

assure a stable system is presented for operation, another step is needed.

The further step activates at the time the OS finishes loading. At this point in time in this embodiment a special driver, previously included in the OS configuration, is executed. This driver then either relocates the application to within the OS, or binds the application into the OS controlled space by “freeing” the resources for the OS, but using them at the same time. In Windows 98™ for example, the driver could use the conventions of the virtual address descriptor (VAD) in conjunction with the virtual memory manager (VMM) to mark the extent of the “recovered” space.

Although the process itself seems very complicated when compared to a conventional boot operation followed by an immediate launch of a desired application, real-time issues have to be considered to understand the advantages of the unique system provided by this embodiment of the invention. Even though an OS boot takes only 30–90 seconds, in some situations that may be an unacceptable length of time to wait for an application to be launched. For example if the user must make an emergency phone call using the computer system to be booted, he or she may have to wait 90 seconds in a conventional system. In a system according to the embodiment presented here, the pre-boot phase may be in the milliseconds or in the low seconds if an MSD such as a hard disk is used.

The embodiment described above teaches a method for loading one application to execute before, during and after OS boot. It will be apparent to one with skill in the art that the method may be extended to load and execute more than one such application. For example, one may configure a VROM or other BIOS to load both an address book, and an IP telephony application enabling IP telephone calls. One might also include an e-mail client. There are many such possibilities, and communication applications are clearly desirable candidates. There are, of course, other possibilities. For example, a writer may wish to go directly to a word processor.

There are many variant possibilities for presenting pre-boot applications. Therefore, the method and apparatus of the present invention should be afforded the broadest scope. The spirit and scope of the present invention is limited only by the claims that follow.

What is claimed is:

1. A method for calling an application on a computer before booting an operating system (OS), and continuing to execute the application during OS boot, the method comprising:

- (a) including a preboot application manager (PAM) software in a BIOS directing computer initialization and the OS boot;
- (b) calling and executing the PAM upon startup prior to OS boot;
- (c) checking a file system structure (FSS) in the BIOS by the PAM for pointers to the application;
- (d) loading the application to a RAM and executing the application under the control of the computer;
- (e) presenting, at OS boot, an address extent of a protected memory region as including the application; and
- (f) continuing to execute the application during OS boot.

2. The method of claim 1 further comprising a step for calling a special driver after OS boot to register the application and include it in the OS configuration, thereby enabling the application to execute in a stable manner after OS boot.

3. The method of claim 1 wherein the PAM is a part of a configurable virtual random access memory BIOS (VROM BIOS) implemented in flash memory.

4. The method of claim 1, wherein presenting, at OS boot, an address extent of a protected memory region comprises representing to the OS that an address space of the application is part of the BIOS, and wherein, in response, the OS will not overwrite said address space during OS boot.

5. The method of claim 1, further comprising loading, using the FSS, a driver for said mass storage into a non-volatile memory of said computer.

6. The method of claim 1, further comprising presenting, by the application, content to a user during said OS boot.

7. The method of claim 1, further comprising accessing said mass storage, by said PAM, after said OS boot.

8. The method of claim 1, further comprising detecting a flag, by said FSS, said flag to indicate that a new mass storage has not been configured, and loading parameters for said new mass storage into said FSS.

9. A computer system, comprising:
a pre-boot application manager (PAM) in a BIOS of the computer system for directing computer initialization and an operating system (OS) boot;
a random access memory (RAM); and
an operating system (OS);

wherein the BIOS is shadowed to the RAM at power-on or reset, the PAM executes to check a file system structure (FSS) in the BIOS for a pointer to an application, upon finding the application the application is loaded to the RAM and executed under the control of the computer system, and at the OS boot, the BIOS represents a BIOS space in RAM to the OS as including a space of the application.

10. The computer system of claim 9 further comprising a special driver called after OS boot to register the application and include it in the OS configuration, thereby enabling the application to execute in a stable manner after OS boot.

11. The computer system of claim 9 wherein the PAM is a part of a configurable virtual random access memory BIOS (VROM BIOS) implemented in flash memory.

12. A basic input-output system (BIOS), comprising:
a BIOS code portion for initializing a computer system and loading an operating system (OS); and
a preboot application manager (PAM) portion;

wherein, after shadowing to a RAM, the PAM portion is called and executed prior to loading the OS, and causes one or more applications to be executed under the control of the computer system before loading the OS, and wherein, at an OS boot, the BIOS represents the extent of its address space in the RAM to include the one or more applications, thereby enabling the one or more applications to continue to execute during OS boot.

13. The BIOS of claim 12 implemented as a virtual random access memory (VROM) BIOS in flash memory.