ADC April 2001 3/8/01 1:57 PM Page 1

ADC
Apple Developer Connection
Direct

Darwin: Mac OS X's Core OS

Beneath Mac OS X's user-friendly and attractive user interface, Aqua, and the application frameworks (Classic, Carbon and Cocoa) is Darwin: Mac OS X's core OS. Unseen by users, Darwin provides a strong yet flexible foundation with features like preemptive multitasking, protected memory and real-time support that make Mac OS X a truly modern operating system.

The focus of this article is to provide a brief overview of Darwin and its components as well as give an introduction to developing kernel extensions—modules that extend Darwin's functionality. For more in-depth information, you should read Inside Mac OS X: Kernel Environment which is available, along with other documents referred to in this article, on the Apple Developer Connection (ADC) web site in the Mac OS X Documentation section:
http://developer.apple.com/techpubs/macosx/macosx.html

ADC Programs and Mac OS X

Register today for Apple's Worldwide Developers Conference (WWDC) from May 21-25, 2001 in San Jose, California. WWDC will feature more than 100 in-depth technical sessions and hands-on labs. You'll have access to Apple engineers and technology experts to answer your software and hardware development questions.

Naturally, WWDC 2001 session topics will cover a full range of Apple development subjects: Mac OS X architecture (Darwin, Quartz, OpenGL, QuickTime, Carbon, Cocoa, Aqua), hardware, BSD UNIX, Java, WebObjects, development tools and much more.

Register now to attend the conference, network with peers and learn about all of the exciting technologies designed in Mac OS X. Pricing, detailed session information and registration information is available at:
http://www.apple.com/developer/wwdc2001/

Remember, ADC Select and Premier members receive special discounts. See you in May!

Most of the reference documents can be found in the /Developer/Documentation/Kernel directory on any Mac OS X system with the Mac OS X Developer Tools package installed.

Components of Darwin

Just like in the old Reese's Peanut Butter Cups commercials ("You've got chocolate in my peanut butter… No, you've got peanut butter on my chocolate!"), Darwin blends a mixture of mature industry standard components such as Mach and BSD with Apple-engineered components to provide Mac OS X with a stable, reliable and extensible foundation. Darwin consists of five main components: Mach, I/O Kit, File System, Networking and BSD.

Mach

At the heart of Darwin is Mach, based on Mach 3.0 from Carnegie Mellon University. Mach manages processor resources such as CPU usage and memory, handles scheduling, provides memory protection and provides a messaging-centered infrastructure to the rest of the operating system layers. Mach provides Mac OS X with protected memory, preemptive multitasking, virtual memory and real-time support.

I/O Kit

Darwin provides an object-oriented framework, I/O Kit, for the development of device drivers. I/O Kit not only facilitates the creation of drivers for Mac OS X but also provides much of the infrastructure that drivers require. It consists of three major components: families, nubs and drivers.

A family defines a collection of software abstractions that are common to all devices of a particular category. Apple provides families for protocols such as USB, SCSI and FireWire, as well as for devices such as storage, HID and frame buffers. Mac OS X developers should rely upon these provided families—not create new families.

A nub is an I/O object that represents a device or logical service. A nub may represent a bus, a disk, a disk partition, a keyboard or any number of similar entities.

A driver is an object that manages a specific piece of hardware, implementing the appropriate I/O Kit abstractions for controlling

John Signa is the Technology Manager for Mac OS X Core OS in Apple Worldwide Developer Relations. John first got involved in the Macintosh industry in 1988 writing software for Orange Micro's printing products. He later spent three years at SuperMac/Radius writing display and video drivers.

that hardware. Mac OS X provides a collection of drivers that handle standard devices such as hard drives and human input devices. If your device complies with an industry standard but has additional functionality, then you simply need to subclass the provided driver and implement just the code that handles the uniqueness of your device.

Anyone working with I/O Kit—either as an application writer or a driver developer—should read Inside Mac OS X: I/O Kit Architecture to get more background on how I/O Kit works. Device driver developers should also read Mac OS X: Writing I/O Kit Drivers. Application developers should read Inside Mac OS X: Accessing Hardware from Applications. Both documents are available on the ADC web site in the Mac OS X Documentation section.

File System

The file system component of Darwin is based on an enhanced Virtual File System (VFS) design, which provides the ability to add in new file systems and enhance those already supported, including HFS, HFS+ UFS, and ISSO 9660. VFS stacks also allow you to create and layer new capabilities, such as file-based compression or encryption onto an existing file system type.

Networking

Mac OS X also provides an extensible networking system. By implementing Network Kernel Extensions (NKEs) developers can add support for additional networking protocols as well as enhance the networking functionality already provided. Developers who need to extend Mac OS X's networking capabilities should read Inside Mac OS X: Network Kernel Extensions for more details.

BSD

Darwin wraps a customized version of BSD 4.4 around the kernel. Darwin's implementation of BSD includes many of the POSIX APIs, exporting them to user-space, and abstracts Darwin's file system and networking. Darwin's BSD also provides Mac OS X's process model, basic security policies and threading support.

For developers, the biggest advantage of Darwin's BSD implementation is that it enables you to quickly port UNIX-style applications to Mac OS X. In some cases, developers have had their UNIX applications up and running on Mac OS X in a matter of hours. Because BSD does not provide GUI APIs, you will need to create a Carbon or Cocoa application to handle the user interface. Much of this work can be done using Interface Builder, Apple's user interface design tool available on the Mac OS X Developer Tools CD.

Developing Kernel Extensions

To handle enhancements to the kernel, Mac OS X provides the ability to dynamically load pieces of code, referred to as Kernel Extensions (KEXTs), without needing to recompile the kernel. All kernel extensions are implemented as "bundles"—folders that the Finder treats as single entities. In addition to having names that end in ".kext," all kernel extensions contain a property list (plist), which is an XML text file describing the KEXT's contents and requirements. Additionally, a KEXT will usually, but not always, include a module (KMOD) that contains the binary code that is actually loaded into the kernel and run. A KEXT can also contain additional resources such as icons for the Finder.

Before you dive into developing a KEXT, you must decide if you really need to run in the kernel space. Compared to code running at the user space, kernel extensions are more difficult to write and debug. Furthermore, bugs in kernel extensions can have far more severe consequences. For example, a memory access error in a user application can, at worst, cause that application to crash yet leave the rest of the OS functional. In contrast, a memory access error in a KEXT causes a system panic, crashing the entire operating system.

When you are trying to decide if a piece of code should be a KEXT, the answer is generally no. Just because your code was a system extension in Mac OS 8 or 9, does not mean that it must neces-

Continued on page ?? ☛

Project Builder Tips & Tricks

• Regular expression searches are very powerful. If you use regular expressions with subexpressions, you can reference the subexpressions in your replace string using "\#" syntax where "#" is the index of the subexpression. For example, if you search for the regular expression "foo(.*)bar" and find an occurrence with is "fooabcdbar," a replace string of "bar\1foo" will change that match into "barabcdfoo."

• Find Options Sets can make your batch searches much faster. Try defining a set that does not search framework headers and one that only searches framework headers, and swap between them depending on what you're looking for.

• Lots of cool features are only available if you index your project. You have to index a project manually, the first time. After that, your index will be kept up to date automatically.

New Mac OS X Related Releases

The following software is available from the Download Software area of the ADC Member Site at:
http://connect.apple.com/

• **CarbonLib 1.2.5 GM SDK**
The CarbonLib 1.2.5 SDK for Mac OS is now available to all developers. This SDK provides all the files needed to begin Carbon development. CarbonLib 1.2.5 supports Mac OS 8.6 and greater.
http://developer.apple.com/technical/

• **CarbonLib 1.3d6 SDK**
The latest prerelease version of the CarbonLib 1.3 SDK for Mac OS is now available to all ADC Members.
http://developer.apple.com/technical/

Developer Documentation

The following new and updated documentation is available to help you on your way to successful Mac OS X application and peripheral development at:
http://developer.apple.com/techpubs/

- iMac Developer Note (Update)
- Inside Cocoa: Object-Oriented Programming and the Objective-C Language
- Mac OS X: An Overview for Developers - A 10-page PDF document that explains the unique combination of technologies in Mac OS X and discusses the benefits of those technologies to developers.
  http://developer.apple.com/macosx/

TN1191 - USB Software Locator

QA1008 - WaitMouseUp documentation errata
QA1001 - Detecting CD/DVD media types
QA1006 - Displaying Help
QA1005 - Open File Limits on Mac OS X
QA1004 - Displaying the device tree in Mac OS X
QA1003 - Enabling Macintosh-style Menu Bars

SAMPLECODE - QuickTime: Importers and Exporters: ElectricImageComponent
SAMPLECODE - Human Interface Toolbox: ScrollingTextUserPane
SAMPLECODE - Human Interface Toolbox: MLTEEditField
SAMPLECODE - Human Interface Toolbox: HTMLUserPane
SAMPLECODE - Human Interface Toolbox: HandyScrollingSample
SAMPLECODE - Networking: URLAccessSample
SAMPLECODE - Files: MoreFiles
SAMPLECODE - Printing: PostScript Output Filters
SAMPLECODE - Java: JNISample
SAMPLECODE - Human Interface Toolbox: CarbonCustomList
SAMPLECODE - Networking: OTLookupNameTest

Upcoming Seminars and Events

For more information on Apple developer events please visit the developer Events page at:
http://developer.apple.com/events/

Training and Seminars

Programming with Cocoa
Taught by Aaron Hillegass at the Big Nerd Ranch, Ashville, NC and Atlanta, GA. Five-day classes are taught on developing web-based and Mac OS X applications.
http://www.bignerdranch.com/when.html

Developer Related Conferences

National Association of Broadcasters (NAB) Conference, Las Vegas, NV
April 21-26
Produced annually by the National Association of Broadcasters, NAB2001 is the world's leading conference and exhibition for the converging electronic media communications industries. Apple will be exhibiting in Booth #M9131.
http://www.nab.org/conventions/nab2001/

Worldwide Developers Conference (WWDC) 2001, San Jose, CA
May 21-25
Register now for Apple's Worldwide Developers Conference 2001, which takes place in San Jose, California from May 21-25. ADC Premier members receive a free pass to the conference and ADC Select members receive discounts for early registration. For schedules and other details check out:
http://www.apple.com/developer/wwdc2001/

MacHack Conference, Dearborn, MI
June 21-23
MacHack, in its sixteenth year, remains centered around cutting edge software development. MacHack's uniqueness derives from the informal feel and the LIVE coding that occurs around-the-clock during the conference.
http://www.machack.com/

Darwin: Mac OS X's Core OS

Continued from page ??

sarily be a kernel extension in Mac OS X. To help you decide, ask yourself the following questions:

- Does your code need to take a primary interrupt? That is, does something in the hardware need to interrupt the CPU?
- Does the primary client of your code reside inside the kernel (for example, a hard drive whose primary client is the file system)?
- Do a sufficiently large number of running applications require a resource that your code provides (for example, a file-system stack)?

If you answered "no" to all of the above, then you should consider developing your code as a library or a background application rather than as a kernel extension. You might also consider using one of the user level plug-in architectures provided by Mac OS X, such as QuickTime components or Printer Modules. However, if you are writing device drivers or code to support a new volume format or networking protocol, KEXTs may be the only solution.

Fortunately, while KEXTs are more difficult to write than user-space code, several tools and procedures are available to enhance the development and debugging process. Currently you'll need to use Apple's Project Builder IDE to create KEXTs and use GDB for debugging. For a quick introduction on creating and debugging KEXTs with Project Builder GDB, you should read Inside Mac OS X:Kernel Extensions Tutorial.

Open Source

In March, 1999, Apple announced the Darwin Open Source initiative, making Apple the first major computer company to make open-source development a key part of its ongoing software strategy. Apple has released the source code to virtually all of the components of Darwin to the developer community, providing you the ability to see how Apple has implemented Mac OS X's core OS. Not only is this helpful in understanding how the OS works, but it also allows you to utilize portions of Darwin within your own products. Before doing so you should review the Apple Public Source License to understand the limitations or obligations this entails. It can be found at:
http://opensource.apple.com/apsl/

Summary

Developing kernel code is never trivial, however Darwin's flexible architecture makes it easier than ever before to write drivers or add additional file systems for Mac OS X. By implementing such a flexible architecture in Darwin, Apple has provided a foundation that delivers the reliability and performance you'd expect from a modern operating system. Moreover, releasing Darwin to the open source community ensures that it will continue to evolve as a high-quality, interoperable system built on open standards.

Did You Know?

Darwin: Document It!

You probably know that you can go to Apple's Open Source web site *(www.opensource.apple.com)* and download the binary and source code of Darwin and other Open Source projects. But did you know that you can get more than code? You can also obtain documentation as well. And you can add to this store of information.

That's because this documentation is created by developers in the Apple Open Source community for other developers. The Darwin Documentation project (also at *www.opensource.apple.com*) provides you with an assortment of tools and guides to help you compose professional-looking documents. These documents are of three types:

- HeaderDoc—Reference documentation produced by a Perl tool that parses structured commentary embedded in C and C++ header files and produces rich HTML output from that commentary.
- HOWTO documents—Conceptual and task-oriented information on specific programming topics. HOWTO documents are based on DocBook XML because from this format tools can process the document into multiple formats (HTML, PDF, etc.). The documentation project provides a template and instructions for creating HOWTO documents. You don't have to learn DocBook XML if you don't want to; you can submit the HOWTO as an HTML document and Apple will convert it to XML.
- Manpages—Traditional UNIX-style documentation of command-line tools and utilities.

"Built for Mac OS X" Badge Now Available

Tell the world your product runs on Mac OS X! The artwork, licensing requirements and guidelines for use of the new "Built for Mac OS X" badge are now available on the ADC Software Licensing web site. Please note that this badge cannot be used for products that launch the Classic environment.

http://developer.apple.com/mkt/swl/agreements.html#macosx