

Auriga's Driver Development Expertise

Overview

The development of stable, reliable, and maintainable device drivers requires specialized knowledge, experience, and tools. Auriga has acquired considerable expertise, backed by almost 20 years of working on a large number of projects involving development, maintenance and testing of a variety of device drivers. Auriga deals with board support packages (BSPs) projects so frequently, that at any given time there are multiple BSP projects being performed by Auriga engineers. Auriga also frequently deals with standalone driver development projects. Typically, the scope of a BSP development project includes development of drivers, interfaces and associated development tools. The experience gained by Auriga enables the development of a device driver for virtually any type of I/O or DMA devices in a process-oriented, predictable manner.

Project Samples

For many years, Auriga has developed drivers and BSPs both for OS and board vendors. Device drivers developed by Auriga for its customers include PnP drivers, bus drivers for specialized system buses, network adapter (NDIS) drivers, filter drivers, and many others.

Drivers for system control devices

One of the projects involved development of a driver for the Hot Swap Controller device in high-end industrial CompactPCI based systems produced by Motorola Computers Group. These systems include two redundant system controllers (single-board computers) and provide the following system control functionality via the Hot Swap Controller device:

- extended CompactPCI slot control: • powering slots up and down, placing slots in reset and taking out of reset
- chassis control: switching on and off power supplies and peripheral bays, controlling fan speed, controlling system LEDs, monitoring alarms
- I/O switchover: passing the control over CompactPCI I/O from one system controller to another without I/O disruption.

The Hot Swap Controller driver provided software support for the functionality described above. The driver has been

implemented in two variants: for Windows NT version 4 and for Windows 2000. The Windows 2000 variant conforms to the WDM driver model and provided more functionality.

For example, I/O switchover is supported only in the Windows 2000 variant of the driver, since specific Windows 2000 features (Plug-and-Play mechanism) are used to prepare the I/O devices for switchover and ensure their visibility after the switchover.

Testing and exercising Plug-and-Play features of the system

One of the projects implemented by Auriga developers involved creation of a test package for testing the Plug-and-Play related features in the Embedded version of Microsoft Windows 2000. This package covered all related mechanisms and APIs provided by the system in both kernel and user mode. It included substantial kernel-mode components, which were implemented as special software-only device drivers, conforming to WDM

In the user mode, the Expect package was used to design test scenarios and run tests on the test controller machine. The test controller (running a Windows or Linux environment) and the system under test (running Windows 2000) communicated with each other via the Remote Shell package over the TCP/IP network connection.

Bus drivers for specialized system buses

In the WDM model, drivers of a special type, called bus drivers, serve system buses. The bus driver is responsible for enumeration of devices on the bus, and for creation and maintenance of physical device objects (PDOs) for them. The system requires the bus driver to support certain functionality for these device objects; these requirements are more complicated than the requirements for a regular (functional) device driver. The project performed by Auriga developers involved development of the Windows 2000 bus driver for the advanced type of peripheral interconnect (system bus). This bus can be used to connect several PCI/CompactPCI bus trees together via the serial switched fabric, and supports native devices directly connected to the fabric. The bus driver implements linkage between PCI buses and supports native devices by maintaining physical device objects (PDOs) for them. For

these PDOs, the bus driver implements all the functionality required by the system, such as providing device identifiers, capabilities and resource requirements. For functional drivers, the bus driver exports a special interface by responding to the IRP_MN_QUERY_INTERFACE. This interface aids functional drivers in performing common operations with the fabric (for example, obtaining device configuration, registering for device events).

Filter drivers

In the WDM model, filter drivers place their device objects between physical device objects maintained by bus drivers and functional device objects maintained by functional drivers. In this way, filter drivers can amend the behavior of the bus driver by intercepting and modifying IRPs traveling between the functional driver and the bus driver. The PCI bus filter driver has been implemented as a part of software support effort for Hot Swap on the CompactPCI bus. The filter driver attaches its device objects between physical and functional device objects for CompactPCI devices and bridges. It monitors the IRP traffic for these devices and participates in resource allocation for the bridges. This allows achieving sparse resource allocation (and sparse bus number assignment in particular) in CompactPCI systems, which is important for Hot Swap support.

Another filter driver has been developed for the Plug-and-Play testing project. This Universal Filter driver can be attached as lower or upper filter to any device or device subtree and monitor and report to the user mode the traffic of various IRPs along the corresponding device stacks. This filter can have some additional logic attached to it and applied to the devices it filters: for example, it has been used on one of the systems to fix the incorrect behavior of serial ports caused by the incorrect ACPI description of the system.

Pseudo-device drivers

Pseudo-drivers do not have any physical devices associated with them and are used to augment the functionality of the kernel and provide additional services to user mode entities. In Windows 2000, pseudo-drivers still have device objects associated with them and are created as regular WDM drivers, subject to dynamic creation and destruction of their device objects by the system's request.

An example of such driver is a special test driver, developed for the Plug-And-Play testing project, which is responsible for receiving kernel-level Plug-and-Play notifications and forwarding them to the user-mode monitoring application.

Network adapter (NDIS) drivers

Drivers for network adapters in Windows NT and Windows 2000 are usually implemented as NDIS miniports, special drivers that conform to the Network Driver Interface Specification (NDIS) driver model. This driver model is layered above the basic driver model (legacy NT 4 driver model or WDM) and shields the developer from the differences between the two.

One of the projects performed by Auriga developers involved creation of an NDIS miniport that provided network-oriented view of the system PCI bus. With that approach, software on intelligent peripheral I/O controllers could interact with another peripheral controller and with the system controller over the system backplane using regular network protocols like TCP/IP. The speed of communication was in that case significantly higher than when using dedicated Ethernet.

Driver installation

A usual way to install a driver in the Windows 2000 system is to create an information (.INF) file for it. This file contains the identification attributes for devices to associate the driver with, points to the location of the driver files and describes the registry settings for the driver. However, in some cases (for example, when the driver is installed as a part of a larger software package), this is not sufficient, and more advanced installation methods should be used. There are several products that aid in the installation of software packages (Install Shield, Wyse). Also, the system provides special API (Setup API) that presents more granular and detailed approach to driver installation than using .INF files.

Auriga's developers have extensive hands-on experience with all the above mentioned techniques and have created several quite sophisticated installers combining commercial tools and system APIs.

FibreChannel Device Drivers Suite for LynxOS RTOS

The goal of the project was to provide a set of LynxOS drivers for the Interphase's x525 family of Fibre Channel adapters that would enable LynxOS to use IP and SCSI Fibre Channel interfaces. The suite was developed as a part of a bigger project targeted to providing a high-capacity SANbased storage solution.

The set of drivers was implemented to allow control of both HBA node and disks attached to the same loop, supporting simultaneous IP and SCSI exchanges. The set of Fibre Channel drivers was designed to allow for maximum scalability and extensibility.

The Fibre Channel software package included the

following drivers and service utilities:

- The Fibre Channel Master Driver: provides general purpose Fibre Channel support services and interfaces that allows attachment of Upper Level Protocol drivers and hardware drivers to the Master component.
- The Fibre Channel Network Interface: maps conventional Ethernet based IP networking on Fibre Channel exchanges.
- The Fibre Channel SCSI driver: maps SCSI transactions issued by the LynxOS kernel to Fibre Channel SCSI exchanges.
- The Fibre Channel Raw Interface: simulates a UDP based network interface but enables maximum performance since the conventional kernel TCP/IP stack is bypassed.
- The Interphase's x526 family Fibre Channel HBA driver: enables the control of the HBA hardware.
- Management and configuration utilities: enables system administrators to control the Fibre Channel drivers.

Special attention was paid to testing the Fibre Channel drivers. Various test suites were used to verify that Fibre Channel interfaces operate correctly.

The software set was accompanied with the necessary internal documentation and the user documentation.

I/O Co-processor control and management system, including a set of drivers and associated APIs for Digital UNIX

The IOCP (I/O Co-Processor) software program facilitates development of hardware device drivers and generic I/O handlers on the Intel 960 processor used as an I/O co-processor in VME Alpha computer running under DIGITAL UNIX. The project included:

- Development of the kernel features in DIGITAL UNIX to support new devices or boards (e.g. TGA board).
- Development and implementation of the drivers in DIGITAL UNIX environment for VME devices.
- Development of a generic VME pseudo-device driver to facilitate user's access to VME devices without writing specific drivers for each VME device.

Target operating systems

Linux and Embedded Linux

- Blue Cat Linux
- Hard Hat Linux
- Proprietary distributions

UNIX

- AIX
- HP-UX
- Solaris
- SunOS
- Digital UNIX/Tru64
- UNIX
- SCO UnixWare
- SCO OpenServer

Real-time and embedded OS

- VxWorks
- LynxOS
- OSE
- pSOS
- QNX

Windows

- NT 4.0
- 2000
- XP
- NT 4.0 Embedded
- XP Embedded
- CE