9801778-3
IBDS

METHOD FOR SIMULATING A COMPUTER STORAGE DEVICE

Background

The present invention relates generally to computers and software code for controlling access to a computer storage device, and, more specifically, to a system and method for re-mapping a logical reference to a first storage device to a second storage device.

In certain computers, especially lower cost computers and network personal computers (NetPC's), a limited number of storage devices are provided. For example, most NetPC's will have a single hard disk but no floppy disk. While this configuration of storage devices will meet most requirements for these computers, some programs have specific, hard-coded references to the absent storage devices. Continuing with the NetPC example, when software is being downloaded to the NetPC during the computer's manufacturing process, often a software download process expects the computer to have a floppy disk. This is mainly due to the fact that the majority of computers do have a floppy disk and therefore hard-coded references to such drives are prevalent throughout the software download process. However, since in the present example no floppy disks exist, the software download process experiences numerous errors.

There are two typical solutions to this problem. One solution is to temporarily install a floppy disk in the NetPC during the software download process,

and then later remove the drive. This solution is undesirable, however, because it is very time consuming and requires the existence of an extra floppy disk. The other solution is to modify the code in the software download process. This solution is also undesirable because the code is often used for many different computers, and thereby it is not desirable to change it. In addition, copyright and/or source code availability may prevent the code from being modified.

Summary

In response thereto, provided is a system and method for re-mapping a logical reference to a first storage device to a second storage device. In one embodiment, a computer includes a processor for running operating system code, random access memory (RAM), and a hard drive memory device (the second storage device). The computer does not, however, have a floppy disk (the first storage device). The hard drive, which can be accessed by the processor, includes at least two partitions. When the computer boots up, operating system code running on the processor includes a first logical reference to the first partition of the hard drive and a second logical reference to the non-existing floppy disk.

The logical references are stored in a table, located in RAM. If software routines running on the processor ever use the second logical reference, an error would normally occur. However, a reference-changing routine is provided for changing the logical reference from the non-existing floppy disk to the second partition of the hard drive. The reference-changing routine does this by altering the table with data from the second partition so that when the logical reference to the second storage device is used, the second partition of the first storage device is accessed.

A technical advantage achieved with the present invention is that the first storage device is not required, even though certain software routines may attempt to access the first storage device.

Another technical advantage achieved with the present invention is that the re-mapping occurs in the computer's RAM, and therefore no permanent changes are made or required in any other software routines.

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer for implementing one embodiment of the present invention.

Fig. 2 is a layout diagram of a hard drive of the computer of Fig. 1.

Fig. 3 is a flow chart of a routine to be run by the computer of Fig. 1.

Detailed Description of the Preferred Embodiment

The following disclosure incorporates by reference U.S. Patent Application Ser. No. (Atty. Docket No. DC-1313), entitled "Method for Changing Partition Mappings to Logical Drives in a Computer Memory Device" and U.S. Patent Application Ser. No. (Atty. Docket No. DC-1314), entitled "Method for Updating Partition Mappings to Logical Drives in a Computer Memory Device", filed concurrently herewith.

Referring to Fig. 1, the reference numeral 10 designates a computer having several components, including a processor 12, a bank of random access memories (RAM) 14a and read only memories (ROM) 14b, a hard disk 16, and a network connection 18. Each component is capable of communication with the processor 12, as graphically represented by a general bus 20. In the present example, the computer 10 is a NetPC running Microsoft DOS operating system code (hereinafter DOS) as provided by Microsoft Corporation of Redmond, WA. Also, the ROM 14b is stored on a network interface card (NIC) and contains remote initial program load (RIPL) code typical of a network-based computer. It is understood, however, that the NetPC 10 and its illustrated components are merely representative of many different types of computers and components and that DOS is merely representative of many different types of operating systems well known and understood by those of ordinary skill in the art.

Referring to Fig. 2, the hard disk 16 is subdivided into three different partitions: a master boot record 30, a utility partition 32, and a primary partition 34. Located in the master boot record 30 is boot strap code 40 and a master partition table 42. The master partition table includes code that represent data specific to each partition 32 and 34, as discussed in greater detail, below. It is understood that

the additional software in the master boot record 30, including the boot strap code 40, is well known in the art and, for the sake of brevity, will not be further discussed.

In the present example, the primary partition 34 is active and DOS includes a logical reference value "C:" to the primary partition by storing information specific to the primary partition in a Drive Parameter Block ("DPB") and a Drive Data Table ("DDT"). The utility partition 32, however, is not active and no information about the utility partition is stored in the DPB or DDT. Such a configuration of the NetPC 10 is typical, thereby allowing special utility and maintenance software to be stored in the utility partition 32, yet keeping such software separate from software in the primary partition 34. Also in the present example, a software download process (not shown) includes hard coded references to a logical A: drive, which typically references a floppy disk. Since a floppy disk does not exist on the NetPC 10, the download process would conventionally encounter errors at the hard coded references to the logical A: drive.

Referring to Fig. 3, a routine 100 entitled "Waffle"allows the software download process to run without encountering any errors due to the hard coded references to the logical A: drive. Prior to Waffle 100 being run, the hard disk 16 is initialized or reconfigured, the utility partition 32 is marked inactive and the primary partition 34 is active. At step 110, Waffle 100 is loaded into RAM 14a and run. Waffle is a logical-reference-changing routine and, in the present embodiment, is accessed through the network connection 18. In addition, DOS is also accessed through the network connection 18 so that the NetPC 10 may be booted from the network. Alternatively, Waffle may be provided on either or both of the partitions 32 and 34 or accessed from another source. Although not further described, Waffle may contain several progress checkpoints during which Waffle will check specific values and notify the user of errors or other conditions.

At step 112, a command is received from a user. A command "virtual" indicates to Waffle that the user wants to convert an inactive partition into a virtual floppy disk. In the present example, the utility partition 32 would be made to emulate a logical A: drive.

At step 114, the master boot record 30 is loaded from the hard disk 16 into RAM 14a. At step 116, a determination is made as to whether the logical A: drive has already been mapped to the utility partition 32, i.e., a virtual (or an actual) floppy disk already exists. If it has, execution is stopped. Otherwise, execution proceeds to step 118 where the portions of RAM 14a not being used by DOS, Waffle, or to store the master boot record 30 are flushed. This step may perform several operations. For one, if the computer 10 includes one or more caches, including disk caches, the caches are flushed. Also, all open files in use by DOS or any other open application running on the computer are closed.

At step 120, the boot record 50 for the utility partition 32 is retrieved. At step 122, the operating system's logical reference for the floppy drive is updated. In the present example, this logical reference, which is stored with DOS in RAM 14a, is an "A:" drive. This is accomplished by using a BIOS Parameter Block ("BPB") from the utility boot record 50 to update a Drive Parameter Block ("DPB") in DOS. At step 124, the BPB from the primary boot record is also used to update a Drive Data Table ("DDT") for the logical drive reference A:.

Although illustrative embodiments of the present invention have been shown and described, a latitude of modification, change and substitution is intended in the foregoing disclosure, and in certain instances, some features of the invention will be employed without a corresponding use of other features. Furthermore, additional features, such as error handling routines, may be added to the illustrative embodiment without altering the scope of the invention. Accordingly, it is appropriate that the appended claims be construed broadly and in a manner consistent with the scope of the invention.

Claims



WHAT IS CLAIMED IS:

1. A computer comprising:

a processor for running software including operating code;

a first storage device accessible by the processor, the storage device being separated into at least two partitions, the first partition being active;

the operating code including a logical reference to a second storage device and including a table for mapping the logical reference to the second storage device, wherein the second storage device is not accessible by the processor; and

a reference-changing routine for dynamically changing the logical reference to the second partition of the first storage device by altering the table;

wherein after the reference-changing alters the table, any references to the second storage device are instead made to the second partition of the first storage device.

2. The computer of claim 1 further comprising a memory system and wherein the reference-changing routine also flushes the memory system after changing the logical reference.

3. The computer of claim 2 wherein the memory system further comprises a disk cache and one or more open files and wherein the reference-changing routine flushes the memory system by flushing the disk cache and closing the one or more open files.

4. The computer of claim 1 wherein the reference-changing routine retrieves data from a boot record for the second partition, the data being used for altering the table.

5. The computer of claim 4 wherein the data includes a BIOS Parameter Block.

* C L M S *

6. The computer of claim 5 wherein the table includes a Drive Parameter Block and a Drive Data Table.

7. The computer of claim 1 wherein the operating code includes a file management system and wherein the reference-changing routine also resets the file management system after changing the logical reference.

8. The computer of claim 1 further comprising a network interface device and wherein the reference-changing routine is accessed through the network interface device.

9. The computer of claim 8 wherein the operating code is accessed through the network interface device.

10. The computer of claim 1 wherein the reference-changing routine updates the logical reference in response to a user command.



11. In a computer having a first storage device and a processor running operating code with logical references to the first storage device and a second storage device, a method for allowing a software process to reference the second storage device even if the second storage device is not accessible by the processor, the method comprising the steps of:

dividing the first storage device into two partitions;

defining the first partition as active;

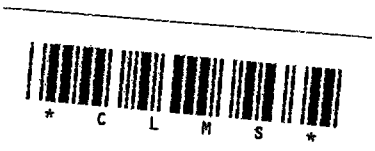
loading a boot record for the second partition and retrieving a BIOS Parameter Block therefrom;

determining if the second storage device is accessible by the processor; and

if the second storage device is not accessible by the processor, updating the logical reference value to the second storage device with information retrieved from the BIOS Parameter Block.

12. The method of claim 11 wherein the step of updating comprises updating a Drive Parameter Block and a Drive Data Table in the operating code.

13. The method of claim 11 wherein the computer includes an open file and a cache, the method further comprising flushing the cache and closing the open file.



14. A program for use in a computer including a first storage device, a processor capable of accessing the first storage device, and operating code with logical references to the first storage device and to a second storage device, wherein the second storage device is not accessible by the processor and the first storage device is divided into two partitions, the first partition being active, the program comprising:

instructions for loading a boot record for the second partition and retrieving a BIOS Parameter Block therefrom; and

instructions for updating the logical reference to the second storage device with information retrieved from the BIOS Parameter Block so that if the processor attempts to access the second storage device using the logical reference for the second storage device, the processor instead will access the second partition of the first storage device.

15. The program of claim 11 wherein the instructions for updating comprises updating a Drive Parameter Block and a Drive Data Table in the operating code.

16. The program of claim 11 wherein the computer includes an open file and a cache, the device further comprising instructions for flushing the cache and closing the open file.

17. The program of claim 11 being stored on a network and accessible by the processor through the network.

18. The program of claim 11 being stored on the first storage device.

19. The program of claim 11 being stored on a third storage device and accessible by the processor through the third storage device.