

Field of the Invention

The FIELD OF THE INVENTION

This invention relates to computer software a computing system, and more specifically to software an architecture that affects and extends services -exported through application libraries.

Background of the Invention

BACKGROUND OF THE INVENTION

Computer systems are designed in such a way that software application programs share common resources. It is traditionally the task of ~~the an~~ operating system to provide mechanisms to safely and effectively control access to shared resources. In some instances the centralized control of elements, critical to software applications, hereafter called critical system elements (CSEs) creates a limitation caused by conflicts for shared resources.

~~In some cases the centralized control of elements critical to software applications creates a limitation caused by conflicts for shared resources. Two~~For example, two software applications that require the same file, yet each ~~require~~requires a different version of the file will conflict. In the same manner two applications that require independent access to specific network services will conflict. ~~The A~~ common solution to these situations is to place software applications that may potentially conflict on separate compute platforms. The current state of the art, defines two architectural approaches to the migration of critical system elements from an operating system into an application context.

~~Current state of the art defines two architectural approaches to the migration of system elements from an operating system into an application context.~~ In one architectural approach, a single server operating system places critical system elements in the same process. Despite the flexibility offered, the system elements continue to represent a centralized control point.

In the other architectural approach, a multiple server operating system places critical system elements in separate processes. While offering even greater options this architecture has suffered performance and operational differences.

Summary of the Invention

An important distinction between this invention and prior art systems and architectures is the ability to allow a CSE to execute in the same context as an application. This then allows, among other things, an ability to deploy multiple instances of a CSE. In contrast, existing systems and architectures, regardless of where a service is defined to exist, that is, in kernel mode, in user mode as a single process or in user mode as multiple processes, all support the concept of a single shared service.

SUMMARY OF THE INVENTION

In accordance with a first broad aspect, ~~the of this~~ invention ~~provides,~~ a computing ~~architecture~~system is provided that has an operating system kernel having operating system critical system elements (OS CSEs) and adapted to run in kernel mode; and a shared library adapted to store replicas of at least some of the critical system elements, for use by the software applications in user mode executing in the context of the application. The critical system elements are run in a context of a software application.

The term replica used herein is meant to denote a CSE having similar attributes to, but not necessarily and preferably not an exact copy of a CSE in the operating system (OS); notwithstanding, a CSE for use in user mode, may in a less preferred embodiment be a copy of a CSE in the OS.

In accordance with the invention, a computing system for executing a plurality of software applications is provided, comprising:

an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode; and,

a shared library having critical system elements (SLCSEs) stored therein for use by the software applications in user mode wherein some of the CSEs stored in the shared library are accessible to a plurality of the software applications and form a part of at least some of the software applications by being linked thereto, and wherein an instance of a CSE provided to an application from the shared library is run in a context of said software application without being shared with other software applications and where some other applications running under the operating system can, have use of a unique instance of a corresponding critical system element for performing essentially the same function.

In accordance with the invention, there is provided, a computing system for executing a plurality of software applications comprising an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode; and,

1. a shared library having critical system elements (SLCSEs) stored therein for use by the software applications in user mode as a service distinct from that supplied in the OS kernel.

2. wherein some of the SLCSEs stored in the shared library are accessible to a plurality of the software applications and form a part of at least some of the software applications, and wherein a unique instance of a CSE provided to an application from the shared library is run in a context of said software application and where some other applications running under the operating system each have use of a unique instance of a corresponding critical system element for performing essentially the same function.

In some embodiments, the computing ~~architecture~~ system has application libraries accessible by the software applications and augmented by the shared library.

~~In some embodiments, the critical system elements are left in the operating system kernel.~~

~~In some embodiments, the critical system elements use system calls to access services in the~~ Application libraries are libraries of files required by an application in order to run. In accordance with this invention, SLCSEs are placed in shared libraries, thereby becoming application libraries, loaded when the application is loaded. A shared library or dynamic linked library (DLL) refers to an approach, wherein the term application library infers a dependency on a set of these libraries used by an application.

Typically, the critical system elements are not removed from operating system kernel.

In some embodiments, the operating system kernel has a kernel module adapted to serve as an interface between a service in the context of an application program and a device driver.

In some embodiments, the critical system elements in the context of an application program use system calls to access services in the kernel module.

In some embodiments, the kernel module is adapted to provide a notification of an interruption to a service in the context of an application program.

~~In some embodiments, the operating system kernel is adapted to provide interrupt handling capabilities to user mode CSEs.~~

In some embodiments, the interrupt handling capabilities are initialized through a system call.

In some embodiments, the kernel module comprises a handler which is installed for a specific device interrupt.

In some embodiments, the handler is called when an interrupt request is generated by a hardware device.

In some embodiments, the handler notifies the service in the context of an application through the use of an up call mechanism.

In some embodiments, function overlays are used to intercept software application accesses to operating system services.

~~intercept software application accesses to operating system services.~~

~~In some embodiments, the operating system kernel is enabled when the software application is loaded into memory.~~

In some embodiments, the critical system elements stored in the shared library are linked to the software applications as the software applications are loaded.

In some embodiments, ~~in a native form~~ the critical system elements rely on kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping.

In some embodiments, the kernel services are replicated in user mode and contained in the shared library with the critical system elements. In the preferred embodiment the kernel itself is not copied. CSEs are not taken from the kernel and copied to user mode. An instance of a service that is also provided in the kernel is created to be implemented in user mode. By way of example, the kernel may provide a TCP/IP service and in accordance with this invention, a TCP/IP service is provided in user mode as an SLCSE. The TCP/IP service in the kernel remains fully intact. The CSE is not shared as such. Each application that will use a CSE will link to a library containing the CSE independent of any other application. The intent is to provide an application with a unique instance of a CSE.

In accordance with this invention, the code shared is by all applications on the same compute platform. However, this shared code does not imply that the CSE itself is shared. This sharing of code is common practice. All applications currently share code for common services, such services include operations such as such as open a file, write to the file, read from the file, etc. Each application has its own unique data space. This indivisible data space ensures that CSEs are unique to an application or more commonly to a set of applications associated with a container, for example. Despite the fact that all applications may physically execute the same set of instructions in the same physical memory space, that is, shared

code space, the instructions cause them to use separate data areas. In this manner CSEs are not shared among applications even though the code is shared.

In some embodiments, the kernel services used by CSEs comprise memory allocation, synchronization and device access.

In some embodiments, the kernel services that are platform specific are not replicated, or provided as SLCSEs.

In some embodiments, the kernel services which are platform specific are called by a critical system element running in user mode.

In some embodiments, a user process running under the computing architecturesystem has a respective one of the software applications, the application libraries, the shared library and the critical system elements all of which are operating in user mode.

In some embodiments, the software applications are provided with respective versions of the critical system elements.

In some embodiments, the system elements which are application specific reside in user mode, while the system elements which are platform specific reside in the operating system kernel.

In some embodiments, a control code is placed in kernel mode.

In some embodiments, the kernel module is adapted to enable data exchange between the critical servicessystem elements in user mode and a device driver in kernel mode.

In some embodiments, the data exchange uses mapping of virtual memory such that data is transferred both from the critical servicessystem elements in user mode to the device driver in kernel mode and from the device driver in kernel mode to the critical servicessystem elements in user mode.

In some embodiments, the kernel module is adapted to export services for device interface.

In some embodiments, the export services comprise initialization to establish a channel between a critical system element of the critical system elements in user mode and a device.

In some embodiments, the export services comprise transfer of data from a critical system element of the critical system elements in user mode to a device managed by the operating system kernel.

In some embodiments, the export services include transfer of data from a device to a critical system element of the critical system elements in user mode.

According to a second broad aspect, the invention provides an operating system comprising the above computing architecturesystem.

According to a third broad aspect, the invention provides a computing platform comprising the above operating system and computing hardware capable of running under the operating system.

According to a fourth broad aspect, the invention provides a shared library accessible to software applications in ~~a~~-user mode, the shared library being adapted to store system elements which are

replicas of systems elements of an operating system kernel and which are critical to the software applications.

According to a fifth broad aspect, the invention provides an operating system kernel having ~~systemssystem~~ elements and adapted to run in a kernel mode and to replicate, for storing in a shared library which is accessible by software applications in user mode, system elements ~~of the system elements~~ which are critical to the software applications.

According to a sixth broad aspect, the invention provides an article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for a computing architecture. The computer readable code means in said article of manufacture has computer readable code means for running an operating system kernel having critical system elements in kernel mode; and computer readable code means for storing in a shared library replicas of at least some of the critical system elements, for use by software applications in user mode. ~~The critical system elements are run in a context of a software application.~~

The critical system elements are run in a context of a software application. Accordingly, elements critical to a software application are migrated from centralized control in an operating system into the same context as the application.

Advantageously, the invention allows specific operating system services to efficiently operate in the same context as a software application.

~~Critical~~In accordance with this invention, critical system elements normally embodied in an operating system are exported to software applications through shared libraries. The shared library services provided in an operating system are used to expose these additional system elements.

~~Brief Description of the Drawings~~

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the invention will now be described with reference to the attached drawings in which:

~~Figure~~FIG. 1 is an architectural view of the traditional monolithic prior art operating system;

~~Figure~~FIG. 2 a is an architectural view of a multi-server operating system in which some critical system elements are removed from the operating system kernel and are placed in multiple distinct processes or servers;

~~Figure~~FIG. 2 b is illustrative of a known system architecture where critical system elements execute in user mode and execute in distinct context from applications in a single application process context;

FIG. 3 is an architectural view of an embodiment of the invention;

~~Figure~~FIG. 4 is a functional view showing how critical system elements exist in the same context as an application;

~~Figure~~FIG. 5 is a block diagram showing a kernel module provided by an embodiment of the invention; and,

~~Figure~~FIG. 6 shows how interrupt handling occurs in an embodiment of the invention.

~~Detailed Description of the Preferred Embodiments~~

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Embodiments of the invention enable the replication of critical system elements normally found in an operating system kernel. These replicated CSEs are then able to run in the context of a software application. Critical system elements are replicated through the use of shared libraries. ~~Replication implies that system elements are not replaced from an operating system, rather they become separate extensions accessed through~~ A CSE is replicated by way of a kernel service being repeated in user space. This replicated CSE may differ slightly from its counterpart in the OS. Replication is achieved placing CSEs similar to those in the OS in shared libraries which provides a means of attaching or linking a CSE service to an application having access to the shared library. Therefore, a service in the kernel is substantially replicated in user mode through the use of shared libraries.

The term replication implies that system elements become separate extensions accessed through shared libraries as opposed to being replaced from an operating system. Hence, CSEs are generally not copies of a portion of the OS; they are an added additional service in the form of a CSE. Shared libraries are used as a mechanism where by an application can utilize a CSE that is part of a library. By way of example; a Linux based platform provides a TCP/IP stack in the Linux kernel. This invention provides a TCP/IP stack in the form of a CSE that is a different implementation of a TCP/IP stack, from Berkeley Software Distribution (BSD) by way of example. Applications on a Linux platform may use a BSD TPC/IP stack in the form of a CSE. The mechanism for attaching the BSD TCP/IP stack to an application is in the form of a shared library. Shared libraries as opposed to being copies from the OS to another space are a distinct implementation of the service (i.e. TCP/IP) packaged in the form of a shared library capable of executing in user mode linked to an application.

In a broad sense, the TCP/IP services in the CSE are the same as those provided in the Linux kernel. The reason two of these service may be required is to allow an application to utilize network services provided by a TCP/IP stack, that have unique configuration or settings for the application. Or the setting used by one application may conflict with the settings needed by another application. If, by way of example, a first application requires that specific network packets using a range of port numbers be passed to itself, it would need to configure route information to allow the TCP/IP stack to process these packets accordingly. On the same platform a second application is then exposed to either undesirable performance issues or security risks by allowing network packets with a broad port number range to be processed by the TCP/IP stack. In another scenario the configuration/settings established to support one application may have an adverse effect on the system as a whole.

By way of introduction, a number of terms will now be defined.

Critical System Element (CSE): Any service or part of a service, “normally” supplied by an operating system, that is critical to the operation of a software application.

A CSE is a dynamic object providing some function that is executing instructions used by applications.

BY WAY OF EXAMPLE CSEs INCLUDE:

Network services including TCP/IP, Bluetooth, ATM; or message passing protocols

File System services that offer extensions to those supplied by the OS

1. Access files that reside in different locations as though they were all in a single locality

2. Access files in a compressed, encrypted or packaged image as though they were in a local directory in a standard format

Implementation of file system optimizations for specific application behavior

Implementation of network optimizations for specific application services such as:

1. Kernel bypass where hardware supported protocol processing is provided

2. Modified protocol processing for custom hardware services

Compute platform: The combination of computer hardware and a single instance of an operating system.

User mode: The context in which applications execute.

Kernel mode: The context in which the kernel portion of an operating system executes. In conventional systems, there is a physical separation enforced by hardware between user mode and kernel mode. Application code cannot run in kernel mode.

Application Programming Interface (API): An API refers to the operating system and programming language specific functions used by applications. These are typically supplied in libraries which applications link with either when the application is created or when the application is loaded by the operating system. The interfaces are described by header files provided with an operating system distribution. In practice, system APIs are used by applications to access operating system services.

Application library: A collection of functions in an archive format that is combined with an application to export system elements.

Shared library: An application library ~~whose code space is shared among all user mode applications.~~ The code space is different than that occupied by the kernel and its associated files. The shared library files are placed in an address space that is accessible to multiple applications.

Static library: An application library whose code space is contained in a single application.

Kernel module: A set of functions that reside and execute in kernel mode as extensions to the operating system kernel. It is common in most systems to include kernel modules which provide extensions to the existing operating system kernel.

Up call mechanism: A means by which a service in kernel mode ~~callsexecutes~~ calls a function in a user mode application context. ~~It is common to provide an up call mechanism within an operating system kernel.~~

FigureFIG. 1 shows a conventional architecture where critical system elements execute in kernel mode. Critical system elements are contained in the operating system kernel. Applications access system elements through application libraries.

In order for an application of **FigureFIG.** 1 to make use of a critical system element 17 in the kernel 16, the application 12 a or 12 b makes a call to the application libraries 14. It is impractical to write applications, which handle CPU specific/operating specific issues directly. As such, what is commonly

done is to provide an application library in shared code space_z which multiple applications can access. This provides a platform independent interface where applications can access critical system elements. When the application 12 a, 12 b makes a call to a critical system element 17 through the application library 14, a system call may be used to transition from user mode to kernel mode. The application stops running as the hardware ~~enters kernel mode.~~18 enters kernel mode. The processor makes the transition to a protected/privileged mode of execution called kernel mode. Code supplied by the OS in the form of a kernel begins execution when the transition is made. The application code is not used when executing in kernel mode. The operating system kernel then provides the required functionality. It is noted that each oval 12 a, 12 b in FigureFIG. 1 represents a different context. There are two application contexts in the illustrated example and the operating system context is not shown as an oval but also has its own context. There are many examples of this architecture in the prior art including SUN Solaris™, IBM AIX™, HP-UX™ and Linux™.

FigureFIG. 2 a shows a system architecture where critical system elements 27 a, 27 b execute in user mode but in a distinct context from applications. Some critical system elements are removed from the operating system kernel. They reside in multiple distinct processes or servers~~as discussed in United States Provisional Patent Application entitled "Drag & Drop Application Management" which is incorporated herein by reference.~~. An example of the architecture described in FIG. 2 a is the GNU Hurd operating system.

The servers that export critical system elements execute in a context distinct from the operating system kernel and applications. These servers operate at a peer level with respect to other applications. Applications access system elements through application libraries. The libraries in this case communicate with multiple servers in order to access critical system elements. Thus_z in the illustrated example, there are two application contexts and two critical system element contexts. When an application ~~needs to make~~requires the use of a critical system element_z which is being run in user mode, a ~~rather convoluted~~ sequence of events must take place. Typically the application first makes a platform independent call to the application library. The application library in turn makes a call to the operating system kernel. The operating system kernel then schedules the server with the critical system element in a different user mode context. After the server completes the operation, a switch back to kernel mode is made which then responds back to the application through the application library. Due to this architecture, such implementations may result in poor performance. Ideally, an application_z which runs on the system of FigureFIG. 1, should be able to run on the system of FigureFIG. 2 a as well. However, in practice it is difficult to maintain the same characteristics and performance using such an architecture.

FIG. 2 b is illustrative of a known system architecture where critical system elements execute in user mode. The invention is contrasted with both of these architectures in that critical system elements also execute in a distinct context from applications. Some critical system elements are removed from the operating system kernel. The essential difference between the architecture described in FIG. 2 a and FIG. 2 b is the use of a single process context to contain all user mode critical system elements. An example of the architecture described in FIG. 2 b is Apple's MAC OS X™.

This invention is contrasted with all three of the prior art examples. Critical system elements as defined in the invention are not isolated in the operating system kernel ~~in as is~~ the case of a monolithic architecture (~~Figures shown in FIG. 1~~);_z also they are not removed from the context of an application_z as is

the case with a multi-server architecture (~~Figure 2~~), depicted in FIG. 2 a. Rather, they are replicated, and embodied in the context of an application.

~~Figure~~FIG. 3 shows an architectural view of the overall operation of ~~the~~this invention. Multiple user processes execute above a single instance of an operating system.

Software applications 32 a, 32 b, utilize shared libraries 34 as is done in United States U.S. Provisional Patent Application Ser. No. 60/512,103 entitled "SOFTWARE SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS" which is incorporated herein by reference. The standard libraries are augmented by an extension, which contains critical system elements, that reside in extended shared libraries 35. Extended services are similar to those that appear in the context of the operating system kernel 36.

~~Figure~~FIG. 4 ~~shows~~illustrates the functionality of an application process as it exists above the~~an~~ operating system ~~kernel all of which is that was described in FIG. 3. Applications exist and~~ run in user mode while the operating system kernel 46 itself runs in kernel mode. ~~The user~~User mode functionality includes the user applications 42, the standard application libraries, ~~and a new extended shared library provided by an~~ 44, and one or more critical system elements, 45 & 47. FIG. 4 ~~shows one~~ embodiment of the invention. The ~~where the CSE functionality is contained in two shared libraries. An extended library, 45, provides control operations while the CSE shared library, 47, provides an implementation of a critical system element.~~

The CSE library includes replicas or substantial functional equivalents or replacements of kernel functions. The term replica, shall encompass any of these meanings, and although not a preferred embodiment, may even be a copy of a CSE that is part of the OS. These functions can be directly called by the applications 42 and as such can be run in the same context as the applications 42. In preferred embodiments, the kernel functions which are included in the extended shared library and critical system element library 45,47 are also included in the operating system kernel- 46.

Furthermore, there might be different versions of a given critical system element ~~forming part of the extended shared library~~47 with different applications accessing these different versions within their respective context.

In preferred embodiments, the platform specific aspects of the critical system element are left in the operating system kernel, ~~for example certain system calls.~~ Then the critical system elements ~~which are included~~running in the extended shared library~~user mode~~ may still make use of the operating system kernel to implement these platform specific functions.

~~Figure~~FIG. 5 represents the function of the kernel module ~~{58, described in more detail below}~~. A critical system element in the context of an application program uses system calls to access services in the kernel module. The kernel module serves as an interface between a service in the application context and a device driver. Specific device interrupts are vectored to the kernel module. A service in the context of an application is notified of an interrupt by the kernel module.

~~Figure~~FIG. 6 represents interrupt handling. Interrupt handling is initialized through a system call. A handler contained in the kernel module 58 is installed for a specific device interrupt. When a hardware device generates an interrupt request ~~is generated by a hardware device~~ the handler contained in the kernel module ~~will be~~is called. The handler notifies a service in the context of an application through the use of an up call mechanism.

Function Overlays

A function overlay occurs when the implementation of a function that would normally be called, is replaced such that an extension or replacement function is called instead. The invention uses function overlays in one embodiment of the invention to intercept software application accesses to operating system services.

The overlay is accomplished by use of an ability supplied by the operating system to allow a library to be preloaded before other libraries are loaded. Such ~~an~~ ability is used to cause the loading process ~~to~~ performed by the operating system} to link the application to a library extension supplied by the invention rather than to the library that would otherwise be used. The use of this preload capability to attach a CSE to an application is believed to be novel.

The functions overlaid by the invention serve as extensions to operating system services. When a function is overlaid in this manner it enables the service defined by the API to be exported in an alternate manner than that provided by the operating system in kernel mode.

Critical System Elements

According to the invention, some system elements that are critical to the operation of a software application are replicated from kernel mode, ~~by the operating system,~~ into user mode in the same context as that of the application.

These system elements are contained in a shared library. As such they are linked to a software application as the application is loaded. ~~Figure 3 shows that an extension library is utilized~~ This is part of the operation performed when shared libraries are used. A linker/loader program is used to load and start an application. The process of starting the application includes creating a linkage to all of the required shared libraries that the application will need. The CSE is loaded and linked to the application as a part of this same process.

FIG. 3 shows that an extension library is utilized. In its native form, as it exists in the operating system kernel, a CSE uses services supplied by the operating system kernel. In order for the CSE to be migrated to user mode and operate effectively, the services that the CSE uses from the operating system kernel are replicated in user mode and contained in the shared library with the CSE itself. Services of the type referred to here include, but are not limited to, memory allocation, synchronization and device access. Preferably, as discussed above, platform specific services are not replicated, but rather are left in the operating system kernel. These will then be called by the critical system element running in user mode.

~~Figure~~ FIG. 4 shows that the invention allows for critical system elements to exist in the same context as an application. These services exported by library extensions do not replace those provided in an operating system kernel. Thus, in ~~Figure~~ FIG. 4 the user process is shown to include the application itself, the regular application library, the extended library and the critical system element all of which are operating in user mode. The operating system kernel is also shown to include critical system elements. In preferred embodiments, the critical system elements which are included in user mode are replicas of elements which are still included in the operating system kernel.

The term replication means that like services are supplied. As was described heretofore, it is not necessarily the case that duplicates of the same implementation found in the kernel are provided by a

CSE; but essentially a same functionality is provided. As discussed previously, different applications may be provided with their own versions of the critical system elements. ~~Advantageously, this can make it appear that multiple applications running on a given platform have their own operating system. However in reality, there is only one operating system with application specific components in user mode, and with non-application specific components only residing in the kernel mode. It is noted that this allows different applications running on a given platform to operate in secure separation without clashing for resources and without versioning problems of libraries. Previous attempts to address this problem have included building hardware that is capable of running multiple versions of an operating system, and building a virtual machine in software which effectively allows each service to have its own operating system, but in software. The new solution provided by this embodiment of the invention yields the same benefits of these two other solutions, but without the requirement from multiple operating systems.~~

Kernel ~~module~~Module

In some embodiments, control code is placed in kernel mode as shown in ~~FigureFIG.~~ 4. ~~FigureFIG.~~ 5 shows that a kernel module is used to augment device access and interrupt notification.

As a device interface the kernel module enables data exchange between a user mode CSE and a device driver in kernel mode. The exchange uses mapping of virtual memory such that data is transferred in both directions without a copy. Services exported for device interface typically include:

~~1. Services exported for device interface typically include:~~

~~@~~ Initialization.

2. Establish a channel between a CSE in user mode and a specific device.

3. Informs the interrupt service that this CSE requires notification.

~~@~~4. Write data.

5. Transfer data from a CSE to a device.

6. User mode virtual addresses are converted to kernel mode virtual addresses.

~~@~~7. Read data.

8. Transfer data from a device to a CSE.

9. Kernel mode data is mapped into virtual addresses in user mode.

10. During initialization, interrupt services are informed that for specific interrupts, they should call a handler in the kernel module. The kernel module handles the interrupt by making an up call to the critical system element.

Interrupts related to a device being serviced by a CSE in user mode are extended such that notification is given to the CSE in use.

As shown in ~~FigureFIG.~~ 6 a handler is installed in the path of an interrupt. The handler uses an up call mechanism to inform the affected services in user mode. A user mode service enables interrupt notification through the use of an initialization function.

The general system configuration of the present invention discloses one possible implementation of the invention.

In some embodiments, the '~~C~~'C' programming language is used but other languages can alternatively be employed.

Function overlays have been implemented through application library pre-load. A library supplied with the invention is loaded before the standard libraries, using standard services supplied by the operating system. This allows specific functions (~~APIs~~APIs) used by an application to be overlaid or intercepted by services supplied by the invention.

Access from a user mode CSE to the kernel module, for device I/~~IO~~ and registration of interrupt notification, is implemented by allowing the application to access the kernel module through standard device interfaces defined by the operating system. The kernel module is installed as a normal device driver. Once installed applications are able to open a device that corresponds to the module allowing effective communication as with any other device or file operation. Numerous modifications and variations of the present invention are possible in light of the above teachings without departing from the spirit and scope of the invention.

~~Numerous modifications and variations of the present invention are possible in light of the above teachings.~~ It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.