METHOD AND APPARATUS FOR PER-PROCESS MOUNTING OF FILE SYSTEMS IN A HIERARCHICAL FILE SYSTEM ENVIRONMENT

BACKGROUND OF THE INVENTION

1. FIELD OF THE INVENTION:

The field of the invention is computer systems employing hierarchical file systems.

2. ART BACKGROUND:

Computer systems available today range in cost and sophistication from low cost micro-processor personal computers to sophisticated, high cost mainframe computers. However, the computer systems can be said to include the same basic components: at least one central processing unit which controls the performance of computer operations, input/output units which control the input and output of information within and out of the computer system, and memory or storage space which stores information used by the CPU to perform computer operations.

The CPU is controlled by the operating system which dictates how certain computer operations are to be performed. One portion of the operating system is the file system which determines and controls the file structure in the computer system.

One type of file structure in use is referred to as a hierarchical file structure. In a hierarchical file structure files are organized into a tree structure having a single root node referred to as the "root". One or more nodes may then be connected to the root and one or more nodes may be connected to each node. Every non leaf node (that is, a node which has a node connected to it) of the tree structure is considered to be a directory of files; and the files at the leaf nodes of the tree are either directories, regular files, or special device

files. The name of any file is identified by a path name which describes how to locate the file in the file system hierarchy. The path name is a sequence of component or node names separated by "/" characters. A component name is a sequence of characters that designates a file name uniquely contained in the previous component. A full path name starts with the "/" character which specifies a file that can be found by starting at the file system root and traversing the file tree following the branches that lead to successive component names of the path names. For example, the path name /usr/src/cmd/one.c represents the portion of a tree structure represented in Fig. 1 and identified as the path through nodes 10, 15, 20, 25 and 30.

There are numerous commands which manipulate, refer to, or access the file structures and files within the file structure. Such commands include the "mount" and "unmount" commands. The mount command connects a first file system (the term "file system" will hereinafter be used, as it is used in the art, to identify a hierarchical file structure existing in a computer system) to a second file system while the unmount system command disconnects the first file system from the second file system. Once the file system is mounted, the information in that file system is accessed through the directory of the root file system it is mounted upon. This is illustrated in Fig. 2. Root file system 40, has directories or nodes bin 42, src 45 and usr 46. The file system 70 is mounted on directory usr and the node 72 may be accessed using the path name /usr/bin/date.

Further information may be found on the mount command in most operating systems manuals. More specific information may be found in the context of the UNIX operating system environment in: Bach, J. The Design of the Unix Operating System, (Prentice-Hall 1986). As can be seen from the above-description, file systems are mounted on a directory-by-directory basis; that is, at least one directory is mounted on a file system comprising at least one

directory. In a convention UNIX operating system, once a file system has been mounted on a specific directory, the inode of that directory may no longer be accessed by processes.

However, as the sophistication of the users and the applications executed on the computer system increase, it has become desirable to have the capability to mount file systems on a per-process basis even though the processes are located within the same directory. Thus, it is advantageous to have several mounts performed within one directory such that different mounts are utilized according to the current process.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a computer system for per-process mounting of file systems in a hierarchical file system structure.

To provide per-process mounting, a temporary directory is created off the current root directory. The file system, starting at the root directory, is then mounted on the temporary directory using an innovative loop-back file system function which results in the file system being virtually duplicated having the temporary directory as the root directory. A mount of the file system containing process specific files can then be performed on the temporary directory without affecting the original file system. A change root directory command is then executed to change the root directory to the temporary directory and execution then continued with respect to the mounted file system having the temporary directory as its root.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the system of the present invention will be apparent from the following description in which:

FIG. 1 illustrates a hierarchical file structure employed in computer systems.

FIG. 2 illustrates the system command mount which is employed in prior art operating systems.

FIG. 3 is a block diagram of the kernel in the UNIX operating system employed in the preferred embodiment of the system of the present invention.

FIG. 4 is a flow chart describing the preferred embodiment of the process of the present invention.

FIG. 5 is a block diagram illustrative of the preferred embodiment of the system of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Through the system of the present invention per-process mounting may be achieved on a computer system employing a hierarchical file structure thereby adding increased flexibility and capabilities to the computer system.

One operating system which uses a hierarchical file structure is the UNIX® (UNIX is a registered trademark of AT&T) operating system. Although the preferred embodiment is described in a UNIX operating system environment, it is evident to one skilled in the art that the system of the present invention may be applied to other computer systems employing an operating system which utilizes a hierarchical file structure.

In a UNIX-based computer system, the operating system interacts directly with the hardware providing common services to programs and isolating the software and users from the hardware idiosyncrasies of a particular computer system. Viewing the system as a set of layers, the operating system is commonly called the "system kernel" or just "kernel", emphasizing its isolation from the user programs. Because the programs are independent of the underlying hardware, it is easy to move the programs between UNIX systems running on different hardware. The UNIX file system, located within the kernel, organizes and controls access to the file structures.

The kernel is depicted generally in Fig. 3. Fig. 3 shows 3 levels: user/programs 74, kernel 75 and hardware 76. The system call module represents the border or interface between the user or user programs and the kernel. The system calls are divided into those system calls that interface with the file system 77 and those that interact with the process control system 78. The process control system is responsible for process synchronization,

interprocess communication, memory management and process scheduling. The file system communicates with the hardware control module 80 which interfaces to the computer system hardware and manages files, allocates files space, administers free space, controls access to files and retrieves data for users and programs.

The system will herein be described in reference to Figs. 4 and 5. At block 100, Fig. 4, a temporary directory is created. This directory is preferably created off the root directory so as to eliminate any subsequent confusion with the following process steps and to clearly distinguish the temporary sub-directory from other sub-directories of the same root. The directory identification as temporary is used because it is preferred that after the process, for which the file system was created, has completed execution, the temporary directory and nodes connected to it are removed such that the file system is restored to its previous structure. However, the temporary directory does not have to be removed and can remain indefinitely in the file system structure. The first step described in Fig. 4 block 100 is illustrated by Figs. 5a and 5b. Fig. 5a shows the root directory 200, sub-directory usr 205 and sub-directory of usr identified as src 210. Fig. 5b illustrates the file system after the step 100 of Fig. 4. The temporary directory tmp 215, has been created having a sub-directory off of tmp identified as foo 220.

At block 105, Fig. 4, the file system comprising directories 200, 205, 210, 215 and 220 are mounted on the directory foo 220, whereby the directory foo 220 reflects the contents of the original root directory "/" 200. This is illustrated in Fig. 5c, wherein the file system shown in Fig. 5b is duplicated by mounting the system on directory foo 220, resulting in the structure, illustrated by the rectangular shaped nodes, foo 220 having sub-directories tmp 225, foo 230, usr 235 and src 240. This process is achieved using an innovative technique referred to as loop-back mounting in which the file

structure or file system is duplicated off a different directory or different file system by adding a second pointer which dictates the location of the directory in the file system. This process, results in the "virtual" placement of the directory in two locations. It should be noted that the apparent duplication of the directories does not result in the actual duplication of the directories; only one set of directories exist but the directories appear to be concurrently connected to two different nodes.

At step 110 Fig. 4 the file system, for example the file system that is desired to be process-specific, is mounted on the temporary file system created. Referring to Figs. 5d and 5e the file system shown in Fig. 5d comprising directories bin and etc are mounted on the file system, specifically the src sub-directory 240. The resulting structure is represented by the trapezoidal shaped nodes shown in Fig. 5e. At step 115, Fig. 4, the root directory is changed to be /tmp/foo node 220 of Fig. 5e. (In the UNIX operating system the change root directory command is CHROOT). As a result, the file system visible to the user or process is as shown in Fig. 5f; since it is visible only to the process, it is termed "per-process. The structure comprises the root node, which in Fig. 5e was directory foo 220 which now appears as the root 260, having sub-directories tmp 265, and foo 270 as well as sub directories usr 280, 75 src 280, bin 290 and etc 300.

Thus, the process may be executed within the file structure of Fig. 5f without affecting the ability of other processes executed in the original file system (depicted in Fig. 5a).

It can be seen that the steps set forth in the flowchart of Fig. 4 may be repeated, for example by creating a temporary directory tmp1 having a sub-directory foo, and mounting the directories containir.g the files to be executed with respect to another process. Furthermore the process described may again

be repeated for a second process, third process or more processes such that each process is mounted on a separate file system thereby emulating or providing per-process mounting.

The loopback mounting technique may be easily accomplished in one implementation of the UNIX operating sytem. This implementation includes a virtual file system in the kernel by which it is able to support multiple file system types such as multiple versions of UNIX and non-UNIX systems such as MSDOS. Such a virtual file system is described in Vnodes: An Architecture for Multiple File System Types in Sun UNIX, S. R. Kleiman, published by Sun Microsystems, Inc., Mountain View, California.

As described in that paper, the conventional file system independent inode of the UNIX is renamed vnode (virtual node). All file manipulation is done with a vnode object. File systems are manipulated through the virtual file system. Each virtual file system includes an analog to the mount table entry of the conventional UNIX system; this specifies the file system type, the directory which is the mount point, generic flags (e.g., read only), and a pointer to file system type specific data. This is linked to a list of mounted file systems so that file system dependent data may be provided. This virtual file system allows loopback mounts to be made.

To make a loopback mount, the user executes the MOUNT () system call. In the command, the user specifies (1) that this call is to be a loopback mount, (2) the directory which is to be mounted, and (3) the directory it is to be mounted on. As pointed out, the vnode structure in the kernel contains a pointer to the virtual file system specific data. This data includes for a loopback file, among other things, a pointer to the "real vnode" for the file. Thus, if directory A is loopback mounted onto directory B, then in the kernel, the vnode for directory B is a loopback vnode whose real vnode is the vnode

for directory A. For example, in Fig. 5(e)-(f), where the root directory "/" is mounted to the directory foo 220, the vnode for foo 220 has a pointer to the root "/" directory as its real directory.

When the kernel receives a filesystem request for directory foo220, then the kernel executes the file system independent lookup code before invoking the file system dependent lookup code for the virtual file system that the vnode is in. Since directory foo's vnode is in the loopback file system, the loopback lookup operation is performed. This operation calls the lookup operation for the real vnode, the vnode of root directory "/". In this way, the contents of directory foo are the same as the contents of the root directory "/" since the same files are found in either directory. Thus, the additional pointer which causes directory foo to point to the root directory "/" is the real vnode pointer in directory foo's loopback vnode.

While the invention has been described in conjunction with a preferred embodiment, it is evident that numerous alternatives modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description.

CLAIMS

1. A computer system comprising a central processing unit (CPU), an input/output unit and memory, said CPU being controlled by an operating system, said operating system comprising a file system module, wherein the file system module organizes and controls access to files in the computer system, said file system module organizing files into a file system having a hierarchical file structure comprising a root directory and one or more sub-directories, said file system module further comprising:

means for creating a temporary directory connected to the root directory;

means for performing a loopback mount of the file system on the temporary directory wherein said file system is virtually duplicated having the temporary directory as its root directory;

means for mounting files on the temporary directory;

means for changing the root directory to the temporary directory;

whereby the file system which appears to the process is a virtual duplication of the original file system with a special mount visible only to that process.

2. The system of claim 1 wherein the means for performing a loopback mount of the file system on the temporary directory comprises

means for creating a virtual file system; and

means for creating a pointer from the temporary directory to the root directory whereby the two directories contain the same information to the process.

* C L M S *

3. The system of claim 1 wherein the operating system is the UNIX (Registered Trade Mark) operating system.

4. The system of claim 3 wherein the means for creating a temporary directory connected to the root directory comprises the execution of the command "make directory" by the file system module.

5. The system of claim 3 wherein the means for performing a loopback mount of the file system on the temporary directory comprises

means for creating a virtual file system; and

means for creating a pointer from the temporary directory to the root directory whereby the two directories contain the same information to the process.

6. The system of claim 3 wherein the means for mounting files containing process data on the file system mounted on the temporary directory comprises the execution of the "mount" command by the file system module.

7. The system of claim 3 wherein the means for changing the root directory to the temporary directory comprises the execution of the "change root directory" ("CHROOT") command by the file system module.

* C L M S *

8. A computer system comprising a central processing unit (CPU), an input/output unit and memory, said CPU being controlled by an operating system, said operating system comprising a file system module, wherein the file system module organizes and controls access to files in the computer system, said file system module organizing files into a file system having a hierarchical file structure comprising a root directory and one or more sub-directories, a method for virtually duplicating file systems to provide per-process mounting comprising the steps of:

creating a temporary directory connected to the root directory;

performing a loopback mount of the file system on the temporary directory wherein said file system is virtually duplicated having the temporary directory as its root directory;

mounting files on the temporary directory;

changing the root directory to the temporary directory;

whereby the file system which appears to the process is a virtual duplication of the original file system and mounts made by this process in the file system are only visible to this process or any process rooted to the same temporary directory thereby providing per-process mounting.

9. The method of claim 8 wherein the step of performing a loopback mount of the file system on the temporary directory comprises the steps of

creating a virtual file system; and

creating a pointer from the temporary directory to the root directory whereby the two directories contain the same information to the process.

* C L M S *

10. The method of claim 9 wherein the operating system is the UNIX (Registered Trade Mark) operating system.

11. The method of claim 10 wherein the step of creating a temporary directory connected to the root directory comprises executing the command "make directory"

12. The method of claim 10 wherein the step for performing a loopback mount of the file system on the temporary directory comprises the steps of

creating a virtual file system; and

creating a pointer from the temporary directory to the root directory whereby the two directories contain the same information to the process.

13. The method of claim 10 wherein the step of mounting files containing process data on the file system mounted on the temporary directory comprises executing the "mount" command.

14. The system of claim 10 wherein the step of changing the root directory to the temporary directory comprises executing the "change root directory" ("CHROOT") command.

15. A central processing unit (CPU), an input/output unit and memory, said CPU being controlled by an operating system, said operating system comprising a file system module in a computer system substantially as hereinbefore described with reference to Figs. 3 to 5 of the accompanying drawings.

16. A method for virtually duplicating file systems to provide per-process mounting substantially as hereinbefore described.

* C L M S *