Introduction

Important: This document is no longer being updated. For the latest information about Apple SDKs, visit the documentation website.

OS X supports a number of application environments, each with its own runtime rules, conventions, and file formats. In OS X, kernel extensions, command-line tools, applications, frameworks, and libraries (shared and static) are implemented using Mach-O (Mach object) files.

The OS X runtime architecture dictates how object files are laid out in the filesystem and how programs communicate with the kernel. The object file format used in OS X is Mach-O .

A Mach-O file has the following regions of data (the complete format is described in *OS X ABI Mach-O File Format Reference*):

- **Header:** Specifies the target architecture of the file, such as PPC, PPC64, IA-32, or x86-64.
- **Load commands:** Specify the logical structure of the file and the layout of the file in virtual memory.
- **Raw segment data:** Contains raw data for the segments defined in the load commands.

The following list describes other runtime environments supported in OS X:

- *Classic* is a Mac app that runs Mac OS 9 within its address space and provides bridging services that allow OS X to interact with Mac OS 9 applications. Both classic 68K applications and PowerPC Code Fragment Manager (CFM) applications can run under Mac OS 9 in Classic. (Mac OS 9 does not support the 68K variant of Code Fragment Manager, so you cannot run CFM-68K applications in OS X.)
- *LaunchCFMApp* is a command-line tool that runs programs created for the PowerPC Code Fragment Manager. The file format used by such programs is called *Preferred Executable Format (PEF)*. Carbon provides bridging for Code Fragment Manager applications that allows them to link to Mach-O-based code, but—for ease of debugging if for no other reason—it's generally a good idea to use Mach-O for Carbon applications.
- The HotSpot *Java virtual machine* is a Mac app that executes Java bytecode applications and applets.
- The *OS X kernel* supports kernel extensions (KEXTs), which are static Mach-O executable files that are loaded directly into the address space of the kernel. Because errant code can write directly to memory used by the kernel, kernel extensions have the potential to crash the operating system. You should generally avoid implementing functionality as kernel extensions if possible.

The Code Fragment Manager is documented in *Mac OS Runtime Architectures*, available from the Apple Developer Connection website.

This document discusses how you use the Mach-O file format. It describes what types of programs you can build, how programs are loaded and executed, how you can change the way programs are loaded and executed, how to load code at runtime, and how to load and link code at runtime. If you create or load bundles, shared libraries, or frameworks, you'll probably want to read and understand everything in this document.

Who Should Read This Document

If you write development tools for OS X, you need to understand the information presented in this document.

This document is also useful for developers of shared libraries and frameworks, and for developers of applications that need to load code at runtime.

Organization of This Document

This document contains the following articles:

- Building Mach-O Files describes how Mac apps are built and describes the types of programs you can develop.
- Executing Mach-O Files provides an overview of the OS X dynamic loading process.
- Loading Code at Runtime describes how to use shared libraries and frameworks and how to load plug-ins at runtime.
- Indirect Addressing explains how a Mach-O file refers to symbols defined in another Mach-O file.
- Position-Independent Code discusses the method by which the dynamic linker loads a region of code at a non-fixed virtual memory address.
- x86-64 Code Model describes differences in the OS X x86-64 user-space code model from the System V x86-64 code model.

This document also contains a revision history and an index.

See Also

You can access full reference documentation for the standard command-line development tools using the `man` tool on the command line, or by choosing Open Man Page from the Xcode Help menu.

This document provides information on the Mach-O runtime architecture. It does not address the following:

- Descriptions of the data structures that make up a Mach-O file. You can find this information in *OS X ABI Mach-O File Format Reference*.
- If you are loading code at runtime but cannot or do not wish to use the `CFBundle` opaque type or the `NSBundle` class, you should refer to *OS X ABI Dynamic Loader Reference*.
- The GCC C++ application binary interface—the specification of C++ class member layout, function/method name mangling, and related C++ issues. This information is documented for GCC 3.0 and later at http://www.codesourcery.com/cxx-abi/abi.html.
- The GCC Objective-C data structures and dynamic runtime functions. For this information, see *The Objective-C Programming Language*.
- The runtime environment of the OS X kernel, Darwin. See *Mac Technology Overview* for more information.

Source code from the Darwin project can be downloaded from http://developer.apple.com/darwin/.

You might also find the following books useful in conjunction with this document:

- *Mac OS Runtime Architectures*, Apple Computer, Inc. Available at http://developer.apple.com/tools/mpw-tools/books.html. Documents the classic 68K segment loader architecture, as well as the Code Fragment Manager Preferred Executable executable format used with classic PowerPC applications and with many Carbon applications.
- *Linkers and Loaders*, John R. Levine, Morgan Kaufmann, 2000, ISBN 1-55860-496-0. Describes the workings and operation of standard linkers from the earliest program loaders to the present dynamic

link editors. Among the contents of this book are discussions of the classic BSD `a.out` format, the Executable and Linking Format (ELF) preferred by many current operating systems, the IBM System/360 linker output format, and the Microsoft Portable Executable (PE) format.

- *System V Application Binary Interface AMD64 Architecture Processor Supplement*. Found at http://www.x86-64.org/documentation, this document describes the System V x86-64 environment, on which the OS X x86-64 environment is based.