DEVICE DRIVER ARCHITECTURE SUPPORTING EMULATION ENVIRONMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is generally related to the design of device drivers utilized in computer operating systems to define and establish an interface between the core operating system and typically hardware devices and, in particular, to a modular device driver architecture providing a virtualized, context switchable interface environment within which to operate typically hardware devices in support of operating system functions, specifically including information display functions.

2. Description of the Related Art

In conventional computer systems, software operating systems provide generalized system services to application programs, including utility and daemon programs. These system services conventionally include access to whatever individual hardware peripheral devices, each generally presenting a well defined hardware interface to the computer system, may be attached directly or indirectly to the computer system. Device drivers, implemented as software modules or components that can be integrated into an operating system, are typically used to provide well defined software application program interfaces (APIs) to the operating system and application programs for each of the hardware interfaces. Device drivers often provide a degree of device independence or virtualizing that may simplify the interaction of an application program or operating system with the specifics of a particular class of hardware interface, such as a video controller. Conventionally, for each implementation underlying a particular hardware interface, a specific device driver is used to implement an otherwise common API that is presented to the application programs and operating system.

A number of problems are inherent in conventional device driver designs. First, conventional device drivers are specific to a particular hardware interface and the function of the underlying device or controller system. Thus, whenever a new or different version of a hardware controller is produced, a new device driver equally specific to the new or different hardware must also be developed. Where there are many different versions of a hardware device, a generally like number of device drivers must be developed. Alternately, single combination device drivers may be constructed to support multiple versions or types of devices. Such device drivers typically incorporate multiple device specific drivers that are otherwise substantially independent of one another into single binary file.

The effective number of device drivers needed to support a particular piece of hardware is also dependant on the number and differences in the operating system environments within which the hardware is to be used. In all but the most closely related operating systems, a substantial redevelopment of the device driver is required to both provide for the proper ability to incorporate the device driver into a particular operating system and, perhaps more significantly, to provide a logically similar though often entirely different API to the operating system and applications. Usually, the detailed definition of the API of the device driver governs the detailed design of the device driver itself. Consequently, device drivers for the same hardware but for different operating systems are often almost completely independently developed.

Another consideration that affects the number of device drivers that are required to support a particular hardware

controller arises from the nature of other hardware and systems that are connected to a particular controller. Again, to provide flexibility in the detailed construction of computer systems, a hardware controller may be capable of supporting a number of distinctly different modes of operation. For example, a video controller may be able to support a significant range of video display resolutions and color depths. However, the range may be constrained by direct limitations such as the amount of video RAM actually implemented on a particular video controller and indirectly by the maximum vertical and horizontal frequencies of an attached video display. The requirements of particular applications may also drive the selection of a particular mode of operation that must be supported by a device driver. Conventionally, a number of device drivers are provided with the hardware controller, each supporting a different set of one or more modes of operation. One of the provided device drivers must therefore be selected for operating system incorporation based directly on the configuration of the particular computer system. Aside from the difficulties of picking a device driver that supports the desired set of operating modes, a substantial difficulty exists in preemptively determining the variety of modes that different individual device drivers are to support. Although the individual drivers may differ only by some modest amount, their number may be significant in terms of development.

A second problem, in part a consequence of the first, is that each device driver must be thoroughly tested in the full variety of environments that the device driver may be used in to ensure commercially acceptable operation. Conventionally, device drivers are essentially monolithic software modules that are incorporated bodily into the operating system. As such, testing of even minor variants of a device driver for a particular operating system requires that the full suite of operational function and application compatibility tests be run to verify correct operation of the device driver. Selective functional testing is generally inappropriate due to the real possibility of collateral operational errors arising from any modification of a monolithically coded device driver. Given the substantial number of effectively different device drivers conventionally supported for a reasonably complex hardware controller and the size and substantial extent of corresponding test suites, the testing of device drivers represents a substantial expense and a very significant delay in bringing new or improved versions of a product to market.

A third problem with conventional device driver designs is that they provide for a substantially static type of hardware controller management. In general, device drivers establish a single set of operating parameters for the hardware controller being managed by the device driver. The operating system and the application programs executing on the computer system all must accept the parameters of this static mode or essentially fail to operate correctly.

In limited instances, a conventional device driver may make some modes available or visible to application programs. To make use of these modes, the device driver therefore relies on application programs to have essentially compiled-in hardware dependencies. In such cases, the application programs may invoke a mode change, though with potential detrimental effects on the other executing programs that, even if capable of invoking a mode switch, are effectively unaware of any such switch.

Some typically multi-tasking operating systems can perform limited dynamic hardware controller mode switching, though only through an operating system supported state change. In these cases, the operating system essentially