



US 20060089950A1

(19) **United States**

(12) **Patent Application Publication**
Tormasov et al.

(10) Pub. No.: **US 2006/0089950 A1**
(43) Pub. Date: **Apr. 27, 2006**

(54) **COMMON TEMPLATE FILE SYSTEM TREE**

Related U.S. Application Data

(60) Provisional application No. 60/277,292, filed on Mar. 20, 2001.

Publication Classification

(51) Int. Cl. **G06F 12/00** (2006.01)
(52) U.S. Cl. **707/200**

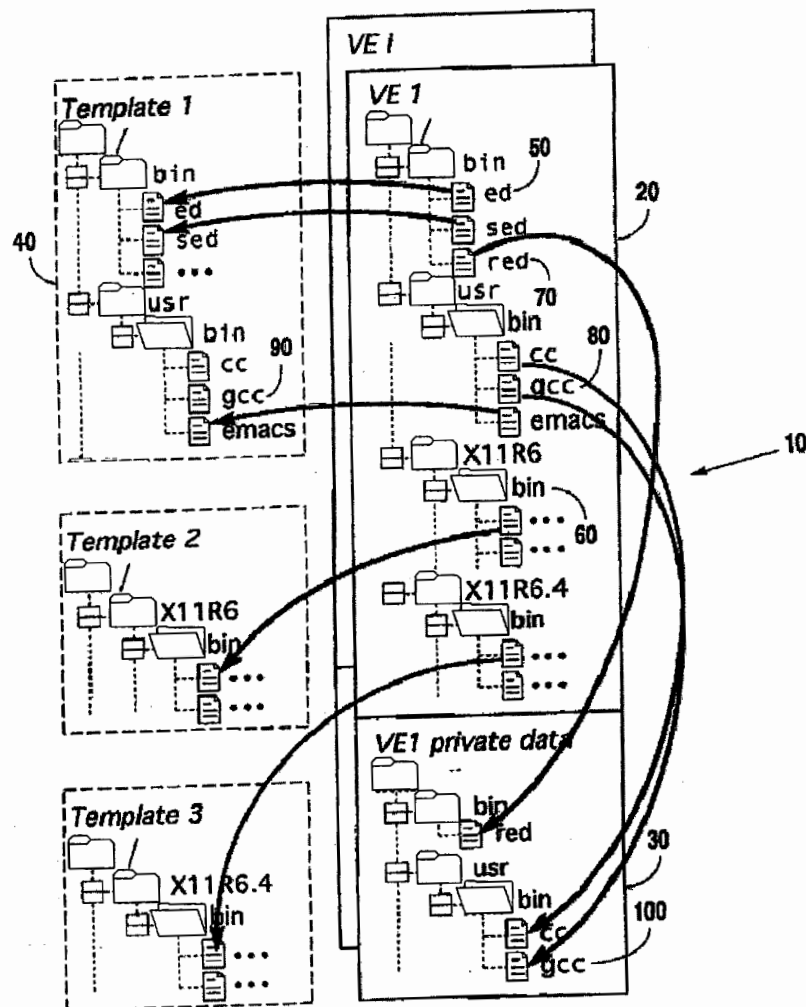
(76) Inventors: **Alexander Tormasov**, South San Francisco, CA (US); **Serguei Belousov**, Monte Sereno, CA (US); **Stanislav Protassov**, Singapore (SG); **Yuri Pudgorodsky**, South San Francisco, CA (US)

Correspondence Address:
BARDMESSER LAW GROUP, P.C.
910 17TH STREET, N.W.
SUITE 800
WASHINGTON, DC 20006 (US)

(21) Appl. No.: **09/908,252**
(22) Filed: **Jul. 18, 2001**

(57) **ABSTRACT**

A common template file system tree is utilized by isolated operating system processes groups for effective read-only common file set access via multiple file system paths. Files are opened from different views of the file system template for write-ability access and copied into a private modification area; this is also convenient for subsequent online changes and replication.



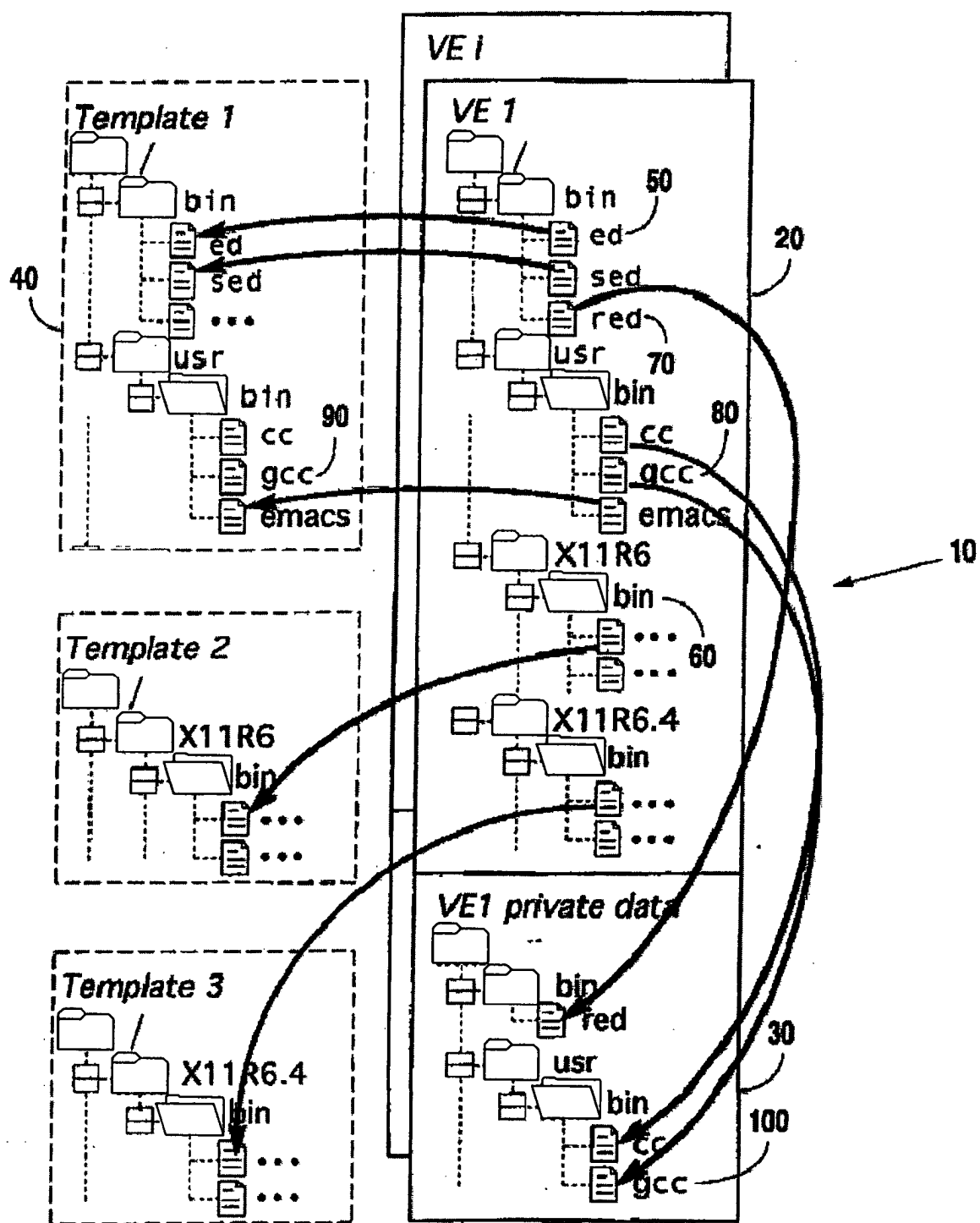


FIG. 1

COMMON TEMPLATE FILE SYSTEM TREE

REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application for Patent No. 60/277292 titled "A Method of Effectively Reusing a Common Template File System Tree in an Environment With Concurrent Access and a Separate Private Modification Area" filed on Mar. 20, 2001 for priority under 35 U.S.C. § 119(e), is related thereto, is commonly assigned therewith, and the subject matter thereof is incorporated herein by reference in its entirety.

FIELD

[0002] The system of the present invention relates to the implementation of a common template computer data storage file system to be used in an operating environment including concurrent access from multiple operating systems.

BACKGROUND

[0003] The problem of effective utilization of computer resources, such as a data storage file system in a multi-user environment, has existed from virtually the first use of computers. In such data storage file systems, the data storage files are traditionally separated into several classes: the personal files of the user, the data storage files shared by all the users, temporary files, etc. The classes of these data storage files usually differ in their disposition in the data storage file system and in their access mode. Depending on the settings of the operating system, only the owner of the data storage file or someone with explicitly declared access rights has write-ability access to these data storage files. Everyone has write-ability access to temporary data storage files. The majority of users have read-only access to the general system files. In the system area, only the auxiliary data storage files are deposited. Therefore, the auxiliary data storage file provides the working information for the operating system.

[0004] Ordinary computer users generally are not interested in modifications of the content of system data storage files. Therefore, most often, all of the system data storage files, including the configuration of files, are usually seen by all the users of the operating system, and they are typically seen only in the read-only mode. Nevertheless, it is often necessary to provide for the possibility of multiple users making corrections to the system area. For example, for the configuration of system utilities, it is desirable that the unique data storage file modifications made by each computer user do not influence the configurations of others. This means that each computer user should be provided with an independent file system tree with the data storage files suitable for writing. If there are many such computer users with independent file systems on the same computer, it would be essential to implement an effective method for each computer user to access such data storage files. Traditionally, this problem is solved in two ways: 1) copying corresponding data storage files for each computer user; 2) using the data storage file system to create and support hard links. With these solutions, multiple references would need to be created to the same data storage file using different names.

[0005] Copying corresponding data storage files for each computer user leads to the multiple duplication of data

storage files and is not an efficient use of data storage resources because of the typically large size of the system area. Another possible solution to this problem in the form of using the file system to create and support hard links, limits the user's ability to modify the data storage file system.

[0006] Since all of the users view the same system data storage file, its modification by one computer user will be immediately apparent to the other computer users who have access to the hard links of the same data storage file.

[0007] Thus, the need arises for an efficient solution that will allow many computer users to work on one physical computer. Each of the computer users has his/her own version, not only of their personal local data storage files, but also the system file area of the operating system.

[0008] Other prior art solutions have been discussed in the following references (See Bibliography). Partial solutions for this problem exist in many operating systems. For instance, in a UNIX type operating system, the primitive chroot is used to provide a certain level of security and separation for the computer users to allow them to have their own version of root data (Bach 1987, THE DESIGN AND IMPLEMENTATION OF THE 4.4BSD OPERATING SYSTEM). But this solution assumes that each computer user should have his/her own unique data storage file system tree and that the data storage file system tree cannot be effectively implemented.

[0009] Another partial solution is the separation of read and write operations provided by a so-called Union file system (Pendry 1995, THE DESIGN AND IMPLEMENTATION OF THE 4.4BSD OPERATING SYSTEM). The Union file system proposes a mount of one data storage file system tree on top of another tree. In such a case both trees become visible during the namespace lookup procedure. This allows the creation of separate write and read areas, but does not allow modification of data storage files which reside in the read-only area.

SUMMARY

[0010] The present invention offers a solution for the effective implementation of a multiple, partially shared tree of the data storage file system with the division of read-ability and write-ability streams in different areas of the namespace. As defined in OPERATING SYSTEMS: A DESIGN-ORIENTED APPROACH, namespace is a collection of unique names, where name is an arbitrary identifier, usually an integer or a character string (See CHARLES CROWLEY, OPERATING SYSTEMS: A DESIGN-ORIENTED APPROACH (Irwin, 1997) ISBN 0-256-15151-2). Usually the term "name" is applied to such objects as files, directories, devices, computers, etc. More information about typical distributed file system name space and related problems can be found in (Kumar 1991, Lebovitz 1992, The Distributed File System (DFS) for AIX/6000, Rosenberry 1992).

[0011] The search for a data storage file to be opened in the read-only mode is carried out in two stages. First, the search is done in the computer user's personal private area; and in case of failure, the search is done in the common read-only shared area. Writing to data storage files is carried out only in the computer user's personal private area. Thus, from the

point of view of the computer user, the computer user has only one tree of the data storage file system that can be modified in every place. Nevertheless, only the differences between the data storage file system common to all of the users of the computer and the modified user copy is actually preserved. The common part of the data storage file is preserved on the computer only in one copy and does not need duplication. Operation parsing is carried out on its type. When a data storage file is opened in read-only mode, then if the private data storage file is present, it is opened in the private modification area. If not, the single copy of one data storage file from the shared data storage file system is opened.

[0012] When the data storage file is opened for write-ability and the private data storage file is present, the file is opened in the private area of the computer user. If the private data storage file is not present, the data storage file from the shared area is at first copied into the private area and only then it is opened. Such organization of the work of the data storage file system makes the implementation much easier for journaling because all the changes for each computer user are localized and are kept in his/her private area.

[0013] The data storage file being observed from different views of the same common template data storage file system is treated by the operation system as the only file. This promotes the efficient utilization of computer memory. For instance, if this data storage file contains executable code, the operation system stores in memory only the shared copy for all instances of processes using this data storage file from different views.

BRIEF DESCRIPTION OF THE DRAWING

[0014] A better understanding of the data storage system and method of the present invention may be had by reference to the drawing figure wherein:

[0015] **FIG. 1** is a schematic illustration of the relationship between a common root template of the data storage file system and the private modification area.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0016] The disclosed invention describes a way to effectively implement multiple, partially shared trees of a data storage file system by separating the read-ability and write-ability streams into different areas of the namespace. Namespace is a collection of unique names, where name is an arbitrary identifier, usually an integer or a character string. Usually the term "name" is applied to such objects as files, directories, devices, computers, etc.

[0017] The idea behind the instant invention is that the disclosed system 10 separates the modified part of the data storage file system from the non-modified part of the data storage file system in such a way that it creates two complete data storage file system trees of the required area of the file system imposed upon each other. Specifically, as VE1 10 uses an integral data storage file system visible by the computer user; some visible data storage files exist only in some templates, and some visible files exist only in the private modification areas.

[0018] In the initial moment of time, the first data storage file system tree for read-only material contains the complete

set of data storage files. The data storage file system tree for write-ability is empty, but it has the same directory structure.

[0019] **FIG. 1** is a schematic illustration of the relationship between a common root template of the data storage file system and the private modification area. Data storage files could appear in read-only templates (see Template I 40) and in private modification areas (see VE1 private data 30) which are visible from the same data file storage tree (see VE1 20). Data storage files visible from VE1 20 could be placed in different read-only templates, as for example, data storage files with names "Ibin/ed"50 and "X11R6/bin"60 here, or in a private area as for example, the data storage file "/bin/red"70. Even if a data storage file such as "/usr/bin/gcc"80 is available in both template, e.g., Template I 40, (see /usr/bin/gcc 90) and private modification area, e.g., VE1 private data 30, (see /usr/bin/gcc 100) the user will see only the file 100 in the private modification area 30.

[0020] As shown in **FIG. 1**, the first data storage file system tree 20 is never changed and is kept in one copy so that it can be seen by all of the computer users. The second data storage file system tree 30 is kept only in the personal private area of the computer user, and it is unique. If a search is made for some data storage file in such a structure, the data storage file is sought out twice: First, in the private tree 30 for write-ability; then second, in the common read-only tree 20, in the case of failure.

[0021] Opening of data storage files in the read-only mode is carried out in two stages. First, an attempt is made to open the data storage file in the personal private modification area of the computer user 30. Then in case of failure, the data storage file is opened in the common read-only shared area 20.

[0022] Writing data into data storage files is carried out only in the personal area of the computer user. At first, it is defined if such a data storage file exists in the personal area. And then it is opened for write-ability in the case of a successful search. If the data storage file does not exist, then the data storage file is copied into the private area 30 of the computer user from the shared data storage file system tree 20 and the obtained copy of the data storage file is opened to allow the writing of data to the data storage file.

[0023] Accordingly, from the point of view of the computer user, the computer user has only one data storage file system tree within the data storage file system that can be modified in every place. Nevertheless, only the difference between the data storage file system common to all of the users of the computer and the modified user copy is actually stored. The shared part of the data storage file system is preserved in the computer only in one copy, and it does not need duplication.

[0024] In case the data storage file is removed, the data storage file is just marked as removed in the personal area of the computer user and the search procedure, when such notation exists, is finished with the reply to the computer user that the file does not exist.

[0025] Such organization of work of the data storage file system sufficiently simplifies the implementation of journaling because all the changes for every computer user are localized and kept in his/her private area. The consequences of writing into a private area is conditioned by algorithms that: i) service the file system, and ii) could be easily tracked

by special processes of the operating system that serve the procedures of journaling. Accordingly, this method for reusing a common template file system may form the separate private modification area as a transaction and could store the separate private modification area in a manner that allows the organization of the standard journaling.

[0026] Those of ordinary skill in the art will understand that numerous embodiments have been enabled by the above disclosed invention. Such other embodiments shall be included within the scope and meaning of the appended claims.

BIBLIOGRAPHY

- [0027] Bach, Marice J. and Maurice Bach. 1987. *Design of the Unix Operating System* (Prentice Hall 1 edition Feb. 27, 1987); ISBN: 0132017997.
- [0028] Crowley, Charles. 1997. *Operating Systems: A Design-Oriented Approach* (Irwin, 1997) ISBN 0-256-15151-2.
- [0029] Kumar, R. Fall 1991. "OSF's Distributed Computing Environment", Aixpert, IBM Corporation, at 22-29.
- [0030] Lebovitz, G. "An Overview of the OSF DCE Distributed File System", Aixpert, IBM, at 55-64 (February 1992).
- [0031] Pendry, J. S. and McKusick, M. K. 1995. *Union mounts in 4.4BSD-Lite*, CONFERENCE PROCEEDINGS OF THE USENIX 1995 TECHNICAL CONFERENCE ON UNIX AND ADVANCED COMPUTING SYSTEMS (NEW ORLEANS), at 25-33. USENIX ASSOCIATION 16-20 (January 1995).
- [0032] Rosenberry, W., et al, September 1992. "Understanding, DCE", 6-100 (O'Reilly & Associates, Inc. Publishers).
- [0033] *The Design and Implementation of the 4.4BSD Operating System, (Unix and Open Systems Series)* (Marshall Kirk McKusick, et al. eds., Addison-Wesley Pub Co.) ISBN: 0201549794.
- [0034] "The Distributed File System (DFS) for AIX/6000", IBM, Doc. No. GG24-4255-00, at 1-15 (May 1994).

1-5. (canceled)

6. A computer data storage file system comprising:

a namespace;

a common template file system tree concurrently accessed by isolated groups of operating system processes using a view to the common template file system tree;

a first set of names from the namespace for direct access of the common template file system tree;

a private modification area accessed by the isolated groups of operating system processes for supporting the view; and

a second set of names from the namespace used for direct access of files in the private modification area.

7. The system of claim 6, further comprising multiple views inside the common template file system tree;

wherein the namespace includes a third set of names associated with the isolated groups of operating system processes for use in establishing the multiple views,

wherein read-only access to a file associated with the isolated groups of operating system processes using the third set of names utilizes the common template file system tree, and

wherein, if the same files are opened from different views of the common template file system tree using a name from the third set of names, these files are cached only once in an operating system file buffer.

8. The system of claim 7, wherein a standard file system interface is utilized to enable file access that is transparent for the isolated groups of operating system processes.

9. The system of claim 7, wherein if a file already exists in the private modification area, the read access is performed on that file.

10. The system of claim 7, further comprising a transaction log, wherein contents of the transaction log corresponds to contents of the private modification area.

11. The system of claim 10, wherein the contents of the transaction log is kept up to date and the transaction log is used for online replication.

12. The system of claim 7, wherein a standard file system is utilized to enable file access that is transparent for any operating system process.

13. The system of claim 6, further comprising multiple views inside a file system name space to the common template file system tree;

wherein a view inside a file system namespace to the common template file system tree includes a third set of names of the namespace associated with an isolated group of operating system processes for use in establishing the multiple views;

wherein the common template file system tree is utilized by the isolated groups of the operating system processes using the third set of names; and

wherein all files opened for write access are copied from the common template file system tree into the private modification area for performing the write access on the copy of the file.

14. The system of claim 13, wherein a standard file system is utilized to enable file access that is transparent for the isolated groups of operating system processes.

15. The system of claim 13, wherein:

if a file already exists in the private modification area, the write access is performed on that file, and

if the file does not exist in the private modification area, then the file is copied to the private modification area prior to the write access.

16. The system of claim 13, wherein, if a file has been removed from the private modification area, a user is informed that the file does not exist.

17. A method of managing a computer file system comprising:

establishing a namespace;

establishing a first set of names from the namespace for direct access of the common template file system tree;

establishing a common template file system tree for concurrent access by isolated groups of operating system processes using a view to the common template file system tree;

establishing a second set of names from the namespace for direct access of files in a private modification area; and

accessing the private modification area by the isolated groups of operating system processes.

18. The method of claim 17, further comprising establishing multiple views inside the common template file system tree,

wherein the namespace includes a third set of names associated with the isolated groups of operating system processes for use in establishing the multiple views,

wherein read-only access to a file associated with the isolated groups of operating system processes using the third set of names utilizes the common template file system tree, and

wherein, if the same files are opened from different views of the common template file system tree using a name from the third set of names, these files are cached only once in an operating system file buffer.

19. The method as defined in claim 18, wherein if a file already exists in the private modification area, the read access is performed on that file.

20. The method of claim 17, wherein a standard file system is utilized to enable file access that is transparent for the isolated groups of operating system processes.

21. The method of claim 17, further comprising generating a transaction log, wherein contents of the transaction corresponds to contents of the private modification area.

22. The method as defined in claim 21, further comprising keeping the the transaction log up to date for online replication.

23. The method of claim 17, wherein a standard file system is utilized to enable file access that is transparent for any operating system process.

24. The method of claim 17, further comprising performing a write access on a file,

wherein the namespace includes a third set of names associated with the operating system processes, and

wherein all files opened for write access are copied from the common template file system tree into the private modification area prior to performing the write access.

25. The method of claim 24, wherein:

if a file already exists in the private modification area, performing the write access on that file, and

if the file does not exist in the private modification area, copying the file to the private modification area prior to performing the write access on that file.

26. The method of claim 24, wherein, if a file has been removed from the private modification area, informing a user that the file does not exist.

27. A computer program product for managing a computer file system, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling a processor, the computer program logic comprising:

computer program code means for establishing a namespace;

computer program code means for establishing a first set of names from the namespace for direct access of the common template file system tree;

computer program code means for establishing a common template file system tree for concurrent access by isolated groups of operating system processes using a view to the common template file system tree;

computer program code means for establishing a second set of names from the namespace for direct access of files in a private modification area; and

computer program code means for accessing the private modification area by the isolated groups of operating system processes.

28. The computer program product of claim 27, further comprising computer program code means for establishing multiple views inside the common template file system tree;

wherein the namespace includes a third set of names associated with the isolated groups of operating system processes for use in establishing the multiple views;

wherein read-only access to a file associated with the isolated groups of operating system processes using the third set of names utilizes the common template file system tree; and

wherein, if the same files are opened from different views of the common template file system tree using a name from the third set of names, these files are cached only once in an operating system file buffer.

29. The computer program product as defined in claim 28, wherein if a file already exists in the private modification area, the read access is performed on that file.

30. The computer program product of claim 27, wherein a standard file system is utilized to enable file access that is transparent for the operating system.

31. The computer program product of claim 27, further comprising computer program code means for generating a transaction log, wherein contents of the transaction corresponds to contents of the private modification area.

32. The computer program product as defined in claim 31, further comprising computer program code means for keeping the transaction log up to date for online replication.

33. The computer program product of claim 27, wherein a standard file system is utilized to enable file access that is transparent for the operating system processes.

34. The computer program product of claim 27, further comprising computer program code means for performing a write access on a file,

wherein the namespace includes a third set of names associated with the operating system processes,

wherein the common template file system tree is utilized by isolated groups of operating system processes, and

wherein all files opened for write access are copied from the common template file system tree into the private modification area prior to performing the write access.

35. The computer program product of claim 34, further comprising:

if a file already exists in the private modification area, computer program code means for performing the write access on that file, and

if the file does not exist in the private modification area, computer program code means for copying the file to the private modification area prior to performing the write access on that file.

36. The computer program product of claim 34, wherein, if a file has been removed from the private modification area, informing a user that the file does not exist.

37. The computer program product of claim 27, wherein a standard file system is utilized to enable file access that is transparent for any operating system process.

* * * * *