]; IBM Corporation, Mail Drop 9032-2 260, 9000 S Rita Rd, Tucson, Arizona 85744-0002 (US). **WURTH, Gregory, Paul** [US/US]; IBM Corporation, Mail Drop 9032-2 253, 9000 S Rita Rd, Tucson, Arizona 85744-0002 (US).

(74) Agent: **LITHERLAND, David, Peter**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,

[Continued on next page]

(54) Title: FILE SYSTEM LOCATION VERIFICATION USING A SENTINEL

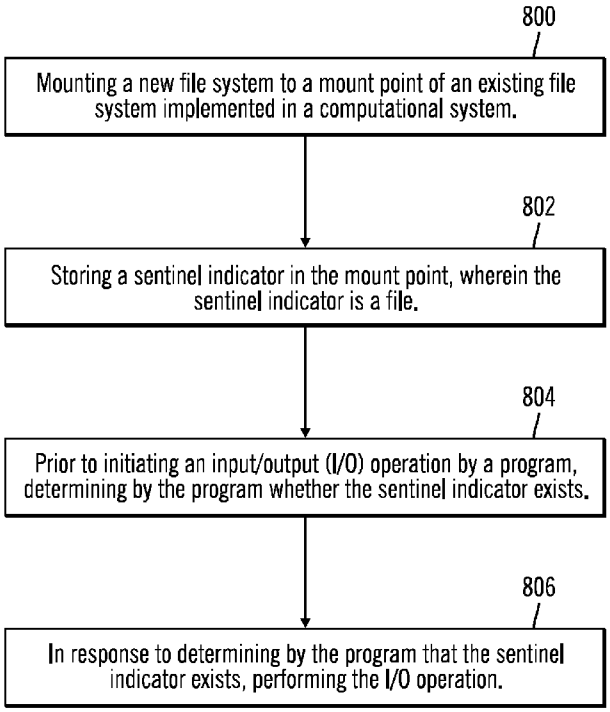

FIG. 8

(57) Abstract: A new file system is mounted to a mount point of an existing file system implemented in a computational system. A sentinel indicator is stored in the mount point. Prior to initiating an input/output (I/O) operation by a program, a determination is made by the program as to whether the sentinel indicator exists. In response to determining by the program that the sentinel indicator exists, the I/O operation is performed by the program. In certain alternative embodiments, in response to determining by the program that the sentinel indicator exists, the I/O operation is not performed by the program.

WO 2010/142537 A1

SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

FILE SYSTEM LOCATION VERIFICATION USING A SENTINEL

Technical Field of the Invention

The invention relates to a method, a system, and a computer program product for verification of a file system location using a sentinel.

Background of the Invention

In UNIX* file system hierarchies a tree structure may exist, wherein the tree structure comprises of directories and files. A directory may include other directories and files, wherein the other directories that are included in a directory may be referred to as subdirectories. The top level directory of the UNIX file system may be referred to as the root or the root directory.

Additionally, in UNIX file systems, the capability exists to mount other file systems into the root of the UNIX file system hierarchy at a "mount point" or at "mount points" within previously mounted file systems. The "mount point" is a specific directory location within the overall UNIX file system hierarchy. This mechanism allows an administrator to construct a flexible and expandable UNIX file system hierarchy which includes a number of mounted file systems that can be local or remote, wherein the remote files systems may include Network File System (NFS) mounted file systems. End users at a command prompt and application programs may see one complete UNIX file system hierarchy and may be unaware as to which specific directories are within which mounted file systems.

Summary of the Invention

Provided are a method, a system, and an computer program product in which a new file system is mounted to a mount point of an existing file system implemented in a computational system. A sentinel indicator is stored in the mount point. Prior to initiating an input/output (I/O) operation by a program, a determination is made by the program as to whether the sentinel indicator exists. In response to determining by the program that the

sentinel indicator exists, the I/O operation is performed by the program. In certain alternative embodiments, in response to determining by the program that the sentinel indicator exists, the I/O operation is not performed by the program.

In certain additional embodiments, the new file system is unmounted. Prior to initiating another I/O operation by the program, the program determines whether the sentinel indicator exists. In response to determining that the sentinel indicator does not exist, an error is returned.

In certain embodiments, the sentinel indicator is a file or a directory.

In further embodiments, the existing file system is a hierarchical file system with a root directory that is a top level file system. The mount point is a directory or a subdirectory of the root directory. Furthermore, in response to the new file system being mounted to the mount point, one or more previously existing files of the existing file system are no longer visible to the program.

Certain embodiments provide a method, comprising: mounting a new file system to a mount point of an existing file system implemented in a computational system; storing a sentinel indicator in the mount point; prior to initiating an input/output (I/O) operation by a program, determining by the program whether the sentinel indicator exists; and in response to determining by the program that the sentinel indicator exists, not performing the I/O operation.

Brief Description of the Drawings

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

FIG. 1 illustrates a block diagram of an exemplary computational device, in accordance with certain embodiments;

FIG. 2 illustrates a block diagram that shows a root directory with a mount subdirectory, in accordance with certain embodiments;

FIG. 3 illustrates a block diagram that shows a root directory with a mount subdirectory which is a mount point for a new file system, in accordance with certain embodiments;

FIG. 4 illustrates a flowchart that shows first operations implemented in the computational device of FIG. 1, wherein no sentinel indicator is used, in accordance with certain embodiments;

FIG. 5 illustrates a block diagram that shows a root directory with a mount subdirectory that has a sentinel file included in the mount subdirectory, in accordance with certain embodiments;

FIG. 6 illustrates a flowchart that shows second operations implemented in the computational device of FIG. 1 in which a sentinel file is used, in accordance with certain embodiments;

FIG. 7 illustrates a flowchart that shows third operations implemented in the computational device of FIG. 1 in which a sentinel file is used, in accordance with certain embodiments;

FIG. 8 illustrates a flowchart that shows fourth operations implemented in the computational device of FIG. 1 in which a sentinel file is used, in accordance with certain embodiments; and

FIG. 9 illustrates a block diagram of a computational system that shows certain elements that may be included in the computational device of FIG. 1, in accordance with certain embodiments.

Detailed Description

Given the possibility that any directory within a UNIX file system hierarchy may be a "mount point" or not be a "mount point", it is possible that any given specific directory location in the overall UNIX file system hierarchy may either be within a selected mounted file system or not within the selected mounted file system. A "mount point" may be in one of two possible states. In a first state, a file system is not mounted at the "mount point". In a second state, a file system is mounted at the "mount point". Mounting a file system at a "mount point" has the effect of hiding or obscuring the existing contents (i.e., files and/or subdirectories) that exist at the "mount point", and mounting the file system at the "mount point" makes only the mounted file system visible. When the file system is unmounted the previous contents may be visible once again. Furthermore, any file system may potentially be mounted at a given "mount point".

Wherever there is a possibility of mounting or unmounting of file systems, the potential exists that at any given point in time the overall UNIX file system hierarchy is not in the state that the UNIX file system was intended to be in by the UNIX file system administrator. Procedures for mounting the file systems may be manual rather than automated (for example, during the boot of the operating system). Anytime, a manual mount is required, the potential for neglecting to perform the corresponding unmount may exist. In addition, file systems may be unmounted for a number of reasons, such as to perform maintenance (i.e. a backup). In certain situations, the wrong file system may be mistakenly mounted at a given "mount point".

All of these possibilities can create difficulties for end users and applications that attempt to access files and directories in the UNIX file system hierarchy. The files or directories that are to be accessed may or may not be present in the properly mounted file systems as anticipated. However, in some cases end users and applications may be operating on known specific directory locations and may not be able to determine whether the intended file system has been mounted or not, and such situations may lead to erroneous and unexpected results. These include the following possibilities:

(i) Attempting to locate or read a file at a specific directory location which is intended to be within a mounted file system, but the file system is not currently mounted. As a result, the file cannot be located or read.

(ii) Attempting to write a file at a specific directory location which is intended to be within a mounted file system, but the file system is not currently mounted. As a result, the file is written, but a later mount of the file system obscures the file that has been written making the file inaccessible; and

(iii) Attempting to write a file at a specific directory location which is intended to be within a mounted file system, but the wrong file system is currently mounted. As a result, the file may be written, but then the UNIX administrator may unmount the wrong file system and mount the correct file system making the file that has been written inaccessible.

Certain embodiments provide a mechanism for an end user or application to ensure that the contents of a specific file system location are in an intended configuration.

In certain embodiments, when a program is dependent on a specific directory being at or within a mounted file system, the program can mandate the existence of a predetermined sentinel at the directory location. When a storage administrator is configuring the directory location within the mounted file system on behalf of the program, the administrator may be responsible for the creation of the sentinel. When the program is performing Input/Output (I/O) operations to the configured location, the program may first verify the existence of the sentinel. If the sentinel is present, the program can be assured that the location is where the administrator wanted the I/O to occur. If the sentinel is not present, the program will ensure the failure of all I/O operations.

In certain embodiments, the program is assured that the program is performing I/O in the location desired by the administrator as a result of the confirmation of the existence of the sentinel. This prevents lost time in problem determination that may be necessary if the data was not in the specified location as a result of the file system not being mounted.

FIG. 1 illustrates a block diagram of an exemplary computational device 100, in accordance with certain embodiments. The computational device 100 may comprise any suitable computational device comprising one or more of a mainframe, a personal computer, a midrange computer, a telephony device, a server, a client computer, a handheld computer, a laptop, a palmtop computer, etc.

The computational device 100 includes an operating system 102, a file system 104 and a program 106. The operating system 102 may by any exemplary operating system, such as the UNIX operating system. In an exemplary embodiment the file system 104 may comprise a hierarchical file system in which mounts are allowed on drives, directories or subdirectories. The program 106 may comprise an application that accesses files and directories stored in the file system 104.

A storage administrator 108 may interact with the computational device 100 via an exemplary command line interface or a graphical user interface and manage the file system 104. In certain embodiments, an automated computer program may perform the operations performed by the storage administrator 108.

FIG. 2 illustrates a block diagram that shows a root directory 200 with a subdirectory referred to as a mount directory 202 implemented in the file system 104 of the computational device 100, in accordance with certain embodiments. The root directory 200 may include other subdirectories 204. The mount directory 202 may include an exemplary file named "myfile" 206 and no new file system has been mounted to the mount directory 202. In such embodiments, the file named "myfile" 206 is accessible to the program 106.

FIG. 3 illustrates a block diagram that shows a root directory 300 with a subdirectory referred to as a mount directory 302 implemented in the file system 104 of the computational device 100, in accordance with certain embodiments. The root directory 300 may include other subdirectories 304. The mount directory 202 may include an exemplary file named "myfile" 306. In FIG. 3, a new file system has been mounted in the mount directory 302. In such embodiments, the file named "myfile" 306 is no longer accessible to the program 106.

FIG. 4 illustrates a flowchart that shows first operations implemented in the computational device 100 of FIG. 1, wherein no sentinel indicator is used, in accordance with certain embodiments.

Control starts at block 400, in which the file system 104 has a directory called /mount (reference numeral 302 of FIG. 3, wherein "/mount" is an exemplary notation that denotes the mount directory 302) that was created in order to be used as a mount point for a new file system for the program 106. The storage administrator 108 creates (at block 402) a file system with the intent of mounting the files system using the /mount directory 302 as a mount point

The program 106 is configured (at block 404) to perform I/O to the /mount directory 302. The program 106 is requested (at block 406) to perform a write to a file called /mount/myfile. The program 106 performs (at block 408) the requested write and since the file system was not mounted, the data is written to the /mount directory (the location where the data is written is shown by reference numeral 306 in FIG. 3).

The storage administrator 108 realizes (at block 410) that the desired new file system was not mounted and mounts the desired file system to the /mount mount point 302. The program 106 is requested (at block 412) to perform a read of a file called /mount/myfile. The read fails (at block 414) because the write occurred (at block 408) before the desired file system was mounted.

FIG. 5 illustrates a block diagram that shows a root directory 500 with a subdirectory referred to as a mount directory 502, wherein a sentinel file 504 is included in the mount directory 502, in accordance with certain embodiments. The file myfile 508 is a file within a new file system mounted at the mount directory 502. In certain exemplary embodiments, the sentinel file may be a file named sentinel.dat. The root directory 500 may include other subdirectories 506.

FIG. 6 illustrates a flowchart that shows second operations implemented in the computational device 100 of FIG. 1 in which the sentinel file 504 is used, in accordance with certain embodiments;

Control starts at block 600, in which the file system 104 has a directory called /mount 502 that was created to be used as a mount point for the program 106. The storage administrator 108 creates (at block 602) a new file system with the intent of mounting the new file system by using the /mount directory 502 as a mount point. The storage administrator 108 creates (at block 604) a sentinel file 504 with the predetermined name sentinel.dat in the /mount directory 502 on the mounted new file system.

The program 106 is configured (at block 606) to perform I/O to the /mount directory 502. The program 106 is requested (at block 608) to perform a write to a file called /mount/myfile. The program 106 tests (at block 610) for the existence of the sentinel file (/mount/sentinel.dat) 504. In certain embodiments, in which the new file system has been unmounted, the sentinel file is not found (at block 612) and the write fails. An error message is issued (at block 614) indicating the sentinel file was not found.

FIG. 7 illustrates a flowchart that shows third operations implemented in the computational device 100 of FIG. 1 in which a sentinel file 502 is used, in accordance with certain embodiments.

Control starts at block 700, in which the storage administrator 108 realizes from an error message (e.g., error message generated in block 614 of FIG 6) that a desired file system was not mounted and mounts the desired file system to the /mount 502 mount point. From block 700 control may proceed to either block 702 or block 708.

The program 106 is requested (at block 702) to perform a write to a file called /mount/myfile. The program 106 tests (at block 704) for the existence of the sentinel file 504 (/mount/sentinel.dat). The sentinel file 504 is found (at block 706) and the write succeeds

The program is requested (at block 708) to perform a read of a file called /mount/myfile. The program tests (at block 710) for the existence of the sentinel file 504 (/mount/sentinel.dat). The sentinel file is found (at block 712) and the read is successful.

In the operations shown in FIGs. 6 and 7, the storage administrator 108 created the sentinel file 504 that clearly identified that location as the intended location for I/O. The testing prior to subsequent I/O operations allow the program 106 to be certain that the program was reading from or writing to the correct location.

FIG. 8 illustrates a flowchart that shows fourth operations implemented in the computational device 100 of FIG. 1 in which a sentinel file is used, in accordance with certain embodiments.

Control starts at block 800 in which a new file system is mounted to a mount point 502 of an existing file system 104 implemented in a computational system 100. A sentinel indicator 504 is stored (at block 802) in the mount point 502. In certain embodiments, the sentinel indicator is a file and in other embodiments the sentinel indictor is a directory. Prior to initiating an input/output (I/O) operation by a program 106, a determination is made (at block 804) by the program 106 as to whether the sentinel indicator 504 exists. In response to determining by the program 106 that the sentinel indicator exists, the I/O operation is performed (at block 806) by the program 106. In certain alternative embodiments, in response to determining by the program 10 that the sentinel indicator exists, the I/O operation is not performed by the program 106.

In certain additional embodiments, the new file system is unmounted. Prior to initiating another I/O operation by the program 106, the program 106 determines whether the sentinel indicator 504 exists. In response to determining that the sentinel indicator 504 does not exist, an error is returned.

In further embodiments, the existing file system is a hierarchical file system with a root directory 500 that is a top level file system. The mount point 502 is a directory or a subdirectory of the root directory 500. Furthermore, in response to the new file system being

mounted to the mount point, one or more previously existing files of the existing file system are no longer visible to the program 106.

Therefore, FIGs. 1-8 illustrate certain embodiments in which a storage administrator 108 creates a sentinel file in the directory and on the file system where I/O is desired. When a program 106 is required to perform I/O to the directory, a test is performed to ensure the existence of the sentinel. If the sentinel is found, the I/O is carried out. If the sentinel is not found, the I/O is failed and a message or error code is presented to indicate that the target location contents are not as expected based on configurations made by the storage administrator 108.

The described operations may be implemented as a method, apparatus or computer program product using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. Accordingly, aspects of the embodiments may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, aspects of the embodiments may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a

computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java*, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

Aspects of the present invention are described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart

illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

FIG. 9 illustrates a block diagram that shows certain elements that may be included in the system 900 in accordance with certain embodiments. The system 100 may comprise the computational device 100, and may include a circuitry 902 that may in certain embodiments include at least a processor 904. The system 900 may also include a memory 906 (e.g., a volatile memory device), and storage 908. The storage 908 may include a non-volatile memory device (e.g., EEPROM, ROM, PROM, RAM, DRAM, SRAM, flash, firmware, programmable logic, etc.), magnetic disk drive, optical disk drive, tape drive, etc. The storage 908 may comprise an internal storage device, an attached storage device and/or a network accessible storage device. The system 900 may include a program logic 910 including code 912 that may be loaded into the memory 906 and executed by the processor 904 or circuitry 902. In certain embodiments, the program logic 910 including code 912 may be stored in the storage 908. In certain other embodiments, the program logic 910 may be

implemented in the circuitry 902. Therefore, while FIG. 9 shows the program logic 910 separately from the other elements, the program logic 910 may be implemented in the memory 906 and/or the circuitry 902.

Certain embodiments may be directed to a method for deploying computing instruction by a person or automated processing integrating computer-readable code into a computing system, wherein the code in combination with the computing system is enabled to perform the operations of the described embodiments.

The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the present invention(s)" unless expressly specified otherwise.

The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise.

The enumerated listing of items does not imply that any or all of the items are mutually exclusive, unless expressly specified otherwise.

The terms "a", "an" and "the" mean "one or more", unless expressly specified otherwise.

Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more intermediaries.

A description of an embodiment with several components in communication with each other does not imply that all such components are required. On the contrary a variety of optional components are described to illustrate the wide variety of possible embodiments of the present invention.

Further, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described does not necessarily indicate a requirement that the steps be performed in that order. The steps of processes described herein may be performed in any order practical. Further, some steps may be performed simultaneously.

When a single device or article is described herein, it will be readily apparent that more than one device/article (whether or not they cooperate) may be used in place of a single device/article. Similarly, where more than one device or article is described herein (whether or not they cooperate), it will be readily apparent that a single device/article may be used in place of the more than one device or article or a different number of devices/articles may be used instead of the shown number of devices or programs. The functionality and/or the features of a device may be alternatively embodied by one or more other devices which are not explicitly described as having such functionality/features. Thus, other embodiments of the present invention need not include the device itself.

At least certain operations that may have been illustrated in the figures, show certain events occurring in a certain order. In alternative embodiments, certain operations may be performed in a different order, modified or removed. Moreover, steps may be added to the above described logic and still conform to the described embodiments. Further, operations described herein may occur sequentially or certain operations may be processed in parallel. Yet further, operations may be performed by a single processing unit or by distributed processing units.

The foregoing description of various embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made

without departing from the scope of the invention, the invention resides in the claims hereinafter appended.

---

* UNIX is a trademark or registered trademark of The Open Group.

* Java is a trademark or registered trademark of Sun Microsystems, Inc.

CLAIMS

1. A method, comprising:

mounting a new file system to a mount point of an existing file system implemented in a computational system;

storing a sentinel indicator in the mount point;

prior to initiating an input/output (I/O) operation by a program, determining by the program whether the sentinel indicator exists; and

in response to determining by the program that the sentinel indicator exists, performing the I/O operation.

2. The method of claim 1, the method further comprising:

unmounting the new file system;

prior to initiating another I/O operation by the program, determining by the program whether the sentinel indicator exists; and

in response to determining that the sentinel indicator does not exist, returning an error.

3. The method of claim 2, wherein the sentinel indicator is a file.

4. The method of claim 3, wherein:

the existing file system is a hierarchical file system with a root directory that is a top level file system;

the mount point is a directory or a subdirectory of the root directory; and

in response to the new file system being mounted to the mount point, one or more previously existing files of the existing file system are no longer visible to the program.

5. The method of claim 2, wherein the sentinel indicator is a directory.

6. A computer program product for verification of a file system location, the computer program product comprising a computer readable storage medium having computer readable

program code embodied therewith, the computer readable program code when executed by a processor performing the steps of any of claims 1 to 5.

7. A system, comprising:

a memory; and

a processor coupled to the memory, wherein the processor performs operations, the operations comprising the method of any of claims 1 to 5.

8. A method for deploying computer infrastructure by integrating computer-readable code from a computer readable medium into the system, wherein the computer-readable code in combination with the system is capable of performing the method of any of claims 1 to 5.

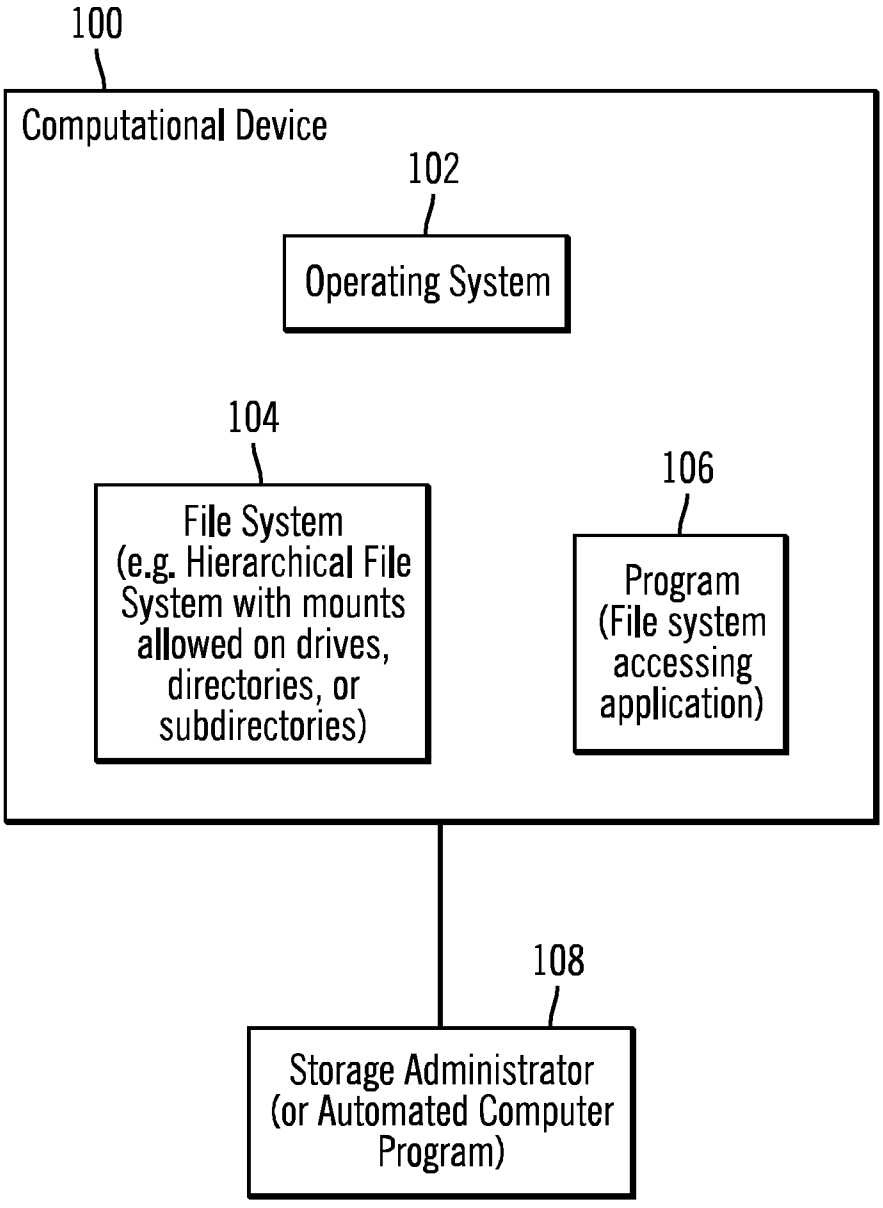
100
Computational Device
102
Operating System
104
File System
(e.g. Hierarchical File System with mounts allowed on drives, directories, or subdirectories)
106
Program
(File system accessing application)
108
Storage Administrator
(or Automated Computer Program)

FIG. 1

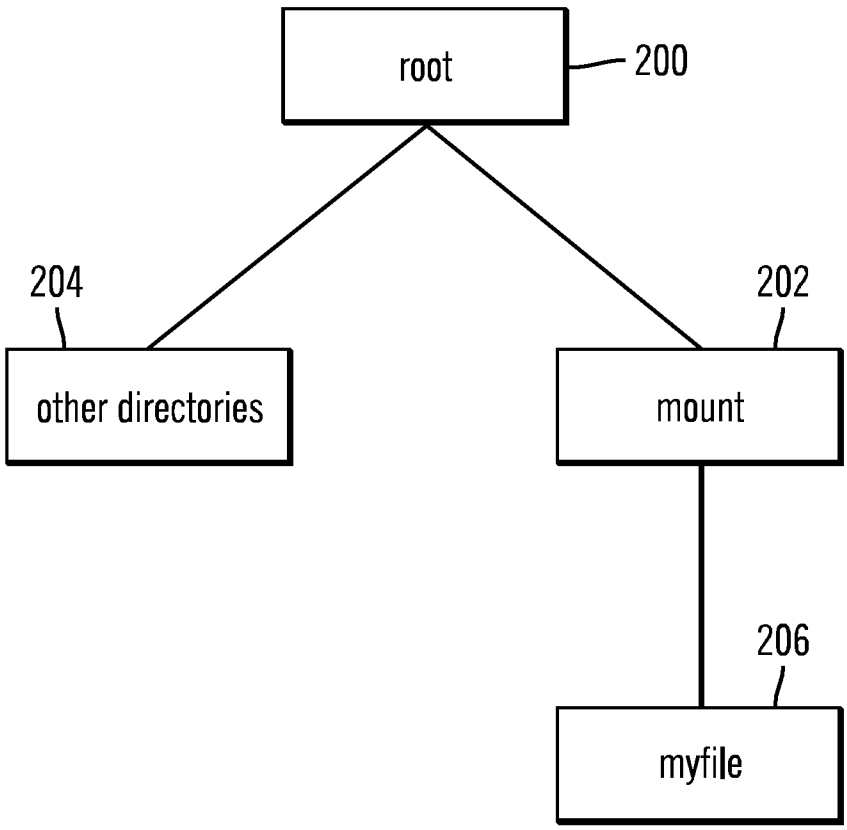

FIG. 2

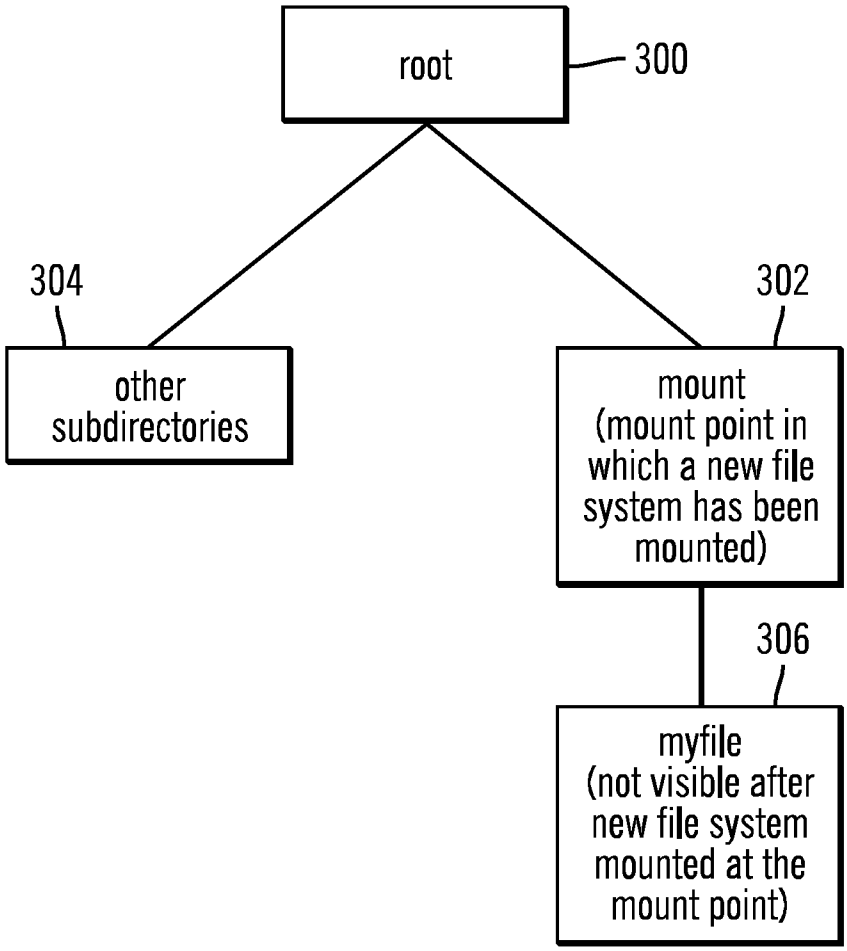
root
300
304
302
other subdirectories
mount (mount point in which a new file system has been mounted)
306
myfile (not visible after new file system mounted at the mount point)

FIG. 3

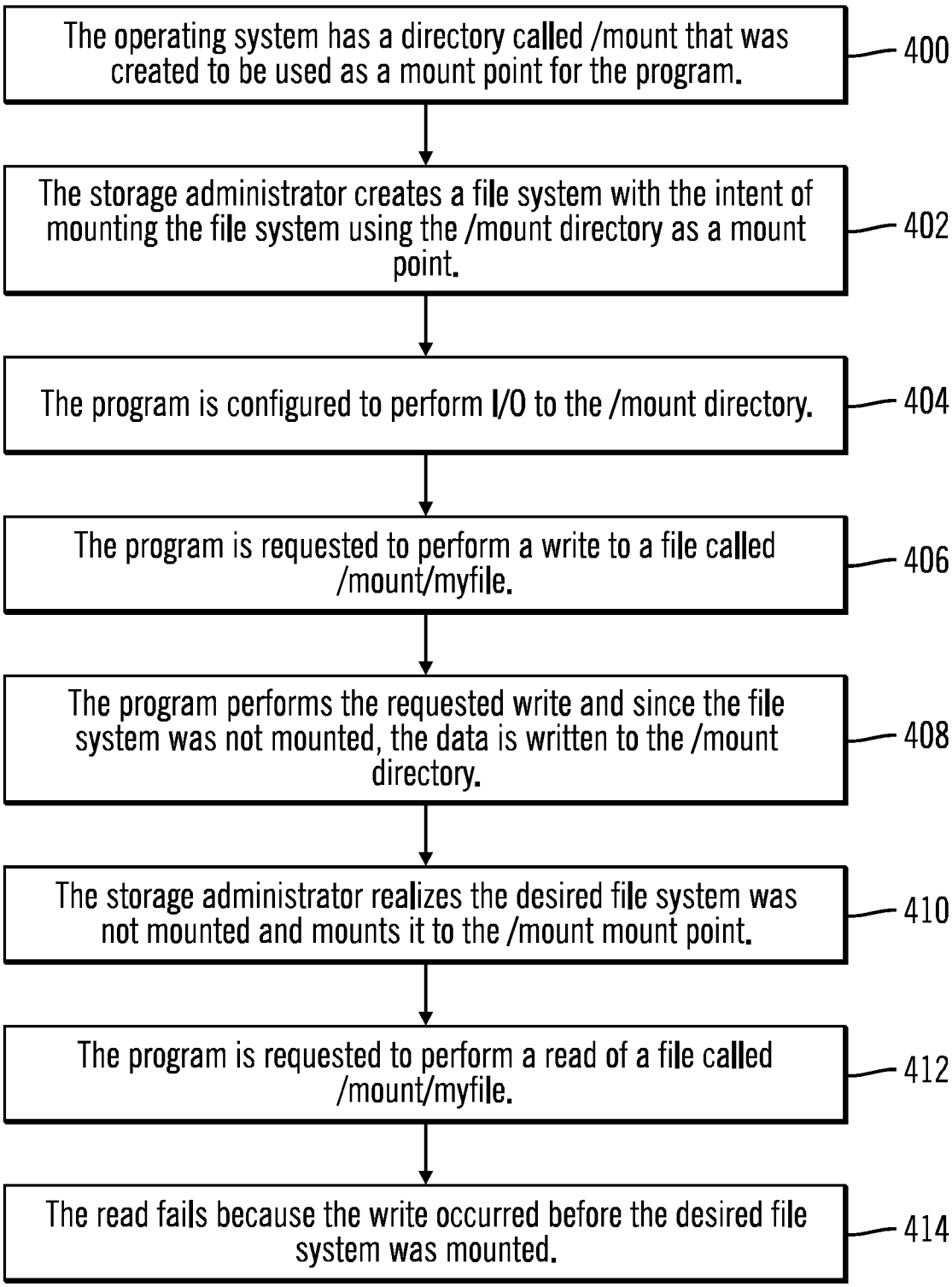
The operating system has a directory called /mount that was created to be used as a mount point for the program.
400
The storage administrator creates a file system with the intent of mounting the file system using the /mount directory as a mount point.
402
The program is configured to perform I/O to the /mount directory.
404
The program is requested to perform a write to a file called /mount/myfile.
406
The program performs the requested write and since the file system was not mounted, the data is written to the /mount directory.
408
The storage administrator realizes the desired file system was not mounted and mounts it to the /mount mount point.
410
The program is requested to perform a read of a file called /mount/myfile.
412
The read fails because the write occurred before the desired file system was mounted.
414

FIG. 4

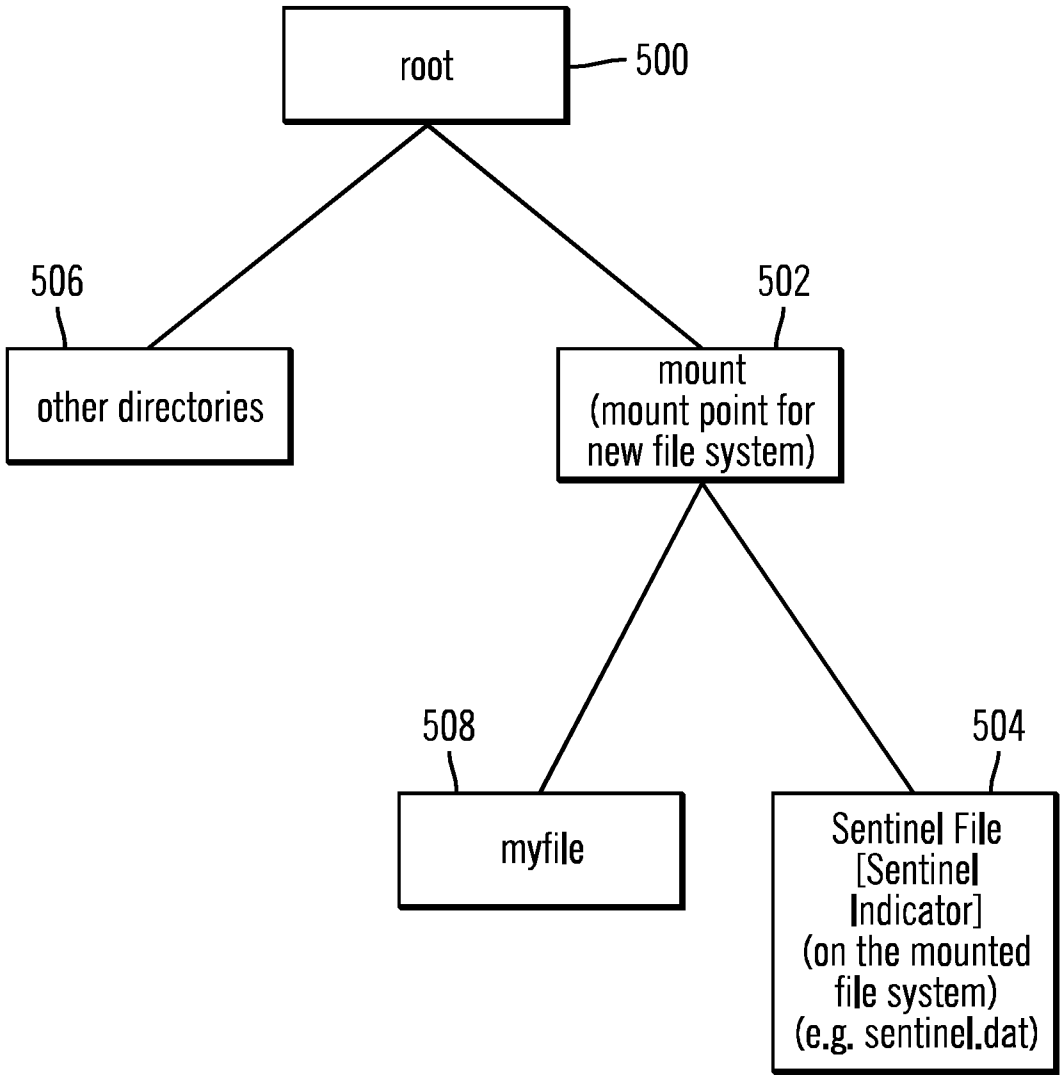
root
500
506
502
other directories
mount
(mount point for
new file system)
508
504
myfile
Sentinel File
[Sentinel
Indicator]
(on the mounted
file system)
(e.g. sentinel.dat)

FIG. 5

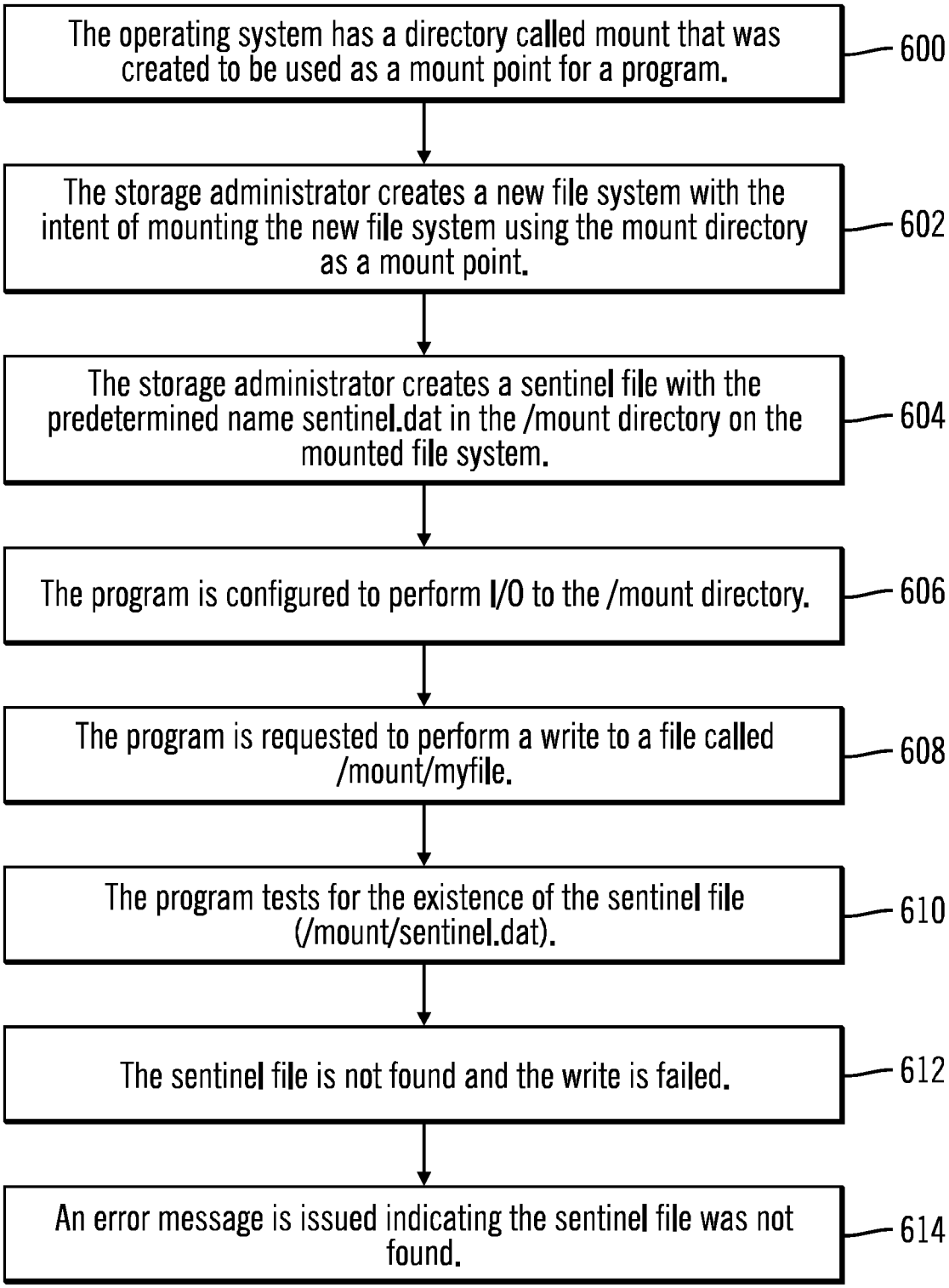
The operating system has a directory called mount that was created to be used as a mount point for a program.
600
The storage administrator creates a new file system with the intent of mounting the new file system using the mount directory as a mount point.
602
The storage administrator creates a sentinel file with the predetermined name sentinel.dat in the /mount directory on the mounted file system.
604
The program is configured to perform I/O to the /mount directory.
606
The program is requested to perform a write to a file called /mount/myfile.
608
The program tests for the existence of the sentinel file (/mount/sentinel.dat).
610
The sentinel file is not found and the write is failed.
612
An error message is issued indicating the sentinel file was not found.
614

FIG. 6

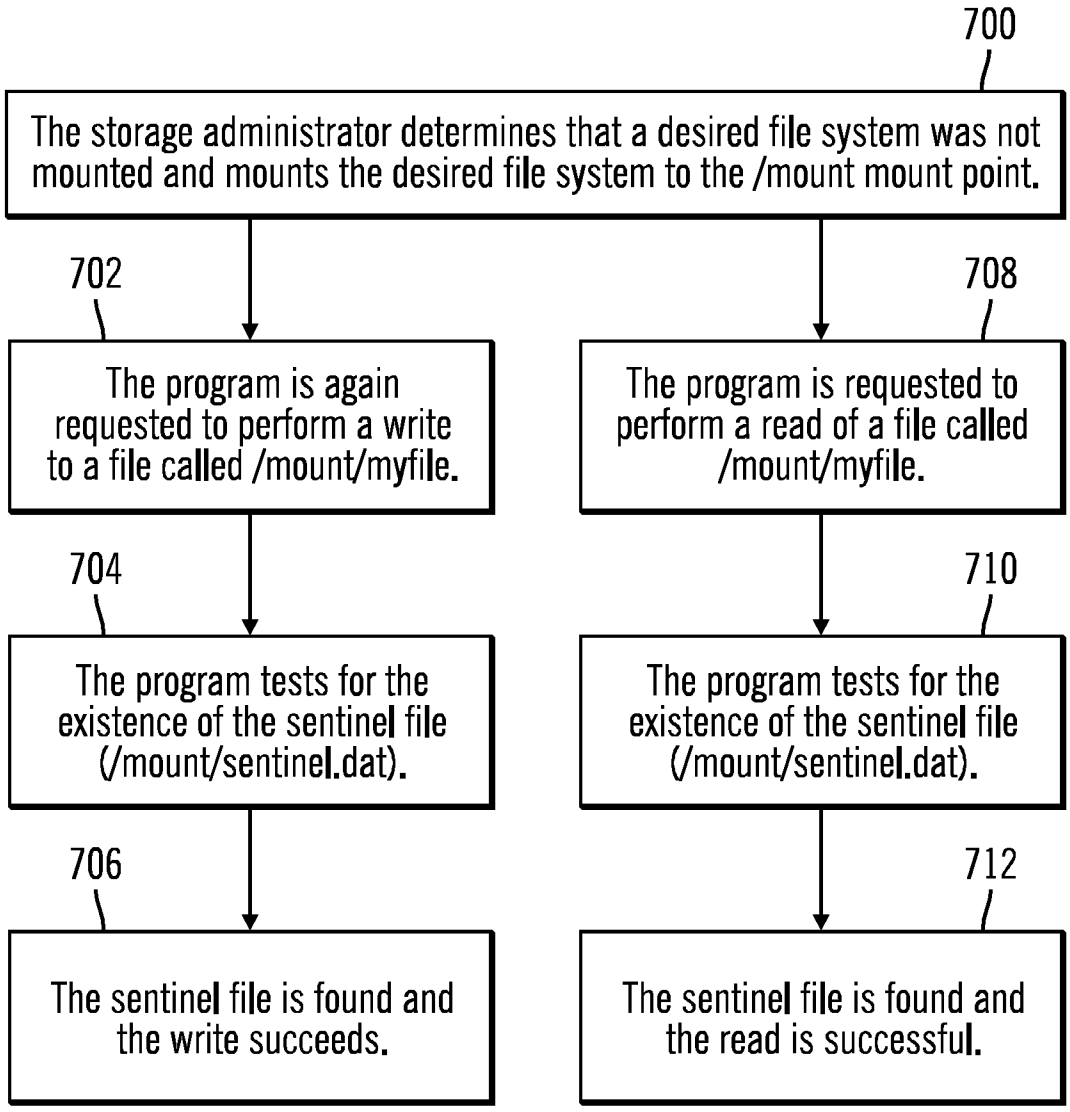
700
The storage administrator determines that a desired file system was not mounted and mounts the desired file system to the /mount mount point.
702
The program is again requested to perform a write to a file called /mount/myfile.
704
The program tests for the existence of the sentinel file (/mount/sentinel.dat).
706
The sentinel file is found and the write succeeds.
708
The program is requested to perform a read of a file called /mount/myfile.
710
The program tests for the existence of the sentinel file (/mount/sentinel.dat).
712
The sentinel file is found and the read is successful.

FIG. 7

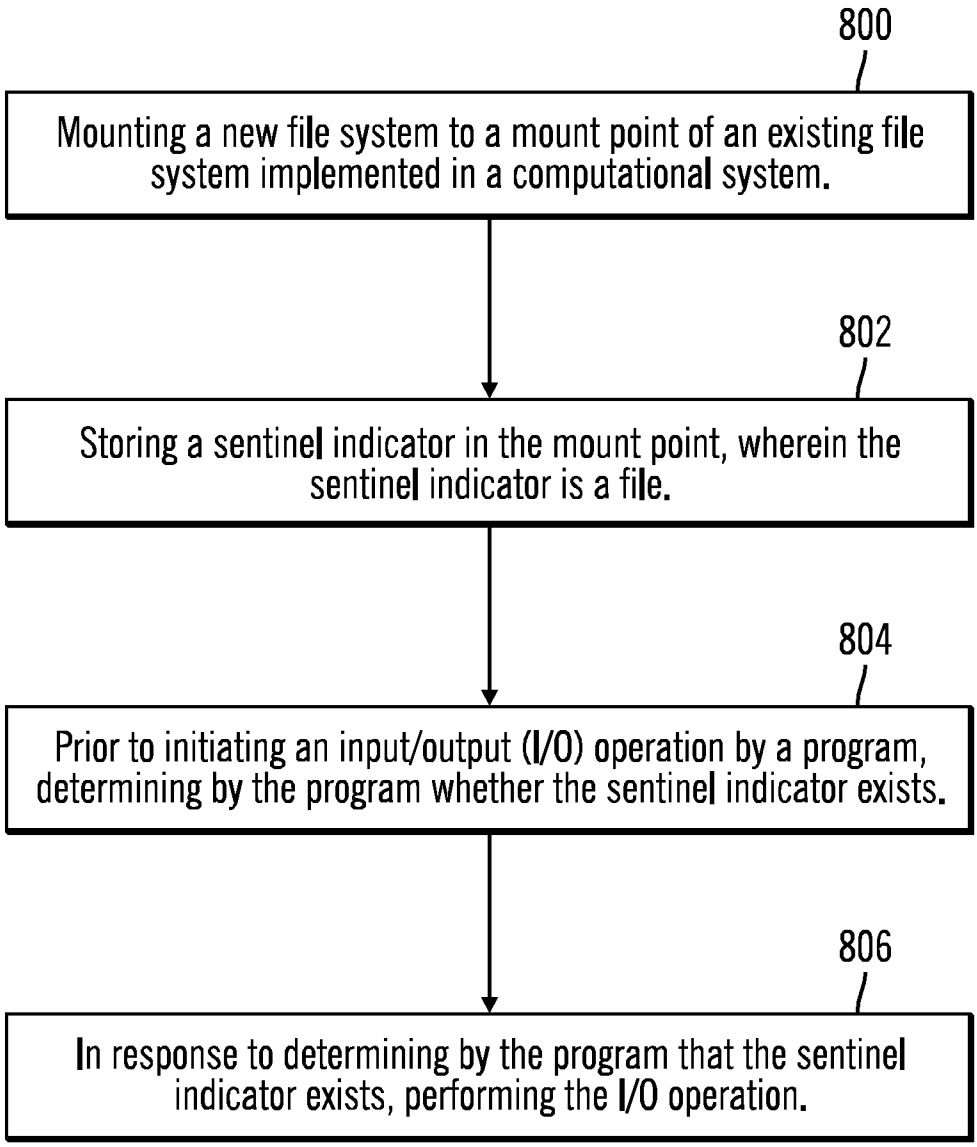
800
Mounting a new file system to a mount point of an existing file system implemented in a computational system.
802
Storing a sentinel indicator in the mount point, wherein the sentinel indicator is a file.
804
Prior to initiating an input/output (I/O) operation by a program, determining by the program whether the sentinel indicator exists.
806
In response to determining by the program that the sentinel indicator exists, performing the I/O operation.

FIG. 8

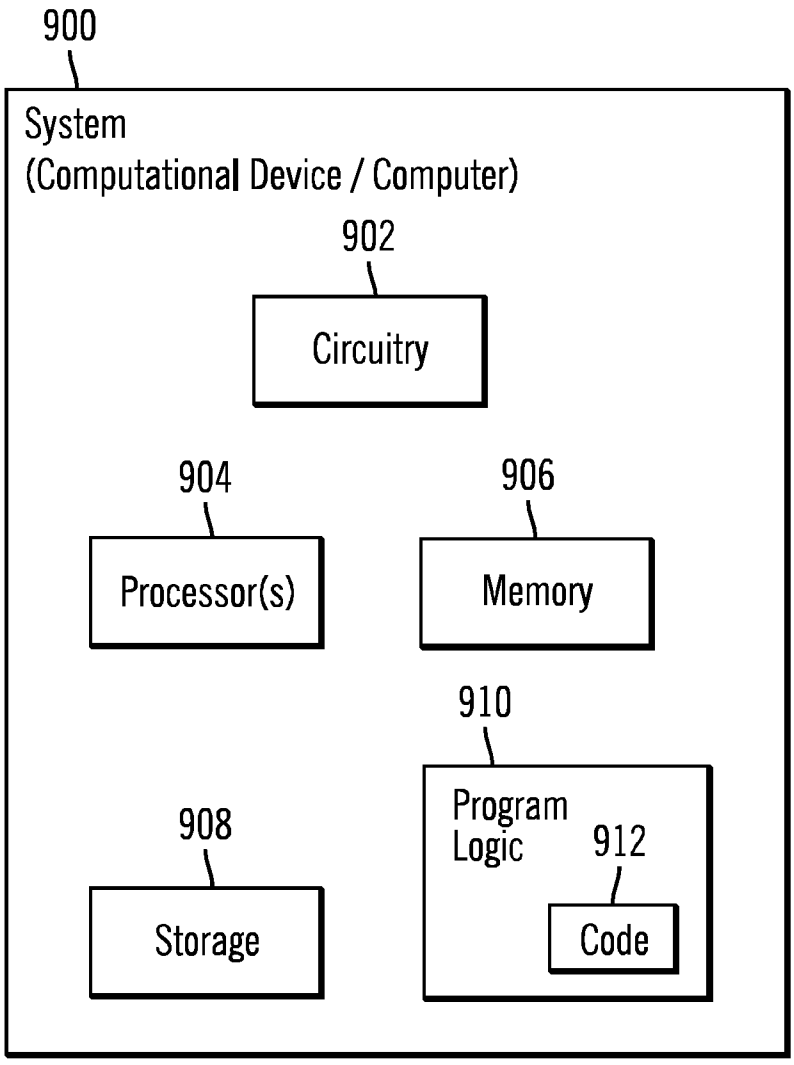
900
System
(Computational Device / Computer)
902
Circuitry
904
Processor(s)
906
Memory
910
Program
Logic
912
Code
908
Storage

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/057384

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	zman58 et al: "Problem with rsync, crontab and backing up to an external hard-drive"[Online] 19 July 2007 (2007-07-19), pages 1-4, XP002596798 UbuntuForums Retrieved from the Internet: URL:http://ubuntuforums.org/printthread.php?t=355030> [retrieved on 2010-08-17] page 4 ----- -/--	1-8

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 August 2010	20/10/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Kalejs, Eriks

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/057384

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Wallenstein D.: "Development Builds Layered on Top of a Stable System by Means of Unionfs"[Online] 1 March 2009 (2009-03-01), pages 1-5, XP002596799 ISSN: 1934-371X Linux Gazette Retrieved from the Internet: URL:http://linuxgazette.net/160/wallenstein.html> [retrieved on 2010-08-17] page 1 - page 2	1-8
X	Mahar R C et al: "Zenworks Content Respitory"[Online] 6 August 2008 (2008-08-06), pages 1-2, XP002596800 Novell Forums Retrieved from the Internet: URL:http://forums.novell.com/novell-product-support-forums/zenworks/configuration-management/zcm-imaging/338996-zenworks-content-respitory-print.html> [retrieved on 2010-08-17] page 1	1-8
X	Jones E et al: "Check if a filesystem is mounted"[Online] 28 March 2006 (2006-03-28), pages 1-8, XP002596801 niXdoc - nix Documentation Project Retrieved from the Internet: URL:http://nixforums.org/about140881-check-if-a-filesystem-is-mounted.html> [retrieved on 2010-08-17] page 3	1-8
X	Nathan K et al: "How To Move SuSE Install To A Larger Drive"[Online] 5 June 2005 (2005-06-05), pages 1-7, XP002596802 WebserverTalk Forums Retrieved from the Internet: URL:http://www.webservertalk.com/printthread.php?s=44ac19966aa3ed57e310afc0dee43f87&threadid=1083578&perpage=17> [retrieved on 2010-08-17] page 5	1-8
A	Korff Y et al: "Mastering FreeBSD and OpenBSD Security" 28 March 2005 (2005-03-28), O'Reilly , XP002596803 ISBN: 978-0-596-00626-6pages 37-37, page 37	1-8

-/--

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/057384

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2004/012379 A2 (DEEPFILE CORP [US]; BONE JEFF G [US]; FUNDERBURG BRETT A [US]; JIMENEZ) 5 February 2004 (2004-02-05) page 12 - page 36	1-8

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2010/057384

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2004012379 A2	05-02-2004	AU 2003265335 A1	16-02-2004