

1

**MECHANISM FOR INTUITIVELY
INVOKING ONE OR MORE AUXILIARY
PROGRAMS DURING A COMPUTER
BOOTING PROCESS**

FIELD OF THE INVENTION

This invention relates generally to computing technology, and more particularly to a mechanism for intuitively invoking one or more auxiliary programs during a computer booting process.

BACKGROUND

In a typical personal computer, there exists a processor, a basic input-output system (BIOS), a main memory, and a hard drive that stores an operating system and one or more application programs. The BIOS usually takes the form of executable instructions stored on a read-only memory (ROM). During a regular boot-up process, the processor accesses and executes the instructions in the BIOS, and under direction of the BIOS, the processor implements the booting process. The BIOS causes the processor to perform some low-level setup functions to prepare the computer for regular operation. After the setup functions are performed, the BIOS causes the processor to load and execute the operating system stored on the hard drive. By doing so, the BIOS in effect transfers control from itself to the operating system. After the operating system is loaded and executed by the processor, the boot-up process is complete and the computer is ready for operation. As this discussion shows, in a typical personal computer, the BIOS controls the booting process.

The ROM on which the BIOS resides (referred to hereinafter as the BIOS ROM) is typically quite small in size. Despite this small size, however, it has been observed that one or more auxiliary programs may be stored on the BIOS ROM. These auxiliary programs may be executed during the booting process prior to and even in lieu of the operating system on the hard drive to provide certain desired functionalities. An example of an auxiliary program is the self-contained browser disclosed in U.S. patent application Ser. No. 09/449,065 entitled "Self Contained Browser Architecture" filed on Nov. 24, 1999, the contents of which are incorporated herein by this reference. Auxiliary programs are generally self-contained, meaning that they comprise all of the components that they need to operate. As a result, they do not need the operating system on the hard drive to function. This can be quite advantageous because even if the hard drive fails or the operating system becomes corrupted, the auxiliary programs are not affected. They can still function. Hence, the auxiliary programs are impervious to many system failures.

Despite their advantages, auxiliary programs have suffered thus far from one major drawback, which is that they are difficult and non-intuitive to invoke. Because they are executed during the booting process, auxiliary programs have thus far been invoked via the BIOS. Unfortunately, invoking a program via the BIOS is not an intuitive process.

Two methods are currently implemented to invoke an auxiliary program via a BIOS. The first method involves manually adjusting the settings of the BIOS to cause the BIOS to invoke the auxiliary program. To do so, a user depresses a special key (e.g. the F1 key) at the beginning of the booting process to enter BIOS setup mode. Once in this mode, the user updates the necessary settings to cause the BIOS to invoke the auxiliary program. After the settings are

2

updated, the user reboots the computer, and during the subsequent booting process, the BIOS causes the auxiliary program to be executed. If the user thereafter wishes to terminate execution of the auxiliary program, and to execute the operating system on the hard drive, the user has to reboot the computer. At the beginning of the subsequent booting process, the user depresses the special key to once again enter BIOS setup mode. While in this mode, the user updates the necessary settings to cause the BIOS to no longer invoke the auxiliary program. Thereafter, the user reboots the computer again, and during the subsequent booting process, the BIOS will bypass execution of the auxiliary program and proceed to executing the operating system. The problem with this approach is that it requires significant technical knowledge on the part of the user. Many computer users lack such knowledge and sophistication; thus, they cannot take advantage of the auxiliary program.

A second method involves the use of a "hot key". During the booting process, if a user depresses a specific key recognized by the BIOS as a command to execute the auxiliary program, then the BIOS will cause the processor to execute the auxiliary program. Otherwise, the BIOS will cause the processor to continue with the booting process by loading and executing the operating system. If the auxiliary program is executed, and the user subsequently wishes to execute the operating system, the user has to reboot the computer. During the subsequent booting process, the user foregoes depressing the hot key, which causes the BIOS to load and execute the operating system. While not as burdensome as the first approach, the "hot key" approach is still not very intuitive or convenient. It requires the user to know which key is the hot key. It also requires the user to understand the significance of depressing the hot key (i.e. that it stops the booting process). As noted above, many computer users lack this level of sophistication. As a result, they may not know how to, or they may choose not to invoke the auxiliary program to avoid any additional complication. In either case, the advantages offered by the auxiliary program are not exploited.

As shown by the above discussion, the current mechanisms for invoking auxiliary programs during a booting process leave much to be desired. As a result, an improved mechanism is needed.

SUMMARY

In accordance with one embodiment of the present invention, there is provided a mechanism in which, during a booting process, the BIOS causes one or more auxiliary programs to be automatically executed. By doing so, the BIOS transfers control of the booting process to the auxiliary programs. Thereafter, it is up to the auxiliary programs to determine whether to continue execution, or to proceed with the booting process. Should the auxiliary programs determine that execution of the auxiliary programs should continue, the booting process is halted and the operating system is not loaded or executed.

In one embodiment, to determine whether execution of the auxiliary programs should continue, the auxiliary programs do not monitor for a "hot key" that indicates execution should continue. Instead, they do the converse. The auxiliary programs monitor for any user input. If any user input is received, unless the user input indicates specifically that execution of the auxiliary programs should not continue, the auxiliary programs will continue execution. Thus, in this embodiment, the default is to continue execution of the auxiliary programs. This makes invocation of the aux-