EXHIBIT 9

* *

What is claimed is:

1. A method for implementing a temporary file system in a computer, the method comprising:

creating a device driver which manages a virtual device;

including in the device driver a procedure which represents data stored in the virtual device using anonymous storage; and

configuring a file system to represent files associated with the virtual device in anonymous storage.

2. The method of Claim 1 wherein the step of configuring comprises:

mounting the file system on the virtual device in a delayed-write mode.

3. A method for accessing subject data in a computer system, the method comprising:

determining that the subject data represents at least a portion of a temporary file;

associating anonymous storage with a block of data which includes the subject data;

[illegible]

determining a location within the anonymous storage which corresponds to the subject data; and

effecting access of data at the location within the anonymous memory.

4. The method of Claim 3 wherein the step of associating comprises:

creating a database of one or more records, each of which associates an associated location within the anonymous storage with an associated offset into an associated block of data.

5. The method of Claim 4 wherein the step of determining a location comprises:
locating within the database a selected one of the one or more records, wherein the selected record associates the offset with the location.

6. The method of Claim 5 wherein the database is a skip list of the one or more records.

7. The method of Claim 3 wherein the step of effecting access comprises:
building a user I/O structure which specifies the location within the anonymous storage.

8. The method of Claim 7 wherein the step of building comprises:
building an iovec structure which specifies the location within the anonymous storage.

9. The method of Claim 3 wherein the step of effecting access comprises:
invoking a VOP_READ() procedure of a swap file system of the computer system.

10. The method of Claim 3 wherein the step of effecting access comprises:
invoking a VOP_WRITE() procedure of a swap file system of the computer system.

11. The method of Claim 3 wherein the step of effecting access comprises:
invoking a VOP_PAGEIO() procedure of a swap file system of the computer system.

12. A computer program product comprising:
a computer usable medium having computable readable code embodied therein for

implementing a temporary file system in a computer, the computer readable code comprising:

a device driver which is configured to manage a virtual device, the device driver comprising:

a procedure module which is configured to represent data stored in the virtual device using anonymous storage.

13. The computer program product of Claim 12 wherein the device driver further comprises:

an attach procedure which is configured to facilitate installation of the device driver such that files associated with the virtual device are represented in anonymous storage.

14. The computer program product of Claim 12 wherein a file system is mounted on the virtual device in a delayed-write mode.

15. A computer program product comprising:

a computer usable medium having computable readable code embodied therein for accessing subject data in a computer system, the computer readable code comprising:

a virtual device initialization module which is configured to direct a file system of the computer system to determine whether the subject data represents at least a portion of a temporary file;

a mapping database which associates anonymous storage with a block of data which includes the subject data;

a mapping module which is operatively coupled to the virtual device initialization module and to the mapping database and which determines from the mapping database a location within the anonymous storage which corresponds to the subject data; and a data access module which is operatively coupled to the mapping module and which is configured to access data at the location within the anonymous memory.

16. The computer program product of Claim 15 wherein the mapping database comprises:

one or more records, each of which associates an associated location within the anonymous storage with an associated offset into an associated block of data.

17. The computer program product of Claim 16 wherein the data access module is further configured to locate within the mapping database a selected one of the one or more records, wherein the selected record associates the offset with the location.

18. The computer program product of Claim 16 wherein the mapping database is a skip list of the one or more records.

19. The computer program product of Claim 16 wherein the data access module comprises:

a user I/O structure constructor which is configured to build a user I/O structure which in turn specifies the location within the anonymous storage.

20. The computer program product of Claim 19 wherein the data access module further comprises:

an iovec structure constructor which is operatively coupled to the user I/O structure constructor and which is configured to build an iovec structure which specifies the location within the anonymous storage.

21. The computer program product of Claim 15 wherein the data access module

comprises:

a VOP_READ caller which is configured to invoke a VOP_READ() procedure of a swap file system of the computer system.

22. The computer program product of Claim 15 wherein the data access module comprises:

a VOP_WRITE caller which is configured to invoke a VOP_WRITE() procedure of a swap file system of the computer system.

23. The computer program product of Claim 15 wherein the data access module comprises:

a VOP_PAGEIO caller which is configured to invoke a VOP_PAGEIO() procedure of a swap file system of the computer system.

24. An apparatus for implementing a temporary file system in a computer, the apparatus comprising:

a device driver which is configured to manage a virtual device, the device driver comprising:

a procedure module which is configured to represent data stored in the virtual device using anonymous storage.

25. The apparatus of Claim 24 wherein the device driver further comprises:

an attach procedure which is configured to facilitate installation of the device driver such that files associated with the virtual device are represented in anonymous storage.

26. The apparatus of Claim 24 wherein a file system is mounted on the virtual device in a delayed-write mode.

27. An apparatus for accessing subject data in a computer system, the apparatus comprising:

a virtual device initialization module which is configured to direct a file system of the computer system to determine whether the subject data represents at least a portion of a temporary file;

a mapping database which associates anonymous storage with a block of data which includes the subject data;

a mapping module which is operatively coupled to the virtual device initialization module and to the mapping database and which determines from the mapping database a location within the anonymous storage which corresponds to the subject data; and

a data access module which is operatively coupled to the mapping module and which is configured to access data at the location within the anonymous memory.

28. The apparatus of Claim 27 wherein the mapping database comprises:

one or more records, each of which associates an associated location within the anonymous storage with an associated offset into an associated block of data.

29. The apparatus of Claim 28 wherein the data access module is further configured to locate within the mapping database a selected one of the one or more records, wherein the selected record associates the offset with the location.

30. The apparatus of Claim 28 wherein the mapping database is a skip list of the one or more records.

31. The apparatus of Claim 28 wherein the data access module comprises:

a user I/O structure constructor which is configured to build a user I/O structure which in turn specifies the location within the anonymous storage.

32. The apparatus of Claim 31 wherein the data access module further comprises:

an iovec structure constructor which is operatively coupled to the user I/O structure constructor and which is configured to build an iovec structure which specifies the location within the anonymous storage.

33. The apparatus of Claim 27 wherein the data access module comprises:

a VOP_READ caller which is configured to invoke a VOP_READ() procedure of a swap file system of the computer system.

34. The apparatus of Claim 27 wherein the data access module comprises:

a VOP_WRITE caller which is configured to invoke a VOP_WRITE() procedure of a swap file system of the computer system.

35. The apparatus of Claim 27 wherein the data access module comprises:

a VOP_PAGEIO caller which is configured to invoke a VOP_PAGEIO() procedure of a swap file system of the computer system.

36. A computer system comprising:

a computer processor;

a memory operatively coupled to the computer processor;

a file processing system which is stored in the memory and which includes one or more computer instructions which are executed within the computer processor to implement a temporary file system, the file processing system including:

a device driver which is configured to manage a virtual device, the device driver comprising:

a procedure module which is configured to represent data stored in the virtual device using anonymous storage.

a

37. The computer system of Claim 36 wherein the device driver further comprises:
an attach procedure which is configured to facilitate installation of the device driver such that files associated with the virtual device are represented in anonymous storage.

38. The computer system of Claim 36 wherein a file system is mounted on the virtual device in a delayed-write mode.

39. A computer system comprising:
a computer processor;
a memory operatively coupled to the computer processor;
a file processing system which is stored in the memory and which includes one or more computer instructions which are executed within the computer processor to access subject data in the computer system, the file processing system including:
a virtual device initialization module which is configured to direct a file system of the computer system to determine whether the subject data represents at least a portion of a temporary file;
a mapping database which associates anonymous storage with a block of data which includes the subject data;
a mapping module which is operatively coupled to the virtual device initialization module and to the mapping database and which determines from the mapping database a location within the anonymous storage which corresponds to the subject data; and
a data access module which is operatively coupled to the mapping module and which is configured to access data at the location within the anonymous memory.

40. The computer system of Claim 39 wherein the mapping database comprises:

one or more records, each of which associates an associated location within the anonymous storage with an associated offset into an associated block of data.

41. The computer system of Claim 40 wherein the data access module is further configured to locate within the mapping database a selected one of the one or more records, wherein the selected record associates the offset with the location.

42. The computer system of Claim 40 wherein the mapping database is a skip list of the one or more records.

43. The computer system of Claim 40 wherein the data access module comprises:

a user I/O structure constructor which is configured to build a user I/O structure which in turn specifies the location within the anonymous storage.

44. The computer system of Claim 43 wherein the data access module further comprises:

an iovec structure constructor which is operatively coupled to the user I/O structure constructor and which is configured to build an iovec structure which specifies the location within the anonymous storage.

45. The computer system of Claim 39 wherein the data access module comprises:

a VOP_READ caller which is configured to invoke a VOP_READ() procedure of a swap file system of the computer system.

46. The computer system of Claim 39 wherein the data access module comprises:

a VOP_WRITE caller which is configured to invoke a VOP_WRITE() procedure of a swap file system of the computer system.

47. The computer system of Claim 39 wherein the data access module comprises:

a VOP_PAGEIO caller which is configured to invoke a VOP_PAGEIO() procedure of a swap file system of the computer system.

48. A distribution system for distributing code (i) which is stored on a computer-readable medium, (ii) which is executable by a computer, and (iii) which includes at least one module, each of which in turn is configured to carry out at least one function to be executed by the computer, the at least one module implementing a temporary file system, the distribution system comprising:

a device driver which is configured to manage a virtual device, the device driver comprising:

a procedure module which is configured to represent data stored in the virtual device using anonymous storage.

49. The distribution system of Claim 48 wherein the device driver further comprises:

an attach procedure which is configured to facilitate installation of the device driver such that files associated with the virtual device are represented in anonymous storage.

50. The distribution system of Claim 48 wherein a file system is mounted on the virtual device in a delayed-write mode.

51. A distribution system for distributing code (i) which is stored on a computer-readable medium, (ii) which is executable by a computer, and (iii) which includes at least one module, each of which in turn is configured to carry out at least one function to be executed by the computer, the at least one module configured for accessing subject data in the computer, the distribution system comprising:

a virtual device initialization module which is configured to direct a file system of the computer system to determine whether the subject data represents at least a portion of a temporary file;

a mapping database which associates anonymous storage with a block of data which includes the subject data;

a mapping module which is operatively coupled to the virtual device initialization module and to the mapping database and which determines from the mapping database a location within the anonymous storage which corresponds to the subject data; and

a data access module which is operatively coupled to the mapping module and which is configured to access data at the location within the anonymous memory.

52. The distribution system of Claim 51 wherein the mapping database comprises:

one or more records, each of which associates an associated location within the anonymous storage with an associated offset into an associated block of data.

53. The distribution system of Claim 52 wherein the data access module is further configured to locate within the mapping database a selected one of the one or more records, wherein the selected record associates the offset with the location.

54. The distribution system of Claim 52 wherein the mapping database is a skip list of the one or more records.

55. The distribution system of Claim 52 wherein the data access module comprises:

a user I/O structure constructor which is configured to build a user I/O structure which in turn specifies the location within the anonymous storage.

56. The distribution system of Claim 55 wherein the data access module further comprises:

an iovec structure constructor which is operatively coupled to the user I/O structure constructor and which is configured to build an iovec structure which specifies the location within the anonymous storage.

57. The distribution system of Claim 51 wherein the data access module comprises:
a VOP_READ caller which is configured to invoke a VOP_READ() procedure of a swap file system of the computer system.

58. The distribution system of Claim 51 wherein the data access module comprises:
a VOP_WRITE caller which is configured to invoke a VOP_WRITE() procedure of a swap file system of the computer system.

59. The distribution system of Claim 51 wherein the data access module comprises:
a VOP_PAGEIO caller which is configured to invoke a VOP_PAGEIO() procedure of a swap file system of the computer system.

add
A1

OIPE JC57
JUL 20 1998
PATENT & TRADEMARK OFFICE

PATENT

12/a
SK
7-31-98

In the United States Patent and Trademark Office

Applicant: FOX et al.
Application No.: 08/675,360
Filed: July 1, 1996
Title: FAST MEMORY-BASED TEMPORARY COMPUTER FILE SYSTEM

Applicant's Ref: SUN1P164/P1251
Examiner: Portaka, G.
Group Art Unit: 2751
Date: July 15, 1998

RECEIVED
98 JUL 22 AM 11:17
GROUP 2700

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on July 15, 1998.

Signed: Sharon D. Coleman
Sharon D. Coleman

AMENDMENT

Assistant Commissioner for Patents
Box Non-Fee Amendment
Washington, D.C. 20231

Sir:

In response to the Office Action dated May 5, 1998, for which the shortened statutory period for response is set to expire on August 5, 1998, please amend this application as follows.

IN THE SPECIFICATION

Please amend the title of the application to read "Temporary Computer File System Implementing Using Anonymous Storage Allocated For Virtual Memory".

A

IN THE CLAIMS:

Please cancel claims 1-59 and insert the following new claims 60 – 99 therefore.

Sub B1

--60. In a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, said operating system including a file system (FS) for cataloguing files, said files including a data portion and a metadata portion, a method for implementing a temporary file within said file system, the method comprising:

(a) providing a file manager within the file system for managing persistent data files and non-persistent data files within said operating system;

(b) mounting and attaching within the operating system a device driver for representing a virtual memory device used for implementing at least one non-persistent data file within said computer system, said virtual memory device including an anonymous storage system;

(c) using said file manager to invoke procedures within said device driver for accessing desired information within said virtual memory device relating to said at least one non-persistent data file; and

a1

(d) using said device driver to effect access of said desired information within said anonymous storage system.

61. The method of claim 60 wherein said computer system includes a second device driver for managing a secondary storage device, said method further including the step of configuring each driver device to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.

3 ~~62~~. The method of claim 1 ~~60~~ including the step of storing the data portion and metadata portion of a non-persistent file in said anonymous storage system.

4 ~~63~~. The method of claim 1 ~~60~~ wherein said operating system is a UNIX-based operating system.

Sub B2 64. The method of claim 60 wherein said mounting step (b) further includes the step of mounting said device driver using a delayed-write mode.

65. The method of claim 60 wherein said virtual storage device further includes a swap file system for allocating anonymous storage space in said anonymous storage system, said method

~~further including the step of using said device driver to manage information stored in said anonymous storage system by invoking procedures defined within said swap file system.~~

~~66. In a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, said operating system including a storage media file system (FS) for cataloging files, a method for implementing a non-persistent data file within said file system, the method comprising:~~

~~(a) providing a file manager within the file system for managing persistent data files and non-persistent data files;~~

~~(b) mounting and attaching within the operating system a device driver representing a virtual memory device used for implementing non-persistent data files within said computer system, said virtual memory device including an anonymous storage system;~~

~~(c) creating a non-persistent data file by invocation of a procedure of the storage media file system, while indicating that said non-persistent data file is to be created within the virtual memory device associated with said device driver; and~~

~~(d) utilizing said device driver to allocate anonymous storage space within said anonymous storage system for storage of information relating to said non-persistent data file.~~

a1

8 ~~67~~. The method of claim ~~66~~ 7 further including the step of releasing said allocated anonymous storage space in response to the storage media file system invoking a callback procedure in said device driver.

Sub B3 ~~68. The method of claim 66 wherein said computer system includes a second device driver for managing a secondary storage device, said method further including the step of configuring each driver device to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.~~

10 ~~69~~. The method of claim ~~66~~ 7 further including the step of storing data and metadata related to said non-persistent data file in said anonymous storage space.

11 ~~70~~. The method of claim ~~66~~ 7 wherein said operating system is a UNIX-based operating system.

Sub B4 ~~71. The method of claim 66 wherein said mounting step (a) further includes the step of mounting said device driver using a delayed-write mode.~~

~~72. The method of claim 66 wherein said virtual storage device further includes a swap file system for allocating anonymous storage space in said anonymous storage system, said method~~

further including the step of using said device driver to manage information stored in said anonymous storage system by invoking procedures defined within said swap file system.

73. A computer program product for use in a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, said operating system including a storage media file system (FS) for cataloging files, the computer program product comprising a computer usable medium having computable readable code embodied therein for implementing a temporary file system in the computer system, the computer readable code comprising:

computer code for providing a file manager within the file system for managing persistent data files and non-persistent data files within said operating system;

computer code for mounting and attaching within the operating system a device driver for representing a virtual memory device used for implementing at least one non-persistent data file within the computer system;

computer code for implementing an anonymous storage system for use by said virtual memory device;

computer code for using the file manager to invoke procedures within said device driver for accessing desired information within said virtual memory device relating to said at least one non-persistent data file;

computer code for using said device driver to effect access of said desired information within said anonymous storage system; and

a computer readable medium that stores the computer codes.

a1

15 ~~74~~. The computer program product of claim 14 ~~73~~ wherein the computer readable medium is a data signal embodied in a carrier wave.

Sub B6

75. The computer program product of claim 73, wherein said computer system includes a second device driver for managing a secondary storage device, said computer program product further including computer code for configuring each driver device to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.

17 ~~76~~. The computer program product of claim 14 ~~73~~ including computer code for storing the data portion and metadata portion of a non-persistent file in said anonymous storage system.

18 ~~77~~. The computer program product of claim 14 ~~73~~ wherein said operating system is a UNIX-based operating system.

Sub B6

~~78. The computer program product of claim 73 further including computer code for mounting said device driver using a delayed-write mode.~~

~~79. The computer program product of claim 73 further comprising:~~

~~computer code for implementing a swap file system for allocating anonymous storage space in said anonymous storage system; and~~

~~computer code for using said device driver to manage information stored in said anonymous storage system by invoking procedures defined within said swap file system.~~

~~80. A computer program product for use in a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, said operating system including a storage media file system (FS) for cataloging files, the computer program product comprising a computer usable medium having computable readable code embodied therein for implementing a temporary file system in the computer system, the computer readable code comprising:~~

~~computer code for providing a file manager within the file system for managing persistent data files and non-persistent data files within said operating system;~~

~~computer code for mounting and attaching within the operating system a device driver representing a virtual memory device used for implementing non-persistent data files within said computer system;~~

~~computer code for implementing an anonymous storage system for use by said virtual storage device;~~

~~computer code for creating a non-persistent data file by invocation of a procedure of the storage media file system, while indicating that said non-persistent data file is to be created within the virtual memory device associated with said device driver;~~

~~computer code for utilizing said device driver to allocate anonymous storage space within said anonymous storage system for storage of information relating to said non-persistent data file; and~~

a computer readable medium that stores the computer codes.

22 ~~81~~. The computer program product of claim 21 ~~80~~ wherein the computer readable medium is a data signal embodied in a carrier wave.

23 ~~82~~. The computer program product of claim 21 ~~80~~ further including computer code for releasing said allocated anonymous storage space in response to the storage media file system invoking a callback procedure in said device driver.

Sub B7

~~83. The computer program product of claim 80, wherein said computer system includes a second device driver for managing a secondary storage device, said computer program product further including computer code for configuring each driver device to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.~~

26 ~~84~~. The computer program product of claim 21 ~~80~~ further including computer code for storing data and metadata related to said non-persistent data file in said anonymous storage space.

26 ~~85~~. The computer program product of claim 21 ~~80~~ wherein said operating system is a UNIX-based operating system.

A1 Sub B8

~~86. The computer program product of claim 80 further including computer code for mounting said device driver using a delayed-write mode.~~

~~87. A system for implementing temporary files in a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, the operating system including a file system (FS) for cataloguing files, the system comprising:~~

~~a file manager configured within the file system for managing persistent data files and non-persistent data files within said operating system; and~~

~~a device driver mounted and attached within the operating system for representing a virtual memory device used for implementing at least one non-persistent data file within said computer system;~~

~~said virtual memory device including an anonymous storage system;~~

~~said device driver being responsive to instructions from the file manager for invoking procedures to access desired information within said virtual memory device relating to said at least one non-persistent data file;~~

~~said driver device including means for effecting access of said desired information within said anonymous storage system.~~

~~88. The system of claim 87 wherein said computer system includes a second device driver for managing a secondary storage device, and wherein each driver device is configured to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.~~

30

~~89~~. The system of claim ~~87~~ 28 wherein a data portion and a metadata portion of said at least one ~~a~~ non-persistent file are stored in said anonymous storage system.

31

~~90~~. The system of claim ~~87~~ 28 wherein said operating system is a UNIX-based operating system.

91. The system of claim 87 wherein said driver device is mounted within said operating system using a delayed-write mode.

92. The system of claim 87 wherein said virtual storage device further includes a swap file system for allocating anonymous storage space in said anonymous storage system, and wherein said device driver includes means for managing information stored in said anonymous storage system by invoking procedures defined within said swap file system.

[illegible]

93. An apparatus for implementing temporary files in a computer system having an operating system (OS) which is structured to allow multiple different file system types to coexist within the operating system, the computer system including at least one volatile memory storage device and at least one non-volatile memory storage device, said operating system including a storage media file system (FS) for cataloging files, the apparatus comprising:

a file manager configured within the storage media file system for managing persistent data files and non-persistent data files within said operating system; and

a device driver mounted and attached within the operating system for representing a virtual memory device used for implementing at least one non-persistent data file within said computer system;

said virtual memory device including an anonymous storage system;

said device driver being responsive to instructions from the storage media file system for invoking a procedure for creating a non-persistent data file within the virtual memory device.

35

~~94~~. The apparatus of claim ~~93~~ 34 wherein said device driver includes means for releasing said allocated anonymous storage space in response to the storage media file system invoking a callback procedure.

Sub B1

95. The system of claim 93 wherein said computer system includes a second device driver for managing a secondary storage device, and wherein each driver device is configured to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system.

37 ~~96~~. The system of claim ~~93~~ 34 wherein said device driver includes means for allocating anonymous storage space within said anonymous storage system for storage of data and metadata relating to said non-persistent data file.

38. ~~97~~. The apparatus of claim ~~93~~ 34 wherein said operating system is a UNIX-based operating system.

A1

Sub B4 98. The apparatus of claim 93 wherein said device driver is mounted within said operating system using a delayed-write mode.

99. The apparatus of claim 93 wherein said virtual storage device further includes a swap file system for allocating anonymous storage space in said anonymous storage system, and wherein said device driver includes means for managing information stored in said anonymous storage system by invoking procedures defined within said swap file system.—

REMARKS

The Examiner objects to the title of the invention of the application as being non-descriptive. In accordance with the Examiner's suggestions, the application has been amended to include a new title which is clearly indicative of the invention to which the claims are directed. Accordingly, it is believed that this objection has been overcome.

On page 3 of the Office Action, the Examiner objects to language used to define the various claims of the application as being vague and/or ambiguous. During a telephonic interview between the undersigned attorney and the Examiner on July 9, 1998, the various objections by the Examiner to the presently pending claims in the application were discussed. In addition the inventive concepts of the present application, as well as their benefits and advantages were discussed and compared against the prior art, including the references cited by the Examiner, Takasaki et al. (U.S. Patent No. 5,088,031). A brief summary of at least some of the details discussed during the telephonic interview are herein described below.

In light of the Examiner's various objections to the vagueness and ambiguity of the claims, it was agreed that the claims of the present application would be revised to more clearly reflect the inventive concepts of the present invention. Accordingly, claims 1-59 have been canceled form the application and replace with newly added claims 60-99. Although it is believed that claims 1-59 are neither anticipated by nor obvious in view of Takasaki et al., claims 1-59 have been canceled from the application. Newly added claims 60-99 have been added to the application for clarification purposes in order to more clearly define the present claimed invention. It is believed, therefore, that the Examiner's objections based on vagueness and ambiguity have now been overcome.

As described in the background of the invention of the present application, the present invention is directed to an efficient and easy technique for implementing temporary files in a computer system having an operating system which is structured to allow multiple different file system types to coexist within the operating system. For example, in UNIX systems, two file systems are used for managing the various files stored within the system. A first system is the UNIX file system (UFS), and a second system is the temporary file system (TempFS). The use of different file systems for managing files within a single operating system has a number of disadvantages. First, the requirement of providing an analogous interface and generally similar behavior in all file systems within the UNIX operating system results in the TempFS file system being rather comprehensive and complex. The complexity of this temporary file system makes maintenance of the temporary file system a sizable task. Second, the implementation of modifications in the UNIX file system require corresponding modifications to the temporary file system which may or may not be easily implemented. Third, since the temporary file system in UNIX maintains a separate catalog for temporary files which is different than the catalog of files maintained by the UNIX file system, and since the metadata of the temporary files in TempFS is stored in volatile RAM, the use of two separate file system within a single operating system inefficiently uses up system resources which could otherwise be used by the system for other applications. What persists, therefore, as an unsolved need in the industry, is a temporary file system which obviates much of the duplicate maintenance required by conventional temporary file systems.

The present invention provides an easy and efficient solution to the many of the problems highlighted above by implementing the temporary file system within the UNIX file system, wherein a new device driver which manages a virtual memory storage device which is used for implementing temporary files in anonymous storage via a swap file system. Thus, as stated, for example, on pages 3-4 of the specification, the devtemp device driver of the present application provides the functionality of temporary files by representing a virtual device and including the virtual device among several devices within which the UNIX file system (or primary disk file system) can store files.

Rather than storing data in and retrieving data from a physical device, the devtemp device driver stores and retrieves non-persistent (or temporary) data from anonymous storage through a swap file system which manages anonymous storage for a virtual memory system. The UNIX file system uses the devtemp device driver in the same manner the file system uses conventional device drivers which store data in and retrieve data from secondary storage devices. Since the devtemp device driver represents a virtual device whose data are stored in anonymous storage and managed by the swap file system, the total amount of storage available for storing data of temporary files is shared among the various components of the computer system to allow more efficient use of the system resources. Additionally, since temporary files in the present invention are implemented by a device driver in the same manner that the file system uses conventional device drivers, no substantial modification to other file system is required to implement temporary files. Furthermore, any modification to the file system is automatically implemented for the temporary files as well since one file system (e.g. the disk file system or storage media file system) implements both permanent and temporary files. In addition, device drivers are generally considerably less complex than a file system such as the UNIX file system. Accordingly, implementation of the devtemp device driver is significantly less complicated unless arduous than implementation of a separate temporary file system which, in conventional systems, simulates the UNIX file system, and yet retains the functionality and performance of conventional implementations of temporary files. Thus, the present invention represents a significant improvement over the prior art.

In contrast, Takasaki is directed to a file control system for virtual machines capable of executing simultaneously a plurality of operating systems in a single computer system. A virtual machine monitor (VMM) is used in Takasaki for coordinating the interface of the operating systems of each virtual machine within the computer system. An illustration of the computer system which Takasaki describes is shown, for example, in his FIG. 1. As shown in FIG. 1, a real machine 11 includes a virtual machine monitor (VMM) 14 which is a control program used to create a plurality of virtual machines 12 within the real machine. Each of the virtual machines 12 runs a separate operating system 15 which maintains its own file system that is separate and distinct from the other file systems within the operating systems of the other virtual machines. Since each virtual machine is capable of running an independent operating system and since each of the operating systems utilized its own file system, the management and shared use of files among the virtual machines was very slow and complex. Thus, as stated in columns 3-4 of Takasaki, one of the primary objectives of Takasaki is to provide a file control system for virtual machines which is capable of speeding up and/or simplifying the management of the files possessed by the virtual machines. Another object of Takasaki is to provide a file control system for virtual machines which permits simplification of the file management on the side of the operating system and facilitation of common or shared use of files among the virtual machines as well as reduction in overhead.

From this description it is quite clear that Takasaki is addressing a problem which is different than the problems addressed by the invention of the present application. More specifically, Takasaki is attempting to coordinate management and sharing of files in a plurality of virtual machines where each machine runs a separate operating system and file system. Takasaki does not teach, describe, or suggest a technique for addressing the problems described in the present application related to a multiple file system within one operating system.

For example, let us suppose that each of the operating systems 15 in FIG. 1 of Takasaki describe a UNIX operating system. Clearly it can be seen that in such an example, each of the UNIX operating systems, for example OS1 (Takasaki, Fig. 1), would still utilize a separate UNIX file system and a separate temp file system within that virtual machine running that operating system. Takasaki does not teach or suggest the technique of the present invention for implementing a temporary file system within the UNIX file system by providing a device driver for implementing temporary files using anonymous storage. Thus, as described in detail below, it is believed that the present claimed invention is neither anticipated by nor obvious in view of Takasaki. In addition, none of the other cited prior art references teach, disclose or suggest the features of the present claimed invention as defined in newly added claims 60-99.

Claim 60 of the present application defines a method for implementing a temporary file within a file system of a computer system having an operating system which is structured to allow multiple different file systems to coexist within the operating system. The method includes a number of steps which, for purposes of illustration, are explained below.

One step is the providing of a file manager within the file system for managing persistent data files and non-persistent data files. As described in the specification of the present application, both persistent data files (e.g. non-temporary files) and non-persistent data files (e.g. temporary files) are managed by one file system, namely the UNIX file system (or, in more general terms, the storage media file system) via a file manager which, for example, may include VSF block 108 and/or VNODES block 110 (Fig. 1). In contrast, as described in the background of the present application, for example, conventional UNIX systems use UFS to manage persistent data files, and TempFS for managing non-persistent data files. It is noted that neither Takasaki nor any of the other cited prior art references teach, disclose, or suggest this feature of claim 60. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art.

Another step of claim 60 includes mounting and attaching within the operating system a device driver for representing a virtual memory device used for implementing at least one non-persistent data file within the computer system, wherein the virtual memory device includes an anonymous storage system. Neither Takasaki nor any of the other cited prior art references teach or suggest this feature. No where is it described, taught, or suggested in any of the cited prior art references for providing a device driver representing a virtual memory device which is used for implementing temporary data files within a computer system as defined in claim 60 of the present application. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art.

Another step of claim 60 relates to <u>using the file manager to invoke procedures within the device driver for accessing desired information within the virtual memory device relating to the temporary file</u>. No where is this feature taught, described, or suggested in any of the cited prior art references. In fact, in conventional UNIX systems, it is TempFS which invokes procedures for accessing desired information within anonymous storage. In contrast, according to the technique of the present invention, for example, it is the UNIX file system (104, FIG. 1) which invokes procedures within the device driver (102, FIG. 1) for accessing desired information within the virtual memory device relating to a temporary file. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art.

Another step of the method of claim 60 relates to <u>using a device driver to effect access of the temporary file within the anonymous storage system</u>. Thus, for example, as stated in the specification, page 17, lines 10-13, to realize temporary files in anonymous storage, the device driver 102 is configured to represent a virtual device, and data access of the virtual device is realized in anonymous storage through swap file system 106. In contrast, in conventional UNIX systems, it is TempFS which effects access of the desired temporary data within the anonymous storage system, not the device driver. Further, there is no teaching or suggestion in either Takasaki nor any of the other cited prior art references for this feature of claim 60. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art.

Based on the above arguments, it is submitted that claim 60 of the present application is neither anticipated by nor obvious in view of Takasaki, the other cited art references, or common knowledge in the art. Accordingly, newly added claim 60 is believed to be allowable.

Additionally, the extra feature defined in claim 61 of the present application is also believed to be allowable over and above the features of claim 60. This feature of claim 61 is directed to <u>configuring each driver device in the computer system to implement an interface and an operating behavior which are substantially similar to each other driver device within said computer system</u>. Support for this feature may be found in the specification, for example, page 9, lines 20-28. <u>By using this feature in the present claimed invention, a temporary file system may be implemented within the</u>

UNIX file system according to standard techniques used for implementing non-temporary files. In contrast, conventional UNIX systems implement temporary files through TempFS, which uses different procedures for implementing files than that of UFS. It is noted that there is no teaching or suggestion in either Takasaki nor any of the other cited prior art references for this feature of claim 61. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art. Claims 68, 75, 83, and 95 are also believed to be allowable for similar reasons.

The extra feature defined in claim 62 of the present application is also believed to be allowable over and above the features of claim 60. This feature of claim 62 is directed to storing a data portion and a metadata portion of a non-persistent file in the anonymous storage system. Support for this feature may be found in the specification, for example, page 3, lines 25-28, which states that the file system of the present invention uses the devtmp device driver in the same manner that the UNIX file system uses conventional device drivers which store data and retrieve data from secondary storage devices.

As commonly known to those skilled in the art, data files include both a data portion and a metadata portion which describes the file attributes. In conventional UNIX systems, the metadata portion of the temporary file (which is handled by TempFS) is stored in volatile RAM while the data portion of the file is stored in anonymous storage using a swap file system. In non-temporary files (e.g. persistent data files) however, the UNIX file system stores both the metadata portion and the data portion of the file in a secondary storage device using a secondary storage device driver. The UNIX file system does not have the ability to store any data in volatile RAM as TempFS does.

Since the temporary file technique of the present invention utilizes the UNIX file system and a device driver configured to operate in the same manner as device drivers which store data in conventional secondary storage devices, both the metadata portion and data portion of the temporary file is stored in anonymous storage using swap file system. The technique of the present invention is advantageous over that of the prior art because, for example, it simplifies the file system procedures by creating a device driver for implementing temporary files which functions in the same manner as a conventional device driver associated with secondary storage devices. Additionally, it is noted that this feature is neither taught, suggested, nor described in either Takasaki or any of the other cited art references for this feature of claim 62. Thus, this feature of the present claimed invention is believed to be both novel and unobvious in view of the prior art. Claims 69, 76, 84, and 96 are also believed to be allowable for similar reasons.

Claim 66 of present application defines a method for implementing a non-persistent data file within a file system in a computer system having an operating system which is structured to allow multiple different file systems to coexist within the operating system. It is noted that at least some of the method steps described in claim 66 are similar in language to at least some of the steps defined in claim 60 . Therefore, claim 66 is submitted to be unanticipated and unobivous in view of the cited prior art references for those reasons discussed above in support of claim 60. Claim 66 is therefore believed to be allowable.

Additionally, the additional feature of claim 67 is believed to be allowable over and above the features of claim 66. Claim 67 defines the method of claim 66 further including the step of <u>releasing allocated anonymous storage space in response to the storage media file system invoking a callback procedure in the device driver</u>. Support for this claim language may be found in the specification, for example, page 23. As stated in lines 15-24, in conventional device drivers, call backs representing release of one or more blocks of data in a physical device are not received or are not recognized since device drivers managing physical secondary storage devices do not care whether a particular block is free. Instead, such device drivers simply effect whatever data access is requested directly or indirectly by any of the components of FIG. 1. However, since the devtemp driver 102 shares anonymous storage with other components of the operating system, the devtemp driver releases anonymous storage when such anonymous storage is no longer needed. The UNIX file system informs the devtemp device driver when anonymous storage is no longer needed by operation of a call back mechanism which informs the devtemp device driver that one or more blocks of the virtual device are freed. It is noted that this feature is neither discussed, taught, nor suggested in any of the cited prior art references including Takasaki. For these reasons, it is believed that claim 67 is unanticipated and unobvious in view of the prior art, and is therefore believed to be allowable. Claims 82, and 94 are believed to be allowable for similar reasons.

Independent claims 73, 80, 87, and 93 include language which is similar to the language defining the features of claims 60, 62 and 67. Therefore, the arguments in favor of the allowability of claims 60, 62 and 67 may also be applied to independent claims 73, 80, 87 and 93. For these reasons, it is submitted that claims 73, 80, 87, and 93 are neither anticipated by nor obvious in view of the cited prior art references not by common knowledge in the art. Claims 73, 80, 87 and 93 are therefore believed to be allowable.

Claims 60-99 are presently pending in the application. For the reasons stated above, it is submitted that claims 60-99 are neither anticipated by nor obvious in view of the cited art references or common knowledge in the art. For these reasons, claims

60-99 are believed to be allowable, and an early indication of the allowability of all the claims is earnestly solicited.

In accordance with the telephonic interview discussion, the Examiner is invited to telephone the undersigned attorney if any matters remain or arise affecting the allowability of the presently pending claims. In the event undersigned attorney is not able to be reached, please contact Steven D Beyer (at the same office), who will handle the case in my absence.

Respectfully submitted,
BEYER & WEAVER, LLP

[illegible]

Dean Wolf
Reg. No. 37,260

P.O. Box 61059
Palo Alto, CA 94306
Tel: (650) 493-2100