Macintosh

Operating System

**ITP51-OPERATING SYSTEM**

TABLE OF CONTENTS

**Chapter 01 Introduction and History 01**

Chapter Objectives 2

Mac OS History 3

The Mac OS Apple File System (APFS) 4

The Mac OS Kernel 5

**Chapter 2 Process Management 7**

Process Management in Mac OS 8

Mac OS Activity Monitor 9

Key Components of MacOS process Management 10

Process Properties 11

Process Environment Terminal on Mac 12

Processes and Threads 13

Mac OS X Features 14

Processor Modes & Privileged Instructions 15

User-Level Threads 16

Kernel - Level Threads 17

**Chapter 3 Scheduling 19**

Thread Scheduling 20

Time Sharing OS Threads 21

Real Time OS Threads 22

Real Time Scheduling 23

Kernel Synchronization 24

Symmetric Multiprocessing 25

**Chapter 4 Memory Management 27**

Temporary Memory 28

Virutal Memory 29

System Heap 30

The System Global Variables 31

Organization of Memory in an Application Partition 32

The Application Stack 33

The Application Heap 34

**Chapter 5 Storage Management 36**

Utility and Disk Structure in Mac OS 37

Catalina 38

Hard Disk Drive 39

**Chapter 6 I/O System 41**

I/O Kit 42

Mac OS User Space 43

Device Driver 44

I/O Registry 45

System and Kernel Extension in Mac OS 46

Kernel Extension 47

**Chapter 7 File System 49**

Apple File System for Mac OS 50

Optimized for Modern Storage Technology 51

Futureproofing 52

APFS Features 53

HFS+ 54

01

CHAPTER 1

**INTRODUCTION AND HISTORY**

The macOS operating system is an advanced, preemptive multitasking client operating system designed for microprocessors implementing various instruction set architectures, including Intel x86, Apple Silicon (ARM-based), and ARM64. Its counterpart on the server side, macOS Server, aligns with the macOS client but exclusively supports the 64-bit Apple Silicon architecture. macOS continues the evolution of Apple's operating systems, stemming from the Mac OS X lineage, and ultimately based on the NeXTSTEP and UNIX foundations. Mac OS is special software made to manage the simplest tasks a computer does, like making new icons on your screen, getting on WiFi, and handling different programs. It stands out because it's tailor-made for the Apple machines it operates on. With Mac OS becoming standard on Apple's computers, you're guaranteed a smooth pairing with all of Apple's gear, such as iCloud and your iPhone. It provides a flawless tech experience that's hard to find anywhere else. Renowned for its user-friendly design, this OS offers a smoother user journey compared to its rivals. It upgrades many existing features and has inspired competitors to adopt its innovative elements. The Mac OS brings a host of pre-installed apps covering video and sound editing, and a comprehensive bundle of data management and scheduling tools. Plus, it effortlessly integrates FaceTime and iMessage, allowing seamless communication right from your desktop.

2

**CHAPTER OBJECTIVES**

• Explore the principles of Mac Operating System design and the specific components of the system.

• Provide a detailed discussion of the Mac OS file system.

• Illuminate the array of networking protocols seamlessly integrated into macOS, providing a comprehensive understanding of the system's connectivity capabilities and network interaction.

• Describe the interface available in Mac OS to system and application programmers.

• Unveil the crucial algorithms implemented within macOS efficiency, responsiveness, and overall performance.

3

**MacOS History**

The first genuine operating system of Apple Inc. is the Mac operating system. The founders of the company were Steve Jobs, Steve Wozniak, and Ronald Wayne back in 1976. There are earlier operating systems have been made before the MacOS that have been associated with Apple Inc. and Apple computers. The first official reference to MacOS came with the public beta release version known as “Kodiak”. Kodiak was released in the Autumn of 2000 after Apple implemented much of the user feedback they had received from previous iterations, but the public beta license expired in the Spring of 2001 as

version 10.0, also called “Cheetah,” was released. Mac OS was designed to replace the old version operating system that was used in the earlier Macintosh computers first developed in 1980. Behind the Mac OS, the software architecture is Andy Hertz Field who created much of the original kernel and desktop applications. Furthermore, Susan Kare created the operating system icons and Steve Capps and Bruce Horn developed the famous Finder and System Utilities applications. The launch of Cheetah kicked off Apple's Mac OS 10 series. It began Apple's pattern of big cat names for their operating systems, like Lion and Tiger. This lasted over ten years until they introduced Mavericks as version 10.9. After that, Apple shifted to naming their operating systems after places, starting with Yosemite, and followed by Sierra and Mojave. Big Sur broke the mold with its version number jumping to 11, leaving behind the 10.x sequence. The latest Mac OS, Monterey, was announced in June 2021 and is set to debut in November, bringing cool enhancements to existing features. There are several easy-to-understand facts about Mac OS that you should know. Each version of Mac OS gets a cool name inspired by mighty wild cats and beautiful places like Lion, Puma, Big Sur, and Yosemite. Nearly every year, you can count on Mac OS to come out with an updated version. Mac OS is made just for Apple’s computers and devices, guaranteeing a smooth

experience with MacBooks, iPhones, and iCloud. The basic structure of Mac OS started with Unix, it’s got its unique file system called APFS, which stands for Apple File System. Mac OS is famous for being super easy to use, thanks to its neat and friendly design. Even though Apple and Microsoft started up at about the same time, Mac OS hit the scene about fifteen years ago after the world saw the first Windows OS

4

**The Mac OS Apple File System (APFS)**

Apple File System was introduced in June 2016 at the Apple Developer Conference (WWDC) as a replacement for HFS+, which had been in use since 1998. With the release of iOS on March 27, 2017, APFS was made available for 64-bit iOS devices. In September 2018, Apple released a partial specification for APFS that allows for read-only access to Apple file systems on unencrypted, non-Fusion storage devices. File systems can be utilized on devices with little or high storage capacities. It uses 64-bit inode numbers for more secure storage. To optimize space management and performance, APFS code, like HFS+ code, employs the TRIM command. The way APFS calculates available data on iOS and macOS can enhance read and write rates and potentially expand capacity on iOS devices. APFS uses the GPT partitioning mechanism. The GPT scheme contains one or more APFS containers (partition type GUID is 7C3457EF-0000-11AA-AA11-00306543ECAC). Each container contains one or more APFS volumes, all of which share the container's space, and each volume may serve as an APFS volume. APFS volume groups were introduced in macOS Catalina (macOS 10.15). This is a collection of volumes that appears in the Finder as a single volume. There are APFS farm links between hard links and soft links that connect volumes. The System Volume role (commonly referred to as "Macintosh HD") is made read-only in macOS Catalina. In macOS Big Sur (macOS 11), it is renamed Signed System Volume (SSV), and only volume snapshots are mounted. The data volume role (often titled Macintosh HD - Data) is used as an overlay or shadow for the system volume, where both the system volume and data volume are part of the same volume group and are viewed as one in the Finder. It is evident. The Apple File System (APFS) is the default file system for Mac computers. It offers rapid directory sizing, space sharing, snapshots, robust encryption, and enhanced file system basics. Although APFS is designed for the Flash/SSD storage found in newer Mac computers, it can also be used on older systems that have external direct-attached storage and conventional hard disk drives (HDD). APFS is supported for both bootable and data volumes on macOS 10.13 or higher. Disk space inside a container (partition) is allotted by APFS dynamically. When an APFS container has more than one volume, the free space inside the container is shared and allotted to the separate volumes as needed. You can set reserve and quota sizes for every volume if you'd like. Since each volume only takes up a portion of the whole container, the available space is equal to the container's overall size less the sum of the volumes' respective sizes.

5

**The Mac OS Kernel**

The macOS operating system, formerly known as Mac OS X and OS X kernel, is its basic component. It oversees managing the system's resources, such as the CPU, memory, and input/output devices, as well as responsibilities such as process scheduling and inter-process communication. XNU is a hybrid kernel with features of both monolithic kernels and microkernels, attempting to make the best use of both technologies, such as the message-passing ability of microkernels allowing greater modularity and larger portions of the OS to benefit from memory protection while maintaining the speed of monolithic kernels for some critical tasks. The macOS kernel is known as XNU (X is Not Unix) and is derived from the Unix-based operating system BSD (Berkeley Software Distribution). XNU is an abbreviation for "X is Not Unix," highlighting its hybrid character and released as free open-source software as part of the Darwin OS. The macOS kernel combines pieces from the Mach microkernel as well as FreeBSD operating system components. Mach is a microkernel created at Carnegie Mellon University that provides a basic and modular base, while FreeBSD provides the networking stack, file system support, and other features.

**The Mac OS**

Apple published the initial version of its Mac OS X operating system, notable for its UNIX design, on March 24, 2001. OS X is now macOS and has long been noted for its ease of use, aesthetically pleasing interface, innovative technologies, applications, security, and accessibility features. The operating system has been integrated into Apple's hardware, as well as iOS devices, from the Finder to Spotlight, the Dock to iCloud, so users can effortlessly work and navigate. It also works on Apple Silicon as of Big Sur. OS X/macOS has matured into a brand identity for both Apple and its Mac lineup, inheriting part of its design from Apple's subsequent debut of iOS, its mobile operating system.

Cheetah

Cheetah, Apple's first major OS X release, was a drastic departure from the prior Mac OS 9, Apple's Classic Operating System. Mac OS X 10.0, announced at the 2000 Macworld Expo in San Francisco, introduced the new Aqua User Interface, which included brushed metal title bars, brightly colored buttons on windows, and photo-realistic icons. OS X Cheetah also introduced the dock, which still distinguishes modern Macs. Mail, Address Book, and TextEdit were also included as new programs.

Puma

At the 2001 Macworld Expo in New York, Apple debuted OS X Puma. It offered speedier menu navigation and log-in, as well as adjustable features including a moveable Dock and new system menus with volume, battery life, and AirPort access settings on the menu bar. iTunes arrived with OS X 10.1 and included CD burning, DVD playback, and compatibility with third-party digital cameras and MP3 players.

Jaguar

While earlier versions of OS X had jungle cat names, they were not exploited for marketing purposes until OS X Jaguar. It was announced at the 2002 Macworld Expo, and it included an upgraded Mail software that moved spam to a garbage folder. iChat became Apple's primary instant messaging app, with built-in support for AOL Instant Messenger. Along with a revamped Finder with enhanced search, OS X 10.2 offered universal access for the first time, allowing users to enlarge screen contents and activate programs with vocal dictation.

Panther

At WWDC 2003, Apple debuted OS X Panther. It included Exposé, a handy feature that allowed users to examine all open windows at once, windows of a current program, or files on the desktop. iChat AV allowed users to chat using audio and video in addition to text, and Safari replaced Internet Explorer for Mac as the Mac's default web browser after Microsoft discontinued support for it. File Vault, Apple's simpler application for controlling system fonts; Xcode, Apple's developer tool for building Mac apps; and a revised Finder with a sidebar for easy access to disk devices, networks, and folders were also introduced in OS X 10.3.

Tiger

WWDC 2004 saw the debut of OS X Tiger. Spotlight, a universal search client, was included, allowing users to search their entire system for files, emails, contacts, photos, calendars, and apps from the menu bar. Dashboard widgets included weather, airline information, stock tickers, and more. Safari included RSS support, and iChat supported up to four video conference participants or ten voice conference participants. In addition, OS X 10.4 introduced Automator, an application that automated tasks and workflows such as renaming large groups of files or resizing dozens of images; and VoiceOver, an accessibility interface that provided magnification options, keyboard control, and spoken English descriptions of what was happening on screen. After Apple's switch to Intel x86 processors, OS X Tiger was the first to support the Apple-Intel architecture, and it became the longest-running version of Mac OS X before Leopard's introduction 30 months later.

Leopard

At WWDC 2006, Apple introduced OS X Leopard, which was described as the "largest update of OS X." It featured a modernized look with a three-dimensional, reflective Dock, a semitransparent menu bar, larger drop shadows for active windows, and new high-resolution icons. Stacks, a grouping feature that displays files in folders on the Dock in a "fan" or "grid" style; an updated Finder that incorporates the Cover Flow visual navigation interface first seen in iTunes, iPhone, and iPod touch; Quick Look, which allows a "preview" of items by hitting the space bar; Spaces, a way to group application windows on more than one virtual desktop; Time Machine, an automatic backup utility that allows users to restore deleted files; OS X Leopard unofficially supported multi-touch gestures with the advent of the aluminum unibody MacBook and MacBook Air with the multi-touch trackpad.

Snow Leopard

OS X Snow Leopard was announced at WWDC 2009, and users were able to purchase it for the first time for a relatively reasonable $29. All of Apple's applications were rewritten in 64-bit code for this update, as well as Grand Central Dispatch, a new way for software developers to write applications that take advantage of multicore processors, and OpenCL, a C-based open standard that allows developers to tap into the power of the GPU for tasks other than graphics. OS X 10.6 also expanded on the preceding OS (thus the name) with additional innovations such as faster disk ejecting, faster Time Machine backup, faster boot and shut down times, and faster OS installation. OS X Snow Leopard was substantially smaller than OS X Leopard, freeing up to 7 GB of space; it was also Apple's first Intel-only operating system. After the introduction of the Mac App Store in the Snow Leopard 10.6.6 update, it became the second-longest running Apple OS after Tiger, and the last to require a physical disc.

Lion

At a special event named "Back to the Mac," Apple unveiled OS X Lion, which officially enabled multi-touch gestures and new motions and reactions such as rubber-band scrolling, page and image magnification, and full-screen swiping. For the first time, programs could be full-screen, and the introduction of Mission Control merged Exposé, Dashboard, Spaces, and full-screen apps to provide users with a single spot to see and navigate everything operating on their Mac. Many iOS developments were ported to OS X 10.7, like as an easily navigable display of installed applications -- Launchpad -- and new thinner, gray scrollbars that disappeared when not in use. Support for the Mac App Store, which was introduced in OS 10.6.6, was also included in OS X Lion. Other innovations included Resume, which allowed apps to be reopened after they had been closed, and Auto Save.

Mountain Lion

Apple debuted OS X Mountain Lion at WWDC 2012, and since then has deleted the "Mac" prefix from all OS X mentions on its website. iCloud synced mail, calendars, contacts, reminders, documents, notes, and other data between Macs, iPads, iPhones, and iPod touches. Notification Center alerted users to new emails, messages, software updates, or calendar alerts; Dictation converted words into text; and a new Sharing button allowed the sharing of photos, videos, files, and links with Mail, Messages, and AirDrop. OS X 10.8 integrated Facebook and Twitter, allowing users to set up and manage their accounts directly from System Preferences. AirPlay Mirroring allowed for simple sharing from a Mac to an Apple TV, and Game Center introduced iOS games to the Mac. Finally, Gatekeeper helped safeguard customers from downloading and installing dangerous software onto their Macs by only allowing them to install apps from the Mac App Store and other identified developers. Users could get this upgrade for a lower cost of $19.99.

Mavericks

Apple unveiled OS X Mavericks during WWDC 2013. It replaced its large cat naming convention with locations in Apple's home state of California, therefore Mavericks was called after a prominent surfing area. OS X 10.9 introduced iBooks from iPad, Maps from iOS, which allowed users to send directions from their Mac directly to their iPhone, and Tags, a new way to organize and find files by labeling them with colors or specific naming conventions; iCloud Keychain stores, encrypts, and automatically enters passwords; the Finder can support tabbed browsing for increased productivity, as well as support for working on multiple displays with menu bars and dock; and Notification Center allows users to For the new MacBook Pro with Retina Display, icons were redrawn twice their original size. For the first time, Mac users could get the most recent Mac OS for free.

Yosemite

For the first time, Mac users could get the latest Mac OS for free.

Apple announced OS X Yosemite at. To complement the design revamp of iOS 7, OS X Yosemite ditched the skeuomorphic interface it has employed for over a decade. The update includes a flat graphic design, blurring translucence effects, a two-dimensional Dock that iterated on the one used in Tiger, updated icons, light and dark color schemes, and the first time the default system typeface was replaced from Lucida Grande to Helvetica Neue. With Continuity, Macs may now receive and make calls from an iPhone on the same WiFi network. Macs could also send and receive SMS text messages, and iCloud synced all iMessages and text messages between Macs and iOS devices. Handoff was featured in OS X 10.10, and it allowed users to write an email or browse Safari while switching from a Mac to an iPhone or iPad and vice versa. The new Photos app replaced iPhoto and consolidated photos from iCloud or an iOS device into a single location. Mail allowed users to markup attachments right in the program, while Mail Drop allowed users to transfer huge files (up to 5GB) without worrying about email client constraints. Finally, iCloud Drive allows users to store all types of files in one location and access them from any device.

EL Capitan

WWDC 2015 saw the unveiling of OS X El Capitan. It improved the design and usability of OS X Yosemite while also improving performance and security. Split View, which placed two full-screen apps side by side; a streamlined Mission Control that made it easier to see all open windows in one place; a smarter Spotlight that delivered results for weather, sports, stock, web, video, and transit directions and can deliver information using natural language; a refreshed Maps app that included public transit information for some U.S. cities. Metal, a new graphics core technology that provided games and apps with nearly direct access to the Mac's graphics processor for improved performance; in-app multi-touch gesture support; and system-wide performance improvements for increased responsiveness and efficiency when using apps. San Francisco, a new font used on iOS devices running iOS 9 and Apple watches running watchOS, also replaced Helvetica Neue, which launched with Yosemite.

Sierra

The rebranded macOS Sierra debuted at WWDC 2016 and was released in full form on Sept. 20 after a public beta program throughout the summer. Sierra introduced a slew of new features, the most notable of which was the addition of Siri to the Mac desktop. Users may use Siri via the upper right-hand corner of the menu bar, between the Notification Center and Spotlight menu icons, a Dock icon, the menu bar icon, or a hot key. The voice-activated assistant may help users find files, gain rapid access to information like the weather, or do searches, with search results that could be saved and pinned to the Notifications area. Other features in Sierra included a Universal Clipboard, which allowed text to be copied from one Apple device to another; iCloud Desktop and Documents, which used the cloud to sync data between Macs; Auto Unlock, which enabled the Apple Watch to unlock a macOS account; and Apple Pay on the web. Photos now has a Memories feature that creates curated collections of users' favorite photos and videos. Sierra also included more than 60 security updates.

High Sierra

MacOS High Sierra was introduced at WWDC 2017 as an iterative update to macOS Sierra; its moniker is a tribute to California's Sierra Nevada mountains. This version included the Apple File System for more efficient data management and a new industry standard for 4K video High-Efficiency Video Coding, generally known as HEVC or H.265. Metal 2 improved graphics and natural language processing, and for the first time, developers could create interactive virtual reality experiences on a Mac. A new Photos app made it easier to find and organize photos, and it included extensive editing tools. Safari became faster, and new capabilities such as Intelligent Tracking Prevention were added to help advertisers erase cross-site tracking data. Safari also stopped those obnoxious auto-playing videos with audio for a more peaceful browsing experience. Other additions included enhanced Siri and Spotlight, as well as fast iCloud Message syncing.

Mojave

Apple revealed macOS Mojave during WWDC 2018. Its biggest feature was a new Dark Mode that rendered the entire UI dark gray. (A new live wallpaper could also adapt to the time of day based on a user's location.) The new Stacks feature organized congested desktops by automatically stacking comparable files into groups, while Gallery View — akin to Cover Flow — allowed users to visually skim through files. Quick Look allowed users to rotate, crop, and mark up PDFs and cut video recordings without opening native applications. The iOS News, Stocks, Voice Memos, and home applications were ported to the desktop; Group FaceTime allowed up to 32 users to participate in a video conference at the same time; and the Mac App Store was redesigned. Safari's Intelligent Tracking Prevention security upgrades prevented user tracking without authorization. The screenshot tool included new on-screen tools for annotating photographs as well as the ability to record video. With Continuity Camera, customers could shoot a photo or scan a document with their iPhone or iPad and have it displayed instantly on their Mac. Integrated markup tools for Finder, Quick Look, and Screenshots make it simple to add comments.

Catallina

Apple launched Apple Arcade, a $5-per-month game subscription service, with macOS 10 (Catalina). Apple also released desktop versions of numerous entertainment apps, including Apple Music, Apple Podcasts, and Apple TV, to assist customers in finding the material they want to listen to or watch. Catalina included a new feature called Sidecar for corporate Mac users, which is aimed to extend the Mac desktop to the iPad. When Sidecar is activated, the tablet transforms into an input device (via Apple Pencil). In addition, with new Voice Control features, Apple tapped Siri to allow customers to run their Macs solely through voice commands. Apple announced Mac Catalyst for developers, which allows them to create macOS equivalents of iPad apps. For the security-conscious, Apple added a read-only system volume to help prevent data loss; Gatekeeper to limit the possibility of malware and dangerous app downloads; and Activation Lock, which allows a user to erase a Mac if it is lost or stolen. macOS Mojave does not support 2010-12 Mac Pro models and requires 4GB of RAM.

Big Sur

The main news with macOS 11 (main Sur) was less about the operating system and more about the hardware that supports it: Macs powered by Apple Silicon chips. At WWDC '20, Apple announced plans to switch all its devices to its CPUs within two years, and Big Sur was the first version of macOS to operate on the new chips. Big Sur received a UI redesign that included curved windows, redesigned Dock icons, a new Control Center (accessible from the menu bar), a redesigned Notification Center, improvements to make the Safari browser faster and more customizable, more "personality" for Messages, and an overhauled Time Machine backup system. Apple emphasized that Big Sur would feature "Rosetta 2" technology, which would let software created for Intel chips to operate on new gear based on Apple silicon. Big Sur has discontinued support for some older Macs (mostly those introduced before 2013).

Monterey

macOS 12 (Monterey) includes some enhancements aimed at making it easier for Mac users to connect to the internet while remaining focused on what they're doing on the desktop. Safari is redesigned to relocate tabs higher in the browser window, effectively integrating tabs, the toolbar, and the search field, and allowing users to create and share tab groups. Monterey also receives Shortcuts, which aids users in automating chores and is incorporated throughout the operating system. FaceTime gets a slew of new features, including Voice Isolation (which makes it easier to hear people on a video call) and Wide Spectrum (which allows callers to hear everything, not just a person's voice) Share Play, which allows multiple people in different locations to collaborate on work or even watch the same movie and chat about it in real-time; blurred backgrounds; and a new grid view. With the introduction of Universal Control, users will be able to work with a single mouse and keyboard and smoothly switch between a Mac and an iPad, with no additional setup necessary. Apple also showed out iCloud+ capabilities aimed at improving security and privacy, as well as Live Text, which uses machine learning to detect information in images such as phone numbers and addresses that can be utilized in other apps. Monterey no longer supports older Macs manufactured before late 2014 (but it does operate on the 2013 Mac Pro).

6

**References**

History Computer Staff July 31, (2023) <https://history-computer.com/complete-history-of-mac-os/?fbclid=IwAR3g1pt7PNAtfZIcRPEwtQogWzwE8MSQN3WANEgXfVUWJh8uKTG-vXNbE8U>

<https://support.apple.com/en-ph/guide/disk-utility/dsku19ed921c/mac#:~:text=Apple%20File%20System%20(APFS)&text=APFS%20allocates%20disk%20space%20within,the%20individual%20volumes%20as%20needed.>

<https://academic-accelerator.com/encyclopedia/apple-file-system>

Daniel A. Steffen May 23, 2023 <https://github.com/apple/darwin-xnu>

<https://en.wikipedia.org/wiki/XNU>

Sean Moreau and Ken Mingis Computer World Jun 14, 2023, 12:36 PM PDT <https://www.computerworld.com/article/3692528/evolution-of-macos-and-mac-os-x.html>

7

CHAPTER 2

**PROCESS MANAGEMENT**

**Process Management**

Processes are the fundamental units of work in macOS. They represent running programs, background services, and system tasks. Every application you launch, every open window, and even invisible background tasks like indexing files or checking for updates are all processes. Process management is the set of operations involved in creating, scheduling, and terminating processes in an operating system. A process management operating system can be viewed as an instance of a program that is currently running on a computer system. Process management is an important feature of every modern operating system because it allows numerous programs to operate concurrently and efficiently share system resources. The process management operating system subsystem oversees allocating system resources such as CPU time, memory, and input/output devices to running processes. Process management software is also in charge of scheduling process execution to enhance system throughput while minimizing response times. If one or more processes absorb all available system resources, a computer system might become unusable or even crash if adequate process management is not implemented. As a result, process management is an essential component of any modern operating system, and its design and implementation have a considerable impact on the system's overall performance and usability.

8

**Process Management in macOS**

The macOS, Apple's operating system for Macintosh computers, has several built-in utilities that improve user experience and system management. Among these, the Activity Monitor stands out as an essential tool for monitoring and regulating Mac operations. As users investigate the complexities of their system's performance, the Activity Monitor becomes an invaluable resource, providing insights into CPU and memory utilization, process status, and priority. The Activity Monitor is used by macOS to manage processes. Furthermore, the Activity Monitor displays information about the programs running on a system, such as CPU and memory consumption, status, and priority. Lastly, Activity Monitor can also be used to end or restart programs, modify their priority, and set their affinity.

9

**Mac OS Activity Monitor**

Processes are apps that are now running on Mac. Apps, system apps utilized by macOS, and invisible background processes are all examples of processes. Use Activity Monitor to learn more about these processes, such as how much memory and CPU time they consume. The Activity Monitor, at its core, provides a complete overview of the processes that are actively executing on a Mac. Users can view critical information such as process names, unique process identifiers (PIDs), CPU and memory use, and overall process status. The user-friendly interface displays real-time data, allowing users to quickly identify resource-intensive apps and comprehend the system's resource distribution.

A screenshot of a computer

Description automatically generated

View Process Activity

Perform any of the following actions in the Activity Monitor software on your Mac:

• Select the process, then double-click it or click the Info button in the Activity Monitor window (or use the Touch Bar) to get information about it.

• Sorting procedures: To sort the list, click a column header.

• Click the arrow in the selected column heading to reverse the order of the items in the column.

• Click CPU in the Activity Monitor window (or use the Touch Bar) to view general information about all processes. The number of open processes and threads is displayed at the bottom of the window.

• Enter the name of a process or program in the search area to find it.

A screenshot of a touch bar

Description automatically generated

Show more Columns.

In the Activity Monitor window, you can specify which columns to display.

Choose View > Columns in the Activity Monitor software on your Mac, then select the columns you want to see (visible columns have a checkbox).

Group processes for easier viewing

Select one of the following from the View menu in the Activity Monitor software on your Mac:

• All Processes: Displays a list of all the processes that are now executing on your Mac.

• All Processes, hierarchically: Displays processes that are children of other processes, allowing you to view their parent/child relationship.

• My Processes: Displays processes that are associated with your user account.

• System Processes: Displays the processes that are owned by macOS.

• Other User Processes: Displays processes that are not owned by root or the currently logged-in user.

• Active Processes: Displays processes that are now operating and are not sleeping.

• Running Processes That Are Sleeping: Displays running processes that are sleeping.

• GPU Processes: Displays the ongoing processes that are owned by the computer's GPU.

• Windowed Processes: Displays processes that can create a window. Apps are typically used for these processes.

• Selected Processes: Displays only the processes that you chose in the Activity Monitor window.

• Applications in the last 12 hours: Display only the apps that have been executing processes in the previous 12 hours.

• Processes by GPU: Displays the running GPU processes organized by GPU.

10

**Key Component of MacOS process management**

• **Kernel:** The kernel is the fundamental building block of a computer's operating system (OS). It is the core that offers fundamental services to all other elements of the operating system. It serves as the primary interface between the operating system and the underlying computer hardware, assisting with activities like as process and memory management, file systems, device control, and networking.

• **Launchpad:** Launchpad is the most user-friendly method to open and manage programs on your Mac. Consider your Mac's Launchpad to be similar to the iPhone's App Library or Home Screen. It contains all of the apps you've downloaded from the Mac App Store or elsewhere. All of your apps may be found and opened using the Launchpad. You may also reorganize the Launchpad so that your most-used apps appear on the first page. You may create several folders to keep your programs nicely arranged, much as on an iPhone.

• **Dock:** Provides quick access to frequently used applications and displays running processes as icons. The dock is the icon bar that shows at the bottom of your screen. The dock allows you to quickly launch and manage applications. To launch an application, simply click on its icon in the Dock. To close an application, click and hold the pointer over its icon until a context menu appears. Quitting the application is one of the alternatives on that context menu.

• **Activity Monitor:** Activity Monitoring is an essential tool for monitoring and regulating Mac operations. It displays information about the programs running on a system, such as CPU, and memory consumption, status, and priority. Lastly, Activity Monitor provides a complete overview of the processes that are actively executing on a Mac.

11

**Process properties**

Each process has a set of properties that define its state and behavior, including:

• **Process ID (PID):** A process ID is a one-of-a-kind, positive number that identifies a process. The process ID can be used to guide signals between processes because it is a unique identifier. More information regarding signals can be found at Signals. The tmpnam function also makes use of the process ID; this function returns a temporary file name based on this unique identification.

• **Name:** The name of the application or executable associated with the process.

• **Priority:** A relative value that determines how much CPU time the process gets compared to others.

• **Memory usage:** Memory Usage is the quantity of RAM (Random Access Memory) allocated to your account by your hosting provider. Every operation done by your website (for example, saving data, loading files, and so on) consumes a specific amount of RAM. The amount of RAM the process is currently using. Memory usage is an important component in server performance. A website must have enough RAM to run as quickly and smoothly as feasible. Higher memory consumption indicates that a larger proportion of the available RAM is being actively used. Lower memory consumption, on the other hand, will often correlate with a faster site speed.

• **CPU usage:** CPU (Central Processing Unit) utilization is an important measure to keep track of on a Mac. It mostly assists you in keeping track of how many jobs the CPU is being requested to complete at once, which might impair computer performance. It's a good idea to monitor CPU utilization on Mac on a regular basis, not just when the computer is running slowly, because the system can be pushed beyond its capacity without an obvious drop in performance. The percentage of CPU time the process is currently using.

• State: Whether the process is running, waiting, sleeping, or stopped.

12

**Process Environment Terminal on Mac**

The shell stores information in environment variables, such as the current user's name, the host computer's name, and the default paths to any commands. Environment variables are inherited by all commands run in the shell's context, and some commands rely on them. You can use environment variables to influence the behavior of a command without having to modify the command itself. You can, for example, use an environment variable to instruct a command to print debug information to the console.

Use the proper shell command to associate a variable name with a value to set the value of an environment variable. To set the variable PATH to /bin:/sbin:/user/bin:/user/bin:/system/Library/, for example, type the following command in a Terminal window:

export PATH=/bin:/sbin:/user/bin:/user/bin:/system/Library/

Enter: to see all environment variables.

% env

When you run an app from a shell, it inherits a large portion of the shell's environment, including exported environment variables. This type of inheritance can be a great technique to dynamically configure the program. Your program, for example, can check for the presence (or value) of an environment variable and adjust its behavior accordingly. For exporting environment variables, different shells offer different semantics. See the man page for your selected shell. Although child processes of a shell inherit that shell's environment, shells are independent execution contexts that do not share environment information. Variables set in one Terminal window are not set in the others. Variables set in a Terminal window are no longer available once you close it. If you want a variable's value to be persistent between sessions and Terminal windows, you must set it in a shell starting script. See the "Invocation" section of the zsh man page for details on altering your zsh shell startup script to maintain variables and other settings across several sessions.

13

**Processes and Threads**

Mac OS X Panther is a UNIX-based operating system with an easy-to-use interface.

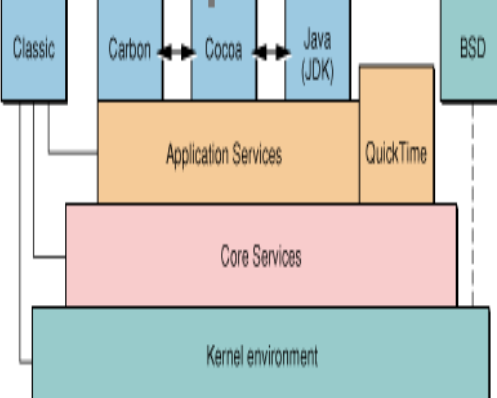
Aqua is a user interface. The Mac OS X Panther operating system has received both technical and user feedback. Since the release of Mac OS X in 2001, its features have evolved. The current UNIX-based core operating system provides advantages such as

Macintosh computing now has protected memory and proactive multitasking. Mac OS X Panther also offers a visually appealing user interface. Translucence and drop shadows are two examples of effects.

14

**Mac OS X features**

The architecture of Mac OS X is based on system software layering, in which one layer is dependent on and interacts with the one behind it. Mac OS X has four distinct layers of system software (listed in descending order of dependability).



Mac OS X Architecture

Application Environments

A diagram of a software company

Description automatically generatedincludes the frameworks, libraries, and services required for the runtime execution of programs built with that API. The current application (or execution) environments in Mac OS X are Carbon, Cocoa, Java, Classic, and BSD Commands.

Mac OS X Architecture

Application Services

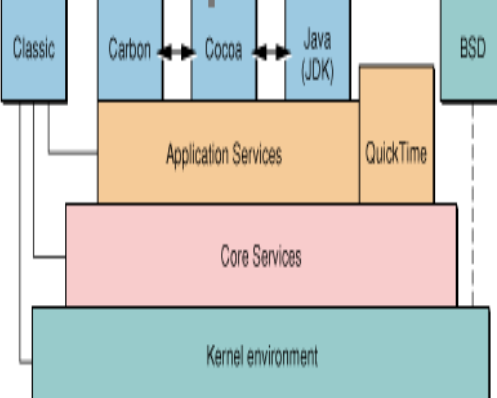
System services that are available to all application contexts and have an impact on the graphical user interface are included. Quartz, QuickDraw, and OpenGL, as well as system managers, are included.

Core Services

System services that have no effect on the graphical user interface are included. It is made up of the essential Foundation, Open Transport, and certain critical Carbon components.

Kernel Environment

The base layer of Mac OS X is included. The kernel environment provides device driver development tools (the I/O Kit) as well as loadable kernel extensions such as Network Kernel Extensions (NKEs).



Process in Mac OS X

A process is essentially a piece of software that is operating. The execution of a process must take place in a logical order. When a program enters memory and becomes a process. It is divided into four sections: stack, heap, text, and data.

A screen shot of a computer

Description automatically generated

15

**Processor Modes & Privileged Instructions**

Supervisor Mode - Supervisor mode, also known as privileged mode, is a computer system mode in which all instructions, including privileged instructions, can be performed by the processor. Some of these privileged instructions include interrupt instructions, input-output management, and so on. This mode is used by the operating system and has full access to all system components. It also allows privileged instructions to be executed and privileged registers to be accessed. Supervisor mode serves as an important barrier between applications and system hardware, as well as access to various peripherals and memory management hardware. Supervisor mode can both create and change memory address spaces. It can also access other operations' memory address spaces. Using the supervisor mode, you can enable or disable certain interrupts. It also helps with the loading of the processor status. The supervisor mode has access to the operating system's numerous data structures.

User Mode - The processor mode that prevents privileged instructions from being executed and privileged registers from being accessed. Any such effort will result in a privilege violation exception.

Process State

SIDL: The process has been partially created.

SRUN: The procedure is executable.

SSLEEP: The event is pending the procedure.

SSTOP: The process is terminated (by signal or parent process).

SZOMB: The process has been largely terminated. (Waiting for status from the parent process)

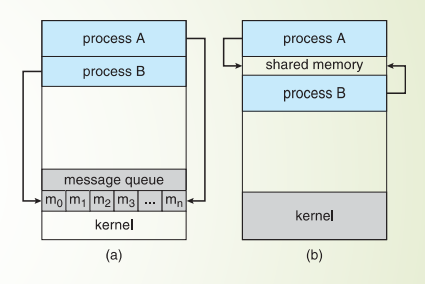
Interprocess Communication in Mac X

• Cooperating processes require some interposes communication (IPC) mechanisms for rapid information transmission.

• Two models of IPC

•  Shared memory

•  Message passing



Grand Central Dispatch (GCD)

Grand Central Dispatch (GCD or libdispatch) is a technology developed by Apple Inc. to optimize application support for multi-core processor systems and other symmetric multiprocessing systems. It is a task parallelism implementation based on the thread pool pattern. The underlying goal is to move thread pool management away from the developer and closer to the operating system. The developer injects "work packages" into the pool while completely unaware of the pool's architecture. This model simplifies, portability, and performance. GCD was first accessible with Mac OS X 10.6, and it is also compatible with iOS 4 and later. The term "Grand Central Dispatch" is an allusion to Grand Central Terminal. The source code for the library that provides the implementation of GCD's services, libdispatch, was released by Apple under the Apache License on September 10, 2009. It has been ported to FreeBSD 8.1+, MidnightBSD 0.3+, Linux, and Solaris. Attempts in 2011 to make libdispatch work on Windows were not merged into upstream. Since about 2017, the old libdispatch repository hosted by Nick Hutchinson has been deprecated in favor of a Swift core library version published in June 2016. The new version supports more systems, most notably Windows. Apple Inc. developed technologies. Boost application support for multi-core processor platforms and other symmetric multiprocessing systems. It is a thread pool-based task parallelism implementation. The basic concept is to move thread pool management away from the developer and closer to the operating system.

There are two examples of how to use Grand Central Dispatch found in Ars Technical Snow Leopard review by John Syracuse

A computer screen shot of a program

Description automatically generated

Thread Management in Mac OS X

In OS X or iOS, each process (application) is made up of one or more threads.

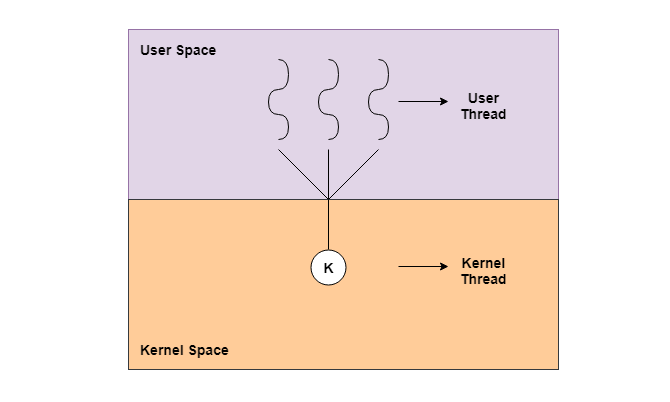
Each application begins with a single thread. The kernel schedules each thread's execution stack and execution time separately. A thread can interact with other threads and processes. conduct I/O operations, and whatever else you may require. In a single application, all threads share the same virtual memory space and have the same access permissions as the process. Because they share the same process space Threading has a significant cost in terms of memory usage and performance for your software (and the system). Each thread requires memory allocation in both the kernel memory space and the memory space of your program.

A screenshot of a computer

Description automatically generated

User Threads and Kernel Threads in Mac OS X

A thread is a lightweight process that a scheduler can handle independently. It boosts application performance by utilizing parallelism. A thread shares information with its peers such as data segments, code segment files, and so on, while it also has its own registers, stack, counter, and so on. Threads are classified into two types: user-level threads and kernel-level threads. This is seen in the diagram below.



16

**User – Level Threads**

User-level threads are implemented by users, and the kernel is unaware of their existence. It treats them as though they are single-threaded processes. Threads at the user level are substantially smaller and faster than threads at the kernel level. A program counter (PC), stack, registers, and a short process control block are used to represent them. Furthermore, there is no kernel involvement in user-level thread synchronization.

17

**Kernel -Level Threads**

Kernel-level threads are handled directly by the operating system, and thread management is managed by the kernel. The kernel manages both the process's context information and the process threads. As a result, kernel-level threads run slower than user-level threads. In kernel-level threads, several threads of the same process can be scheduled on different CPUs. Kernel routines can be multithreaded as well. If a kernel-level thread is stalled, the kernel can schedule another thread from the same process. A mode switch to kernel mode is required to transfer control from one thread to another in a process. Kernel-level threads are slower to create as well as manage as compared to user-level threads.

18

**References**

Akshat Dhawan February 27, 2023 <https://datatrained.com/post/process-management-os/#:~:text=What%20is%20process%20management%20os,%2C%20and%20I%2FO%20devices.>

<https://support.apple.com/en-ph/guide/activity-monitor/actmntr1001/mac>

Stephen J. Bigelow and Jessica Lulka <https://www.techtarget.com/searchdatacenter/definition/kernel>

Sergio Velasquez March 26, 2022<https://www.makeuseof.com/introducing-launchpad-mac/>

<https://etc.usf.edu/techease/mac/files-sharing/what-is-the-dock-and-how-do-i-use-it/>

<https://www.ibm.com/docs/en/ztpf/2019?topic=process-id>

Natalie Polly March 01, 2023 <https://setapp.com/how-to/check-cpu-usage-on-mac#:~:text=CPU%20(Central%20Processing%20Unit)%20usage,the%20performance%20of%20the%20computer.>

<https://www.dreamhost.com/glossary/hosting/memory-usage/#:~:text=Memory%20Usage%20is%20the%20amount,a%20certain%20amount%20of%20RAM.>

<https://support.apple.com/en-ph/guide/terminal/apd382cc5fa-4f58-4449-b20a-41c53c006f8f/mac>

<http://www.hasanbalik.com/LectureNotes/OpSys/Assignments/Mac%20OS%20X%20Process%20and%20Thread%20%20Management.pdf>

Alex Onsman November 04, 2023 <https://www.tutorialspoint.com/user-level-threads-and-kernel-level-threads#:~:text=The%20user%2Dlevel%20threads%20are,faster%20than%20kernel%20level%20threads>.

Ricky Barnes June 22, 2023 <https://www.tutorialspoint.com/Supervisor-Mode-Privileged-Mode>

19

CHAPTER 3

**SCHEDULING**

Scheduling refers to scheduling or planning the specific time at which certain events will occur. It may also refer to elements of an event occurring. It also entails organizing and prioritizing a series of actions in such a way that the desired goal is met on time. CPU scheduling in operating systems allows one process to use the CPU while the other processes are put on hold or remain in the waiting state. This hold or waiting state is applied when one or more system resources, such as I/O, are unavailable. Thus, the goal of CPU scheduling in operating systems is to produce a more efficient, speedier, and equitable system. CPU scheduling is a procedure that permits one process to use the CPU while another process's execution is on hold (in a waiting state).

The fundamental thread support is provided by Mac OS X's kernel environment, notably Mach. Mach keeps track of its threads' register states and schedules them ahead of time in relation to one another. Multitasking can be cooperative or preemptive in nature. Cooperative multitasking was not very sophisticated on classic Mac OS. In cooperative CPU scheduling, the operating system mandates that each job voluntarily relinquish control so that other tasks can execute, resulting in unimportant yet CPU-intensive processes. Intensive background events may consume so much of a processor's time that more critical foreground processes become slow and unresponsive. Preemptive multitasking, on the other hand, permits an external authority to delegate execution time to the available tasks. Mach in Mac OS X enables preemptive multitasking, which allows it to process many tasks at the same time.

To change the structure of the address space or to refer to a resource other than the address space, the thread must execute a specific trap instruction that causes the kernel to perform actions on the thread's behalf or send a message to some agent on the thread's behalf. These traps, in general, modify resources related with the thread's task.

20

**Thread Scheduling**

Mac OS employs a hybrid scheduling strategy that incorporates elements of both time-sharing and priority-based scheduling. The XNU (X is Not Unix) kernel, which is the kernel used in macOS, manages thread scheduling. Here are some crucial points of Mac OS thread scheduling.

Multilevel Queue Scheduler - It is possible that processes in the ready queue will be separated into multiple classes, each with its own scheduling requirements. A typical distinction is between a front (interactive) process and a background (batch) process. These two classes have distinct scheduling requirements. Multilevel Queue Scheduling is employed in this case. Multiple Queue Scheduling, or MLQ scheduling, is a strategy that organizes operations into multiple queues based on their priority levels. Each queue represents a different priority level, with higher-priority processes residing in higher-priority queues and lower-priority processes resting in lower-priority queues. Priorities are assigned to processes based on their nature, characteristics, and relevance. For example, interactive operations, such as user input/output, may be given more priority than batch processes, such as file backups. Preemption is a critical aspect of MLQ scheduling; it allows a higher priority process to interrupt the execution of a lower priority process, giving the CPU to the higher priority task. This ensures that key processes are completed on time. Furthermore, MLQ scheduling allows for the use of several scheduling algorithms for each queue, adjusting the selection based on the individual needs of the processes within that queue. Round Robin scheduling, for example, is appropriate for interactive operations, but First Come First Serve scheduling is appropriate for batch processes. A feedback mechanism can be used to improve the adaptability of MLQ scheduling. This method modifies a process's priority based on its past behavior. For example, if an interactive process has been waiting in a lower-priority queue for a long time, its priority may be increased to ensure timely execution. MLQ scheduling excels at allocating CPU time efficiently by prioritizing higher-priority operations while allowing lower-priority processes to operate during idle CPU periods. The system also ensures fairness by distributing CPU time fairly among different types of operations, taking into consideration their individual priorities and requirements. Furthermore, MLQ scheduling is highly flexible, allowing it to be tailored to the specific needs of various process types, making it a versatile and adaptive scheduling strategy.

21

**Time Sharing OS Threads** - The time-sharing operating system permits concurrent program execution by quick switching, providing each process the same amount of time to execute. Switching method/function is provided in this operating system. This switching is extremely fast, allowing users to run their programs as if they were the only ones operating, unaware that the system is being shared. The switching mechanism in time-sharing operating systems is notable for its rapidity. The system switches between processes quickly, allocating tiny time slices or time quanta to each one. This quick switching gives users the impression that their software is the only one operating on the system, oblivious to the fact that CPU time is shared across numerous processes. The switching mechanism's principal job is to control the allocation of CPU time to various programs in a way that assures fairness and responsiveness. Each process is given a short time period in which to carry out its instructions. When this time slice expires, the operating system swiftly moves on to the next process in the queue, allowing it to use the CPU for the time allotted. This cycle is repeated indefinitely, giving the appearance of concurrent execution. The time-sharing strategy is especially useful when numerous users or programs need to access the system at the same time. Time-sharing operating systems promote equitable resource use and prevent any single program from monopolizing system resources by allocating a fair share of CPU time to each process. Furthermore, time-sharing operating systems help to improve overall system efficiency. The quick switching between activities allows for optimal CPU utilization, reducing idle time and increasing system throughput. Users can engage with their programs in near real-time, which promotes a dynamic computing environment. Time-sharing has evolved over time, and modern operating systems continue to use variations of this strategy to facilitate multitasking and improve user experience. Time-sharing operating systems are widely used in personal computer environments as well as large-scale servers when several users or applications require concurrent access to computing resources.

22

**Real Time OS Threads** - Real-time operating systems (RTOS) are critical in computing environments where the emphasis is on job completion rather than reaching nominal functionality. RTOS is specifically built to handle the tight timing requirements of real-time applications, as opposed to typical operating systems, which prioritize general-purpose computing and provide services to a variety of applications. Tasks in real-time systems must not only generate precise results, but they must also do so within predetermined timeframes. This temporal element is vital for applications requiring precise timing, such as industrial automation, control systems, aeronautical applications, medical equipment, and others. The real-time operating system guarantees that these tasks are completed within the time limits provided, avoiding the system from failing due to missed deadlines. The capacity of RTOS to deliver deterministic behavior, which means that the system's response time is predictable and consistent, is an important feature. Determinism is essential in real-time applications because timing discrepancies might have serious effects. This predictability is achieved by reducing the variability in task execution durations and providing tools for prioritizing and scheduling tasks based on deadlines. To support real-time applications, RTOS provides a number of capabilities and techniques. Task scheduling is one such feature, in which activities are prioritized and done based on their urgency. Priority-based scheduling guarantees that higher-priority jobs come first, allowing vital procedures to take precedence. Furthermore, RTOS provides mechanisms for inter-task communication and synchronization, allowing for smooth collaboration between jobs to achieve overall system goals. Another essential feature of real-time operating systems is their ability to efficiently manage interrupts. Interrupts are occurrences that require immediate attention, and the RTOS is meant to respond to them in a timely and predictable manner. This guarantees that crucial events, such as sensor inputs or external signals, are processed quickly and without interfering with overall system performance. Real-time operating systems are crucial for applications that require time-sensitive activities. RTOS are distinguished from general-purpose operating systems by their emphasis on fulfilling deadlines and offering deterministic behavior. Its characteristics, such as priority-based scheduling, inter-task communication, effective interrupt handling, and support for real-time clocks, all contribute to the essential reliability and predictability in real-time applications. The relevance of real-time operating systems is projected to expand as technology advances, influencing a greater number of industries and applications.

23

**Real-Time Scheduling**

Mach offers a customizable framework for thread scheduling policies. Both the multilevel feedback queue scheduling technique and the round-robin (RR) scheduling algorithm are supported by Mac OS X. The multilevel feedback queue scheduling technique divides the ready queue into many queues and allows a process to switch between them. Each run queue in the multilevel feedback queue scheduling algorithm has different priorities that are handled differently. The priority of a multilevel feedback queue scheduling thread is raised and decreased to balance its resource consumption with other threads. Round-robin threads execute for a specific time quantum (time slice) before being pushed to the back of a queue of threads of equal priority. Setting the quantum of a round robin thread to infinity effectively causes the thread to run-till-block within its priority.

Mac OS X features 128 internal priority levels that range from 0 (lowest priority) to 127 (highest priority). They are classified into many major bands based on their characteristics, as shown in Figure 1-2. The numbers 0 through 51 correspond to what is provided via the standard BSD interface. The priority is set to 31 by default. Priorities 52 through 63 are of high importance. Regular threads 64 through 79 have the highest priority. Threads in kernel mode are represented by the numbers 80 through 95. Finally, 96 through 127 belong to real-time threads, which are treated differently by the scheduler than regular threads.

Thread priority Band

|  |  |
| --- | --- |
| **Priority Band** | **Characteristics** |
| Normal (0-51) | Normal application thread priorities |
| System high priority (52-79) | Threads whose priority has been raised above normal threads |
| Kernel mode only (80-95) | Reserved for threads created inside the kernel that need to run at a higher priority than all user-space threads (I/O Kit workloops, for example) |
| Real-time threads (96-127) | Threads whose priority is based on getting a well-defined fraction of total clock cycles, regardless of other activity (in an audio player application, for example). |

Threads can change priority levels for a variety of reasons. This movement, however, occurs within a certain band.

Threads designated as real-time priority are handled differently by the scheduler. A real-time thread informs the scheduler that it must run for A cycles out of the next B. To stay up, it might need to run for 3000 of the following 7000 clock cycles. It also indicates to the scheduler if the cycles must be consecutive. If a real-time thread regularly exceeds its time quantum without blocking, it will be penalized (and may be demoted to standard thread priority).

Threads that are heavily compute-bound are given lower priority in order to aid decrease response time for interactive jobs and prevent high-priority compute-bound threads from monopolizing the system and starving. A continual stream of higher-priority processes can prevent a low-priority process from ever receiving CPU time. Even at a lower priority, the compute-bound threads run frequently because the higher-priority I/O-bound threads only accomplish a brief amount of processing, then block on I/O again, allowing the compute-bound threads to execute.

The Mach scheduler gives priorities to incoming processes, so that the processes with higher priorities are treated first, and these mechanisms run continuously. This means that threads' priorities are frequently shifting dependent on their behavior and the behavior of other threads in the system. When numerous processes share the same queue, the scheduler employs the Round-Robin scheduling technique.

24

**Kernel Synchronization**

Apple's operating system, macOS, has a sophisticated architecture based on Unix ideas. The XNU (X is Not Unix) kernel, a unique hybrid kernel that combines parts from the Mach microkernel and the FreeBSD kernel, is at the heart of macOS. This technological fusion is critical to the operating system's robustness and versatility. The XNU kernel, a critical component of macOS, manages system resources, facilitates input/output activities, and establishes communication between hardware and software components. Its architecture finds a precise compromise between microkernel and monolithic kernel architectures, resulting in a performance-modularity fusion. This novel approach exemplifies Apple's ambition to developing an operating system that is both user-friendly and technologically powerful. The Mach microkernel, which forms a basic layer within the XNU kernel, is responsible for a major chunk of macOS's underpinnings. The Mach microkernel, which is well-known for its multitasking skills and efficient resource management, contributes critical services like as task scheduling, inter-process communication, and virtual memory management. MacOS inherits these capabilities by embedding the Mach microkernel into its architecture, establishing the groundwork for a responsive and scalable operating system. Components drawn from FreeBSD, a Unix-like operating system based on the Berkeley Software Distribution (BSD), supplement the Mach microkernel. FreeBSD provides critical components to macOS, including as networking components, file systems, and a suite of Unix utilities. This integration not only improves macOS's general functioning, but also aligns it with Unix standards, promoting compatibility and interoperability with a wide range of software applications. The operating system's Unix-based core is a crucial strength, providing it with a robust, standardized structure. This foundation supports multi-user functionality while complying to industry standards, resulting in a secure and stable environment. Developers working on macOS benefit from the Unix-based architecture by having access to familiar tools and programming interfaces, which speeds up the development process and allows for easy software porting. In conclusion, the architecture of macOS demonstrates Apple's commitment to technological innovation and user-centric design. The XNU kernel, which has its roots in the Mach microkernel and FreeBSD, forms the operating system's strong and versatile core. The Unix-based foundation guarantees stability, security, and compatibility, while open-source contributions demonstrate Apple's commitment to community-driven development. This complex mix of technologies not only defines the user experience on macOS, but also adds to the broader landscape of software development, making macOS a compelling platform for both consumers and developers.

Kernel synchronization is an important feature of operating system design because it ensures that several threads or processes can access shared resources safely without triggering data corruption or race conditions. To coordinate thread execution and guarantee data consistency, synchronization techniques are provided at the kernel level.

Here are some of the most frequent kernel synchronization techniques utilized in macOS.

Locks - Spinlocks, mutexes, and read-write locks are the three primary types of locks in OS X (and in general in Mach). Each of these has different applications and issues. Many other types of locks, such as spin-sleep locks, are not implemented in OS X, and some of them may be helpful to implement for performance comparison purposes.

Spinlocks - A spinlock is the most basic sort of lock. In a system that includes a test-and-set instruction or its equivalent. Until the lock is available, it just "spins" in a tight loop that keeps checking the lock until the thread's time quantum expires and the next thread starts to execute. A spinlock is very wasteful of CPU time because the entire time quantum for the first thread must be completed before the next thread can execute and (possibly) release the lock. It should be used only in places where a mutex cannot be used, such as a hardware exception handler or low-level interrupt handler. In OS X, there are three types of spinlocks: lck\_spin\_t (which supersedes simple\_lock\_t), usimple\_lock\_t, and hw\_lock\_t. You are highly advised not to use hw\_lock\_t; it is simply included for completeness. Only lck\_spin\_t is accessible via kernel extensions.

Mutex (Mutual Exclusion) - A mutex, mutex lock, or sleep lock is similar to a spinlock in that it places itself on a queue of threads waiting for the lock, then returns the remainder of its time quantum. It does not run again until the thread that holds the lock wakes it up (or, in some user space variants, until an asynchronous signal comes). For the most part, mutexes outperform spinlocks. They are less efficient in multiprocessing systems with a short predicted lock-holding period. Spin/sleep locks may be a preferable alternative if the average time is low but occasionally long. Although spin/sleep locks are not supported in the kernel of OS X, they can be simply constructed on top of existing locking primitives. If the use of such locks improves the efficiency of your code, you should probably look for ways to restructure it, such as utilizing more than one lock or switching to read-write locks, depending on the nature of the code in question. More details can be found under Spin/Sleep Locks. Mutexes can only be used in places where blocking is permitted because they are based on blocking. Mutexes, as a result, cannot be used in the context of interrupt handlers. Interrupt handlers are not permitted to block because interrupts are disabled for the length of an interrupt handler. If an interrupt handler blocked, the scheduler would be unable to receive timer interrupts, preventing any other thread from running, resulting in deadlock.

Semaphores- Semaphores and locks are similar, except that with semaphores, many threads can do the same task at the same time. Semaphores are widely employed to safeguard several indistinguishable resources. A semaphore, for example, could be used to prevent a queue from overflowing its bounds. Rather than binary semaphores which are effectively locks, OS X employs classic counting semaphores. Mach semaphores follow Mesa semantics, which means that when a thread is awakened by a semaphore becoming available, it is not instantly executed. This raises the possibility of starvation in multiprocessor systems with low overall load because other threads may continue to deplete the semaphore before the newly awakened thread has a chance to run. This is something to keep in mind while writing programs that use semaphores. Semaphores can be utilized anywhere that mutexes are possible. This prevents them from being used in interrupt handlers or inside the context of the scheduler, and severely discourages their use in the VM system. The public API for semaphores is divided into two files: the MIG-generated task.h file (found in your build output directory and included with #include mach/task.h>) and osfmk/mach/semaphore.h (also included with #include mach/semaphore.h>).

25

**Symmetric Multiprocessing**

SMP is the processing of computer code by several processors that share a common operating system and memory subsystem. The processors in SMP share memory and the I/O bus. All processors are managed by a single copy of the operating system. Apple did not fully commit to total SMP for the first version of Mac OS X since the company did not have any MP hardware at the time. With the implementation of SMP in Mac OS X Panther, Apple has been able to market multi-processor systems for use as graphics workstations, high-power servers, and more. Mac OS X features cutting-edge multi-processor capability since it was designed from the ground up to take advantage of many processors. Mac OS X automatically uses both processors, so all of your apps benefit from the increased performance provided by the second processor. As needed, Mac OS X assigns application tasks to processors.

Memory coherence and shared memory are the two most difficult difficulties for SMP designs. When a CPU reads or writes to system memory, the values are stored in an on-chip, high-speed cache. Once in cache, the CPU no longer needs to read memory from the system bus, enhancing performance and lowering the strain on the system bus's limited resources.

Each CPU in a multiprocessing system has its own private copy of small sections of system memory due to the way cache memory operates. Over time, these copies may differ from one another. Because the most recent value for a cached memory location is saved within the CPU's cache memory rather than in system memory, each CPU renders the main system memory incoherent by caching memory writes. To resolve this issue, the CPU monitors memory addresses placed on the system bus by other devices. When a CPU detects a cached memory address on the system bus, it writes the values of those memory addresses from its cache to system memory prior to the bus cycle's completion.

26

**References**

<https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/scheduler/scheduler.html>

<https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/>

<https://www.educative.io/answers/what-is-multilevel-feedback-queue-scheduling>

<https://en.wikipedia.org/wiki/XNU>

<https://www.cse.iitk.ac.in/users/deba/teaching/2017/cs698z/resources/ksync_notes.pdf>

<https://www.geeksforgeeks.org/multilevel-queue-mlq-cpu-scheduling/>

<https://www.geeksforgeeks.org/difference-between-time-sharing-os-and-real-time-os/>

27

CHAPTER 4

**MEMORY MANAGEMENT**

**Memory Management**

The available RAM on a Macintosh computer is used by the operating system, programs, and other software components such as device drivers and system extensions. This section discusses both the operating system's overall memory organization and the memory partition allotted to your application when it is launched. This section also includes a rough discussion of three memory-related topics:

• temporary memory

• virtual memory

• 24- and 32-bit addressing

28

**Temporary Memory**

Temporary memory, often known as volatile memory, is a type of data storage that only retains information for a limited time. In contrast, permanent storage, such as a hard drive, retains data even when the power is switched off. Volatile memory, such as that found in a computer's RAM, is often used to store data that must be accessed rapidly. This sort of memory is quick and simple to access, but it is also quite expensive. As a result, it is typically employed to store small amounts of data.

29

**Virtual Memory**

Virtual memory is a memory management approach that allows secondary memory to be used as if it were primary memory. Virtual memory is a typical approach used in the operating system (OS) of a computer. Virtual memory enables a computer to compensate for physical memory shortages by temporarily shifting data from random access memory (RAM) to disk storage. Mapping memory chunks to disk files allows a computer to treat secondary memory as if it were main memory. Most personal computers (PCs) nowadays have at least 8 GB (gigabytes) of RAM. However, this is not always enough to execute multiple programs at the same time. This is where virtual memory enters the picture. Virtual memory frees up RAM by transferring data that hasn't been utilized in a while to a storage device such as a hard disk or solid-state drive (SSD). Virtual memory is essential for increasing system performance, multitasking, and running huge programs. However, users should not rely too heavily on virtual memory because it is significantly slower than RAM. If the operating system has to swap data between virtual memory and RAM too frequently, the machine will begin to slow down, a process known as thrashing. Virtual memory was created at a time when actual memory, commonly known as RAM, was prohibitively expensive. Because computers have a limited quantity of RAM, memory will eventually run out if numerous programs are running at the same time. A virtual memory system emulates RAM by employing a part of the hard drive. A system can use virtual memory to load larger or numerous programs at the same time, allowing each one to operate as if it had more space without needing to buy more RAM.

24-bit addressing

Computers that support 24-bit addressing may locate memory addresses that are 24-bits (3 bytes) wide, allowing for a theoretical maximum of 16 gigabytes of memory, though logic board layouts may impose further constraints.

The Macintosh 128K's Motorola 68000 CPU has a 32-bit core but only supported 24-bit addressing. Early compact Macs only linked 22 of the 24 address lines, limiting them to 4 MB of RAM. The Macintosh Portable and PowerBook 100 could use the whole 24-bit address space, making them the only 68000-based Macs that could use more than 4MB of RAM. Although the Motorola 68020 and 68030 computers featured 32-bit address lines, their ROMs were still intended for 24-bit addressing and were not "32-bit clean." Such computers were limited to 8 MB of RAM in 24-bit mode. Access to up to 15 MB of virtual memory or Connectix Maxima (minus 1MB for each attached NuBus card) would be possible.

The Macintosh IIci was the first computer to come with 32-bit addressing. To add such support, Connectix Optima or Mode 32 could be used to patch the ROMs of earlier 68030-based systems. To handle 32-bit addressing and virtual memory, 68020-based computers must also include a 68851 paged memory management unit. Applications that were not "32-bit clean" in the 24-bit era would crash in a 32-bit machine. To allow such 24-bit applications to operate, the Memory control panel in System 7 to 7.5.5 contained a toggle switch. Quadra AV and Power Macintosh systems, as well as Mac OS 7.6, do not support 24-bit addressing.

32-bit addressing

Computers that support 32-bit addressing can locate memory addresses that are 32 bits (4 bytes) wide, allowing for a theoretical maximum of 4 gigabytes of memory, though logic board and CPU configurations may impose additional constraints. System 7 was the first Macintosh operating system to support 32-bit addressing, while the Macintosh IIci was the first Apple computer with "32-bit clean" ROMs to support it. Connectix created Optima and MODE32 to retrospectively add such compatibility to the ROMs of older Macintosh II models, allowing them to exceed the 8-to-14 megabyte limit of classic Mac OS's previous 24-bit addressing space. Because applications from the 24-bit era that were not "32-bit clean" would fail in a 32-bit system, System 7 until 7.5.5 included a toggle switch to enable 24-bit support in the Memory control panel. Quadra AV and Power Macintosh systems, as well as Mac OS 7.6, do not support 24-bit addressing and only run in 32-bit mode. Under Mac OS X, 32-bit applications may be allotted up to 3GB of RAM, however Macintosh models in the 2000s quickly found this restriction to be a bottleneck. The inclusion of 64-bit CPUs in future Macintosh models, such as the PowerPC G5 and Intel Core 2, enabled 64-bit addressing, allowing memory configurations larger than 4GB.

When the Macintosh Operating System boots up, it divides the available RAM into two portions. It sets aside a memory zone or partition known as the system partition for itself. The system partition always starts with the lowest addressable byte of memory (memory address 0) and works its way up. The system partition has a system heap and a set of global variables, which are explained in the following two sections. Outside of the system partition, all memory is available for allocation to applications or other software components. Multiple apps can be opened at the same time in system software versions 7.0 and later (or while MultiFinder is active in system software versions 5.0 and 6.0). When you launch an application, the Operating System allocates it a piece of RAM known as its application partition. In general, an application only uses the memory allocated to it in its application partition.

Figure 1 depicts memory organization when multiple apps are open at the same time. The system partition is located at the bottom of the memory. Part of the remaining space is occupied by application partitions. It is worth noting that program partitions are loaded first into the top region of memory.

Figure 1 Memory Organization with Several Applications Open

High memory

System

partition

BufPtr

Low memory

ApplZone

ApplLimit

CurrentA5

System heap

System global variables

Heap

A5 world

Stack

Application 3

partition

Application 2

partition

Heap

A5 world

Stack

Application 1

partition

Stack

Heap

A5 world

Used Area

Unused area

Figure 1-1 shows three open apps, each with its own application partition. The active application is tagged Application 1. (The labels on the right side of the diagram are system global variables, as detailed in "The System Global Variables" on pages 1-6.)

30

**System Heap**

The fundamental component of the system partition is a memory space known as the system heap. In general, the system heap is reserved for the exclusive use of the Operating System and other system software components, which load various items into it such as system resources, system code segments, and system data structures. The system heap is where all system buffers and queues, for example, are allocated.

The system heap is also used for code and other resources that do not belong to any one program, such as code resources that add features to the operating system or control special-purpose peripheral equipment. During the system starting process, system patches and system extensions (stored as code resources of type 'INIT') are loaded into the system heap. When a hardware device driver is opened, its code resources of type 'DRVR' are loaded into the system heap.

Most apps do not require anything to be loaded into the system heap. However, in some circumstances, you may need to load resources or code segments into the system heap. For example, if you want a vertical retrace job to continue running while your program is in the background, you must load the task and all associated data into the system heap. Otherwise, when your program is running in the background, the Vertical Retrace Manager ignores the task.

31

**The System Global Variables**

A collection of global variables known as system global variables (or low-memory system global variables) occupy the lowest section of memory. These variables are used by the operating system to store various types of information about the operating environment. The Ticks global variable, for example, stores the number of ticks (sixtieths of a second) that have passed since the system was last started up. Similar variables include the menu bar's height (MBarHeight) and pointers to the heads of several operating-system queues (DTQueue, FSQHdr, VBLQueue, and so on). The majority of low-memory global variables are of this type: they hold information that is generally only beneficial to the Operating System or other system software components.

Other low-memory global variables contain application-specific information. The ApplZone global variable, for example, stores the address of the first byte of the active application's partition. The address of the last byte that the active application's heap can extend to include is stored in the global variable ApplLimit. The address of the boundary between the active application's global variables and its application parameters is stored in the CurrentA5 global variable. Because these global variables provide information about the currently running program, the Operating System modifies their values whenever a context switch happens.

It is preferable to avoid reading or writing low-memory system global variables in general. The majority of these variables are undocumented, and changing their values can have unanticipated implications. When the value of a low-memory global variable is likely to be beneficial to applications, the system software usually provides a routine to read or write that value. Calling the TickCount function, for example, returns the current value of the Ticks global variable.

32

**Organization of Memory in an Application Partition**

When you launch an application, the operating system creates a memory partition called the application partition for it. That partition contains required chunks of the application's code as well as other application-related data. Figure 1-2 depicts the general layout of an application partition.

Figure 2 Organization of an Application Partition

A5 world

ApplZone

ApplLimit

CurStackBase

CurrentA5

Stack

High memory

Heap

Low memory

Used Area

Unused area

Your application partition is split into three sections:

• the stack of applications

• the heap of applications

• A5 world and the application global variables

The heap is at the low-memory end of your program partition and always grows (when needed) to high memory. The A5 world is fixed in size and is stored at the high-memory end of your application partition. The stack starts at the low-memory end of the A5 universe and works its way up to the top of the heap. Figure 2 shows that there is generally an unused space of memory between the stack and the heap. This vacant space allows the stack to develop without encroaching on the space allotted to the application heap. However, in some instances, the stack may overflow into application heap space. If this occurs, the data in the heap is very likely to become corrupted.

The ApplLimit global variable specifies the maximum size of your heap. When you run the MaxApplZone method at the start of your program, the heap immediately expands to this limit. If you used up all of the heap's spare space, the Memory Manager would not let you allocate any more blocks. If you do not call MaxApplZone, the heap will grow to ApplLimit if the Memory Manager discovers that there is insufficient memory on the heap to fill a request. However, once the heap reaches ApplLimit, it can no longer grow any further. Thus, regardless of whether you optimize your application heap or not, you can only use the space between the heap's bottom and ApplLimit.

The stack, unlike the heap, is not bound by ApplLimit. If your application contains a lot of nested procedures with lots of local variables, or if you utilize a lot of recursion, the stack may expand below ApplLimit. Because you do not use Memory Manager functions to allocate memory on the stack, the Memory Manager cannot prevent your stack from going past ApplLimit and potentially infringing on heap space. A vertical retrace job, on the other hand, checks around 60 times per second to verify if the stack has moved into the heap. If this occurs, the task, known as the "stack sniffer," causes a system fault. This system error informs you that you have allowed the stack to extend too far, allowing you to make necessary adjustments.

33

**The Application Stack**

The stack is a memory space in your application partition that can expand or shrink at one end while remaining fixed at the other. This ensures that stack space is always allocated and released in last-in, first-out (LIFO) order. The object assigned last is always the first to be released. It also implies that the stack's assigned space is always continuous. Because space is only released at the top of the stack and never in the center, there can never be any "holes" in the stack. The stack grows from high memory locations to low memory addresses by convention. The growing or shrinking end of the stack is sometimes referred to as the "top" of the stack, despite the fact that it is at the lower end of the memory occupied by the stack.

The stack is notably effective for memory allocation associated with the execution of functions or procedures due to its LIFO nature. When your program calls a routine, space on the stack is automatically allotted for a stack frame. A stack frame comprises the parameters, local variables, and return address of the routine. During a function call, the stack expands and contracts as shown in Figure 3 The stack is shown in the leftmost figure just before the function is called. The stack is enlarged to hold the stack frame in the middle diagram. The local variables and function parameters are popped off the stack once the function is performed. If the function is a Pascal function, the preceding stack with the function result on top is all that remains.

Figure 3 Application Stack

High memory

Top

of stack

High memory

High memory

Function result

Low memory

Low memory

Low memory

Used Area

Unused area

34

**The Application Heap**

An application heap is a memory area in your application partition where space is dynamically allocated and released on demand. The heap starts at the low-memory end of your application partition and works its way up. Almost all items that are not allocated on the stack are stored in the heap. For example, your application heap stores the code segments and resources that are currently loaded into memory. Other dynamically allocated things in the heap include window records, dialog records, document data, and so on. You allocate heap space in your application by calling the Memory API.

Manager, either directly (using the NewHandle function) or indirectly (by a routine such as NewWindow, which invokes Memory Manager operations). The heap's space is allocated in blocks of any size required for a specific object.

The Memory Manager does all of the necessary housekeeping to keep track of heap blocks as they are created and removed. Because these activities can take place in any sequence, the heap does not typically expand and decrease in the same manner that the stack does. Instead, after a while of running your application, the heap may get fractured into a patchwork of allocated and free blocks, as seen in Figure 4. This fragmentation is known as heap fragmentation.

Figure 4 A Fragmented Heap

High memory

Allocated blocks

Free blocks

Low memory

As a result of heap fragmentation, the Memory Manager may be unable to fulfill your application's request to allocate a block of a specific size. Even when there is adequate free space, it is divided into blocks that are smaller than the necessary size. When this occurs, the Memory Manager attempts to create the required space by combining allotted blocks and collecting the free space in a single larger block. This is referred to as heap compaction. Figure 5 depicts the outcome of compacting the fractured heap depicted in Figure 4.

Figure 5 A Compacted Heap

High memory

Allocated blocks

Free blocks

Low memory

Heap fragmentation is usually not an issue as long as the memory blocks you allocate are free to migrate during heap compaction. However, a block is not free to move in two situations: when it is a nonrelocatable block and when it is a locked, relocatable block. To reduce heap fragmentation, utilize nonrelocatable blocks sparingly and only lock relocatable blocks when absolutely essential. See "Relocatable and Nonrelocatable Blocks" on page 1-16 for further information on relocatable and nonrelocatable blocks, and "Heap Fragmentation" on page 1-24 for information on how to avoid fragmenting your heap.

35

**References**

<https://encyclopedia.pub/entry/34716>

Usman January 26, 2023 <https://www.learnatnoon.com/s/en-pk2/difference-between-temporary-memory-and-permanent-storage/>

Alexander S. Gillis <https://www.techtarget.com/searchstorage/definition/virtual-memory>

<https://apple.fandom.com/wiki/24-bit_addressing>

<https://apple.fandom.com/wiki/32-bit_addressing>

36

CHAPTER 5

**STORAGE MANAGEMENT**

**Storage Management**

Disk Utility is a sophisticated system utility application that has long been a feature of the macOS operating system. It is a complete solution for managing and preserving storage devices on Apple Macs, with a variety of functions to assure disk health, performance, and organization. Disk Utility is a versatile utility that caters to the different demands of casual users and sophisticated specialists, from disk management to troubleshooting and repair.

Disk Utility, at its core, allows users to manage their storage devices, including as internal and external hard disks, SSDs, USB drives, and others. Disk formatting is one of its key features, allowing users to prepare a storage device for use with their Mac. Disk Utility supports a variety of file systems, including HFS+, APFS, and FAT32, when creating a new disk or reformatting an existing one. This adaptability is essential for compatibility with various devices and operating systems.

Disk Utility excels at partition management in addition to formatting. Users can build, resize, and delete partitions on their storage devices, allowing them to better manage their data. This functionality is very handy for users who want to utilize different parts of a drive for different purposes, such as creating a separate partition for backups or setting up a dual-boot system with multiple operating systems.

Another important part of Disk Utility's usefulness is its diagnostic capabilities. The program performs health and integrity checks on storage devices, assisting users in identifying and addressing potential issues before they worsen. Users can check and repair disk permissions, which is essential for a stable and efficient file system. Disk Utility can also be used to launch First Aid, a diagnostic program that scans and repairs common disk-related issues such as directory corruption and file system failures.

Disk Utility offers strong encryption options for those who want to encrypt their data. FileVault, Apple's disk encryption tool, is simple to install and use, offering an extra degree of security to sensitive data. Encryption ensures that even if a device falls into the wrong hands, the data is safe and inaccessible without the appropriate credentials.

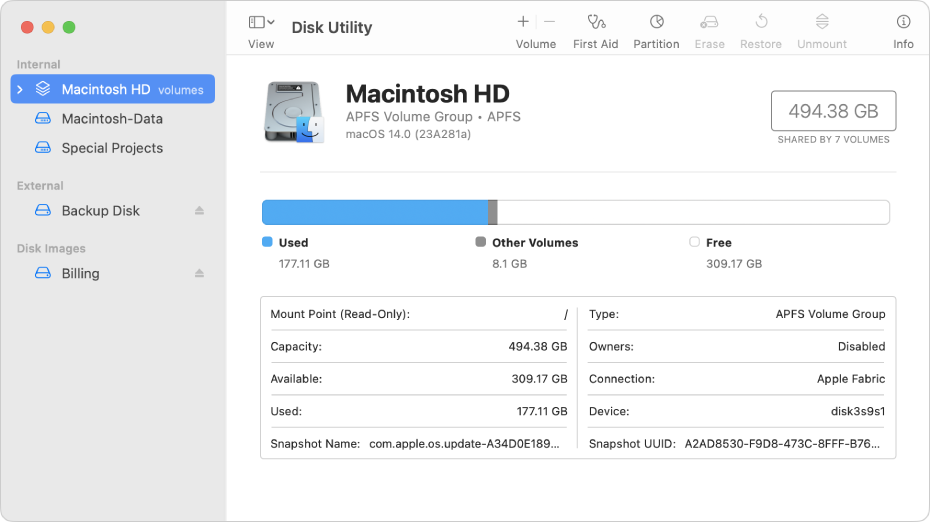
Disk Utility is also important in the administration of disk images. Disk image files, which are virtual representations of real disks or storage volumes, can be created, mounted, and manipulated by users. This capability is extremely useful for operations such as backup creation, software distribution, and data archiving. Disk Utility supports a range of disk image types, giving users the flexibility, they require.

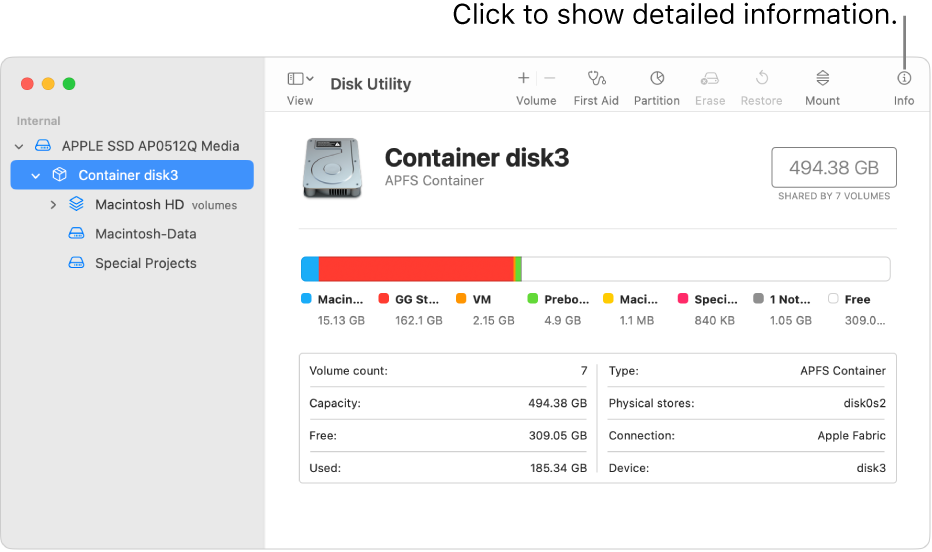
Disk Utility has grown alongside macOS. Disk Utility has effortlessly transitioned to the move from the traditional HFS+ file system to the more contemporary and efficient APFS (Apple File System). APFS has advantages like higher performance, space efficiency, and data integrity, and Disk Utility aids in the transition by providing tools for converting and managing drives using this advanced file system.

The user interface of Disk Utility is designed to be intuitive and user-friendly, making it accessible to a wide range of users. The app is separated into tabs, each of which focuses on a different area of disk management, such as First Aid, Partition, Erase, and RAID. This structure simplifies navigation and ensures that consumers have quick access to the tools they require without undue complication.

Disk Utility has a command-line interface (CLI) counterpart known as "diskutil" for experienced users and IT professionals, which enables the automation and scripting of disk-related activities, enabling a more hands-on approach for individuals who prefer command-line interactions.

Disk Utility is a comprehensive and necessary utility for macOS users, offering a rich collection of capabilities for disk management, diagnostics, and troubleshooting. Disk Utility's versatility and dependability make it a cornerstone of macOS' utility toolset, whether you're a casual user wanting to format a new external drive or a seasoned professional dealing with sophisticated storage difficulties. Disk Utility will most likely adapt to meet the shifting world of storage devices and user needs as technology advances, guaranteeing that Mac users can continue to rely on this vital tool for years to come.





37

**Utility and Disk Structure in Mac OS**

The normal structure of our Mac's disks and volumes was relatively basic before Apple introduced APFS. What we saw in the Finder and Disk Utility was reasonably similar to what we wanted to utilize, and it wasn't too far off from what you'd discover in Terminal using the diskutil command. Then came High Sierra and Mojave with APFS, which complicated things even more with features like containers, and the simple illusion provided in the Finder didn't bore much relation to what Disk Utility presents, and is much further removed from Terminal's shell. With Catalina and its strangely coupled System and Data volumes, everything appears to be a jumbled mess. This tutorial will help you learn the Finder, Disk Utility, and the command line in the most recent version of macOS.

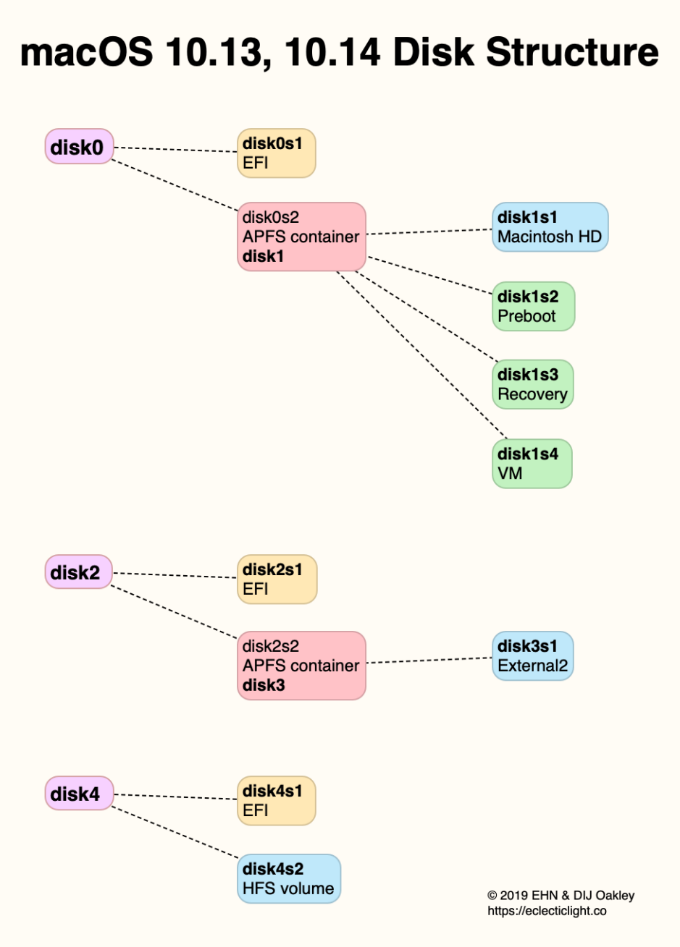
There are three ways to view your Mac's drives and volumes, which do not necessarily correspond exactly:

• The Finder displays a streamlined view that is required for everyday handling of documents, programs, and so on. By using Command-Shift-, you can make it more literal. (period or full stop) to uncover concealed elements, although this is only a temporary solution and still does not give full detail.

• Disk Utility demonstrates how to create and operate with storage volumes on various disks. However, it does not provide complete information or even display all accessible volumes.

• The diskutil command in Terminal provides a lot of detail that you rarely require in day-to-day work, and is meant for use when referring to volumes using their Unix names, such as disk3s2. There are two important commands that disclose detailed disk and volume structures: the older diskutil list and the newer diskutil apfs list, which deals with APFS volumes.

**High Sierra and Mojave**

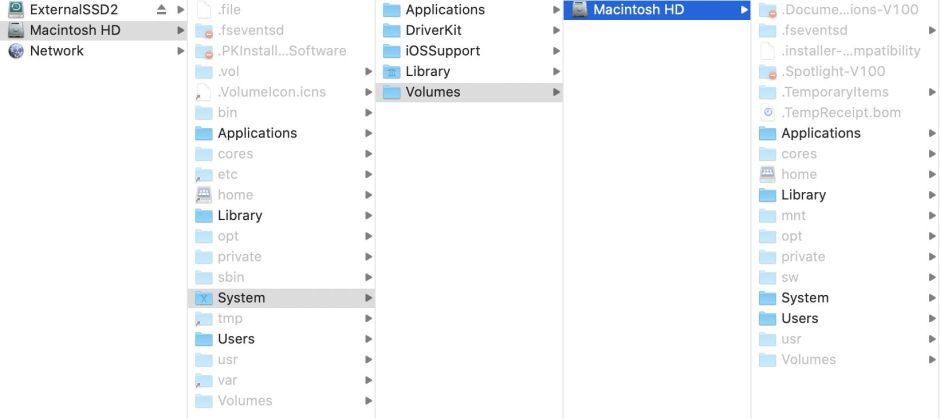


Internal and external disks are typically divided into two types: those that use the old HFS+ (Mac Extended) file system, which is still necessary for local Time Machine backups, and those that use the new APFS. Each disk has a secret EFI partition, as well as HFS+ volumes and APFS containers as they are generated. One significant distinction between HFS+ and APFS is that APFS places its volumes within containers rather than at the top level of the storage system. This is due to the fact that containers behave like HFS+ volumes, with defined sizes and the inability to share free disk space. However, because they all share space within the same container, APFS volumes within the same container can expand and shrink in size as needed. A simple external disk used for storage in APFS format comprises one APFS container, which contains a single APFS volume. This is seen in the diagram's second example, disk2, and disk4 is its equivalent in HFS+.

38

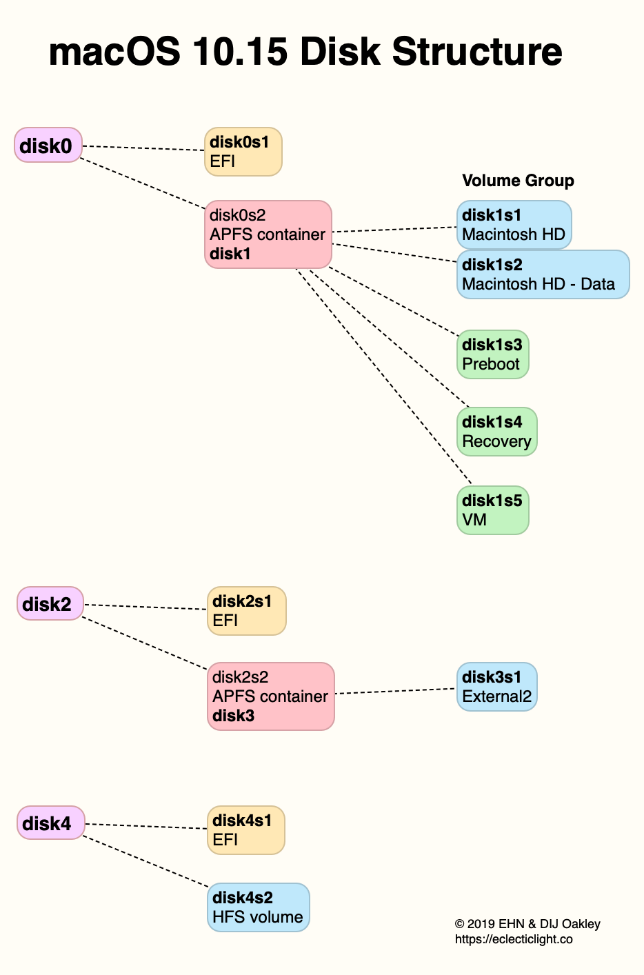
**Catalina**

The change introduced to this structure in Catalina may appear simple: there is a fifth volume in a bootable APFS container. To maintain the illusion of simplicity in the Finder and to connect the new System and Data volumes, more smoke and mirrors are required, resulting in quite different structures.



When viewed in the Finder with hidden items revealed, there appear to be two volumes both named Macintosh HD, with the top-level volume containing essentially the same folders as an older single-volume layout, but drill down through the System folder to Volumes, and you'll find another Macintosh HD with a different set of folders. In actuality, the top-level Macintosh HD volume is the new read-only System volume, while the lower-level one is what you see as Macintosh HD - Data, which contains all of your writable files and folders. Instead of mounting the Data drive in the regular Volumes folder (as it would have in Mojave), macOS mounts it in /System/Volumes.

**Mac OS 10.15 Disk Structure**



The main difference in the actual volume configuration is that disk1s1 and disk1s2 form a Volume Group within your boot disk disk0's bootable container disk1. Otherwise, the layout is the same, and this only applies to bootable APFS containers on conventional storage, not non-bootable APFS containers.



Disk Utility displays both the System and Data volumes within the bootable container for consistency, but because Macintosh HD is read-only and protected by SIP, you won't be doing much with it here. The container's contents are listed in their whole, with the two volumes composing its Volume Group, VM, and the two unmounted as previously. As expected, the old command diskutil list returns minimal information about the volumes. Their true nature is revealed by the new diskutil apfs list. Disk1s1 has the System role and is mounted as /System/Volumes/System, whereas disk1s2 has the Data role and is mounted as /System/Volumes/Data. Paths grow more convoluted in Terminal as well, thanks to firmlinks that ensure as much consistency as feasible with previous disk directory trees. This article offers comprehensive roadmaps to the new trees if you wish to see these in depth.

The nomenclature of Unix volumes is not fixed. When no volumes have been removed or recreated, the names I've supplied above are what you should expect to see in macOS. If you delete the Data volume during a clean re-install, you should anticipate the individual volumes within that container to have different Unix names than those presented. Unless you or a script is expected to discover the Data volume at disk1s2, this shouldn't make a difference. That would be incredibly bad form.

39

**Hard Disk Drive**

A hard disk drive (HDD; alternatively hard drive, hard disk, or HD) is a non-volatile, magnetic data storage device with random access. It has inflexible rotating platters on a motor-driven spindle inside a protective casing. Data is magnetically read from and written to the platter by read/write heads that hover above the platters on a film of air. Hard disk drives, first disclosed by IBM on September 14, 1956, have fallen in cost and physical size while expanding in capacity substantially over the years.

Since the early 1960s, hard disk drives have been the dominating device for secondary data storage in general-purpose computers. They have retained this position because advancements in recording density have kept pace with secondary storage requirements. Today's hard disk drives (HDDs) use high-speed serial interfaces, such as serial ATA (SATA) or serial attached SCSI (SAS).

Apple Computer's first hard drive, the ProFile, was released in September 1981 for the Apple III. This external drive included a 5 MB Seagate mechanism, which provided more storage space and faster performance than floppy drives. The Apple II and Lisa series computers were also supported by the ProFile. The Hard Disk 20 was Apple's first external hard drive available for early compact Macs. Though it offered more space than the 400 KB floppy disks used at the time, its maximum throughput was limited by its reliance on being connected via the floppy drive connector. The first Macs to be designed to accommodate an internal hard drive were the Macintosh SE and II, released in March and April 1987, respectively. These made use of a dedicated internal SCSI interface to increase maximum throughput. Apple began using the faster SATA interface for internal hard drives with the Macintosh Quadra 630 in July 1994, transitioning from SCSI to the more frequently used IDE interface. With the release of the Power Mac G5 in June 2003, Apple then began using the faster SATA interface for internal hard drives. Apple began using solid-state drives (SSDs) using flash memory instead of magnetic platters of hard drives to reduce the size of their iPod music player line with the release of the first-generation iPod shuffle in January 2005. This extended to their Mac product line in January 2008 with the introduction of the MacBook Air, which offered an optional SSD upgrade in place of a hard drive for high-speed, low-latency storage. Apple no longer provided hard disks (or hybrid drives) as a factory option in their current product portfolio with the introduction of the M1-based 2021 iMac.

40

**References**

<https://support.apple.com/en-ph/guide/disk-utility/dskutl1029/mac>

<https://support.apple.com/en-ph/guide/disk-utility/dskutl1005/mac>

Alexander S. Gillis <https://www.techtarget.com/searchstorage/definition/hard-disk-drive>

<https://apple.fandom.com/wiki/Hard_disk_drive>

Hoakley December 18, 2019 <https://eclecticlight.co/2019/12/18/making-sense-of-disk-utility-and-disk-structure-in-macos-10-13-10-15/>

41

CHAPTER 6

**I/O SYSTEM**

**I/O System**

The Input/Output (I/O) system is a critical component in macOS, controlling the sophisticated web of communication that connects the operating system, device drivers, and hardware peripherals. The I/O Kit framework, a strong and object-oriented architecture precisely built to support the smooth interaction of varied hardware components, is at the heart of this system. The I/O Kit framework, known for its versatility, is purpose-built for the development and integration of device drivers, operating as a bridge between software and hardware layers. This framework enables developers to construct drivers that are efficient, flexible, and scalable and can efficiently interface with a wide range of devices.

The I/O Kit framework adheres to object-oriented principles, with a modular structure that improves code reusability and maintainability. The framework's object-oriented design allows for the construction of driver classes that encapsulate the functionality and behavior unique to different hardware devices. This modular approach simplifies driver development while simultaneously fostering a high level of abstraction, protecting developers from the delicate minutiae of hardware intricacies. As a result, the I/O Kit framework enables developers to concentrate on designing device-specific features rather than being overwhelmed by the intricacies of low-level hardware interactions.

The I/O Kit framework orchestrates a smooth communication flow within the macOS I/O system. It controls the interaction between device drivers and the operating system, ensuring that data is delivered between software and hardware components in an efficient and reliable manner. This orchestration includes a number of tasks, such as device finding, driver loading, and communication channel setup. The I/O Kit architecture contributes to the entire system's stability and reliability by providing a standardized and consistent interface for device drivers.

Furthermore, the I/O Kit architecture facilitates the creation of user-space programs that require interaction with hardware peripherals. Developers can construct applications that take advantage of the capabilities of connected devices by using properly defined interfaces and APIs. This expands the I/O Kit framework's reach beyond kernel-level development, promoting a full ecosystem for programs to use the many features afforded by macOS hardware peripherals.

Apple's commitment to encouraging a seamless and efficient interface between software and hardware is exemplified by the Input/Output system in macOS, which is powered by the powerful I/O Kit foundation. The I/O Kit framework helps developers to construct safe and scalable solutions that contribute to the overall stability and performance of the macOS operating system by encapsulating the intricacies of device driver development within a modular and object-oriented architecture. The I/O Kit framework is a cornerstone in the architecture of macOS, facilitating a harmonious interplay between software and hardware components by facilitating communication between device drivers and the kernel or enabling user-space applications to harness the power of hardware peripherals.

42

**I/O KIT**

The I/O Kit is a collection of system frameworks, libraries, tools, and other resources for developing device drivers in OS X. It is based on an object-oriented programming model written in a restricted form of C++ that excludes aspects that are unsuitable for usage within a multithreaded kernel. The I/O Kit accelerates the device-driver development process by modeling the hardware connected to an OS X system and abstracting common functionality for devices in specific categories. OS X is essentially the result of two operating-system strains: Mac OS 9 (and its successors) and BSD. Given this background, one may anticipate Apple to embrace the Mac OS 9 or FreeBSD device-driver model. Apple instead chose to revamp the device. This choice was motivated by several factors. The OS X kernel is significantly more advanced than its Mac OS predecessors; it handles memory protection, preemptive multitasking, multiprocessing, and other features do not present in previous versions of Mac OS. Although FreeBSD can handle these features, the BSD model does not offer other features expected in a modern operating system. As a result, the fundamental impetus for developing the I/O Kit was the insufficiency of already available driver models. The redesign of the I/O architecture had to take advantage of and support OS X's operating system features. To that end, the I/O Kit's designers chose an object-oriented programming model that abstracted an OS X system's kernel capabilities and hardware and provided a view of this abstraction to the operating system's upper layers. The most appealing aspect of this abstraction is the implementation of behavior shared by all device drivers (or types of device drivers) in the I/O Kit classes. Consider virtual memory as an example. Virtual memory is not a required feature of Mac OS 9, but it is available as an option. As a result, when building a driver, a developer must always consider virtual memory, which adds complexities. In contrast, virtual memory is a built-in feature of OS X that cannot be disabled. Because virtual memory is a fundamental and presumed feature, it is built into system software and driver authors are not required to account for it. The I/O Kit serves as a foundation and a coordinator for device drivers. This model differs from prior driver models. All software development kits (SDKs) in Mac OS 9 are independent of one another and duplicate similar features. The I/O Kit is delivered as part of a single kernel development kit (KDK); all components of the KDK share similar underpinnings. OS X allows developers to benefit from hardware complexity without having to embed software complexity into each new device driver. In most cases, they merely need to add the code that distinguishes their drivers.

43

**Mac OS User Space**

Beginning with macOS 11, the Carbon Black Cloud macOS sensor (v3.5.1) runs in user-space via System Extensions (user-space) rather than Kernel Extensions (KEXTs) as in previous versions of the agent. As a result, while using the sensor in System Extension mode on macOS 11 and later, there are some functional variations. Using the sensor in KEXT mode on macOS 11 delivers the same functionality as it does on prior operating systems. Unless otherwise noted, documentation about macOS functionality on the Carbon Black Cloud applies to macOS 10.15 and older, or to functionality given via the KEXT on macOS 11. The matrix below describes macOS functionality on the Carbon Black Cloud. The functionality listed in the macOS 11+ column refers to the sensor's user space (System Extension) capabilities in the initial macOS 11-compatible sensor release (v3.5.1+). Refer to the macOS 10.12 - 11+ column for functionality provided by the kernel extension.

44

**Device Driver**

A device driver is software that controls a computer's hardware component or peripheral device, such as a hard drive or printer. MacOS devices are managed by the DriverKit framework. A device driver is in charge of accessing the device's hardware registers and frequently contains an interrupt handler to handle interruptions issued by the device.

Device drivers are frequently part of the operating system kernel's lowest level, with which they are linked when the kernel is constructed. Some newer systems provide loadable device drivers that can be installed from files after the operating system has started.

45

**I/O Registry**

The I/O Registry is a dynamic database that specifies a set of "live" objects (nubs or drivers) and keeps track of their provider-client interactions. When new or removed hardware is added or removed from the system, the Registry is immediately updated to reflect the new device configuration. The Registry, a dynamic component of the I/O Kit, is not saved on disk or archived between boots. It is instead constructed at each system boot and stored in memory. APIs in the I/O Kit framework make the I/O Registry available from user space. These APIs feature robust search methods that allow you to search the Registry for objects that meet specific criteria. You can also use the developer version of OS to inspect the current state of the Registry on your computer.

46

**System and Kernel Extension in Mac OS**

Developers can extend the capabilities of macOS by installing and administering system extensions that run in user space rather than at the kernel level in macOS 10.15 or later. System extensions improve macOS stability and security by executing in user space. Unlike kexts, which have full access to the whole operating system, extensions running in user space are only allowed the capabilities required to fulfill their designated job.

System extensions can be managed robustly using MDM, including the ability to allow all extensions from a certain developer or type (such as network extensions) to load without user intervention. MDM can optionally prevent users from approving their own system extensions from loading. Making changes to a system extension profile directly affects the state of an extension in macOS 11.3 through macOS 11.6.4. For example, if an extension is awaiting approval and a configuration profile that authorizes the extension is pushed, the extension is permitted to load. If an approval is revoked, the system extension is unloaded and marked for removal on the next Mac restart. If a system extension attempts to unload itself, an interactive authentication dialog occurs in which administrator credentials are required to authorize the unloading. In macOS 12.0.1 or later, a dictionary called RemovableSystemExtensions in the System Extensions payload lets an MDM administrator to designate which apps should be able to delete their own system extensions. To remove the system extensions, no local administrator credentials is necessary. This is especially beneficial for suppliers who offer automated uninstallers for their apps.

47

**Kernel Extension**

If third-party kernel extensions (kexts) are enabled in macOS 11 or later, they cannot be loaded into the kernel on demand. They require the user's approval and restarting of macOS to load the changes into the kernel, and they also require that the secure boot on a Mac with Apple hardware be set to Reduced Security. Developers can utilize frameworks like DriverKit and NetworkExtension to create USB and human interface drivers, endpoint security solutions (such as data loss prevention or other endpoint agents), and VPN and network tools without writing kexts. Third-party security agents should be utilized only if they use these APIs or have a solid plan to migrate away from kernel extensions.

48

**References**

Garry Kranz <https://www.techtarget.com/whatis/definition/input-output-I-O>

Christopher Trick Jun 3, 2022 <https://www.trentonsystems.com/blog/what-is-i/o#:~:text=I%2FO%20(Input%2FOutput)%20is%20an%20information%20processing,to%20other%20systems%20or%20devices.>

<https://docs.vmware.com/en/VMware-Carbon-Black-Cloud/services/cbc-endpoint-standard-oer/GUID-DCE8B7D9-67CA-46A2-8B79-B08B0B41F35F.html>

<https://developer.apple.com/library/archive/documentation/DeviceDrivers/Conceptual/IOKitFundamentals/Features/Features.html>

Kernel and Device Layer at Apple Developer (2015-09-16) <https://apple.fandom.com/wiki/Device_driver>

<https://developer.apple.com/library/archive/documentation/DeviceDrivers/Conceptual/IOKitFundamentals/TheRegistry/TheRegistry.html>

<https://support.apple.com/en-ph/guide/deployment/depa5fb8376f/web>

**49**

CHAPTER 7

**FILE SYSTEM**

50

**Apple File System for Mac OS**

APFS (Apple File technology) is a proprietary data organization and structuring technology for storage systems. Apple's APFS file system, which debuted with macOS High Sierra, replaces the 30-year-old HFS+ system that had previously been used on Macs. HFS+ and HFS (an earlier version of the Hierarchical File System) were developed during the period of floppy disks, which served as the primary storage media for the Mac when spinning hard drives was an expensive option provided by third parties. Over the years, Apple has toyed with the idea of replacing HFS+. APFS, which is already included on iOS, tvOS, and watchOS, eventually became the default file system for Macs running macOS High Sierra and later. APFS is included in macOS 10.15, macOS Mojave (10.14), and macOS High Sierra (10.13), as well as iOS 10.3 and later, tvOS 10.2 and later, and watchOS 3.2 and later. HFS+ and its predecessor HFS were developed during an age dominated by floppy disks, when the concept of rotating hard drives was considered a luxury given by third-party businesses. These file systems were designed to satisfy the needs of the Mac ecosystem at a period when storage options were scarce and expensive. However, as technology improved, the limits of HFS+ became clear, pushing Apple to look at alternatives. The path to a new file system was not impetuous, but rather a planned evolution. Before committing to APFS, Apple experimented with numerous approaches. The need for a more robust, efficient, and modern file system became especially apparent as Macs entered a new computing era. With the release of macOS High Sierra, HFS+ was officially phased out in favor of the more advanced APFS. APFS is distinguished not just by its private nature, but also by its versatility. APFS has made its way into other Apple operating systems in addition to macOS High Sierra, consolidating its position as the default file system. This cross-platform integration includes iOS, tvOS, and watchOS, demonstrating Apple's dedication to providing a consistent user experience across its wide product lineup. As technology advances, the significance of APFS in molding the user experience on Apple products is expected to grow even more. Apple's continuing attention to refinement and innovation demonstrates the company's commitment to provide people with cutting-edge solutions that not only meet but exceed their expectations. In summary, APFS represents a watershed moment in Apple's history, signaling a departure from the past and setting the way for a future in which storage systems are optimized for the demands of modern computing.

51

**Optimized for Modern Storage Technology**

When 800 KB floppies were the norm, HFS+ was implemented. Floppy disks are no longer used in modern Macs, and spinning hard drives are becoming obsolete. With Apple pushing flash-based storage in all of its products, a file system geared for rotational media and the inherent latency of waiting for a disk to spin around makes little sense. APFS was created by Apple specifically for SSD and other flash-based storage systems. APFS is optimized for solid-state storage, but it also works well with current hard drives.

52

**Futureproofing**

A 64-bit inode number is supported by APFS. The inode is a unique identifier that identifies a file or folder in the file system. With a 64-bit inode, APFS can contain approximately 9 quintillion file system objects, above the previous maximum of 2.1 billion. Nine quintillion is a large figure, and you may be wondering what storage system can hold so many objects. The solution necessitates a look at storage trends. Consider this: Apple has already begun to integrate enterprise-level storage technology into consumer-level devices, such as the Mac and its tiered storage capability. This was originally observed in fusion drives, which transfer data between a fast SSD and a slower, but much larger, hard drive. Data that is often accessed is maintained on the fast SSD, while items that are utilized less frequently are kept on the hard drive. Apple expanded on this notion with macOS by including iCloud-based storage. Allowing customers to keep previously watched movies and TV shows in iCloud frees up local storage. While this example does not necessitate a unified inode numbering system across all disks in use by this tiered storage system, it does demonstrate Apple's general direction of bringing together multiple storage technologies that best fit the needs of the user and having the OS see them as a single file space.

53

**APFS Features**

APFS has several features that distinguish it from older file systems.

Clones: Clones provide near-instantaneous file copying without the need for additional storage capacity. Clones, rather than duplicating a file bit by bit from one location to another, refer to the original file, sharing data blocks that are identical between the two files. Change one file, and only the altered block of data is written to the new clone, while the original and clone continue to share unchanged blocks of data. This not only speeds up file transferring and storing, but it also reduces storage space requirements.

Snapshots: APFS has the ability to create a volume snapshot that represents a point in time. Snapshots provide efficient backups and allow you to return to how things were at a specific moment in time. Snapshots are read-only references to the original volume's data. A new snapshot consumes no real space other than the space required to store a pointer to the old volume. As time passes and modifications are made to the original disk, the snapshot is updated to reflect only those changes.

Encryption**:** APFS offers AES-XTS or AES-CBC encryption modes for powerful entire disk encryption. Files and metadata are both encrypted. Clear (no encryption) is one of the encryption methods supported.

Space Sharing: Predefined partition sizes are no longer necessary with space sharing. Instead, all volumes share the drive's underlying free space. Space sharing enables different volumes on a disk to grow and shrink as needed without the requirement for repartitioning.

Copy-On-Write: This data protection strategy permits the sharing of data structures as long as no changes are performed. When a change is requested (written), a new unique copy is created while the original is preserved. The file metadata is only updated to point to the most recent data when the write is complete.

Atomic Safe-Save: This concept is similar to copy-on-write, except it applies to any file operation, such as renaming or moving a file or directory. In the case of rename, the file about to be renamed is copied with the new data (the file name). The file system is not modified to point to the new data until the copy procedure is complete. This ensures that if the write fails for whatever reason, such as a power outage or a CPU hiccup, the original file remains intact.

Sparse Files: This more efficient method of allocating file space allows file space to expand only when it is required. Even if no data is ready to be stored, file space must be reserved in advance in non-sparse file systems.

54

**HFS+**

HFS+ (Hierarchical File System Plus), popularly known as Mac OS Extended, debuted with Mac OS 8.1 in 1998 and served as the primary filesystem for Mac computers, as well as iPod and Xserve products, until it was replaced by APFS in macOS High Sierra 10.13. It began as an expansion to the legacy HFS format, which is now nearly forty years old. HFS+ employs a journaling method to prevent structure corruption. All changes are documented in the Journal area, allowing them to be quickly restored in the event of an unexpected incident, such as a power outage. The disk Header is a basic structure of HFS+ that is provided at the beginning of an HFS+ disk. It includes the general FS parameters as well as the placements of other critical items. The majority of the additional service information is grouped into special files that may be found throughout the book and are primarily represented by B-trees. In HFS+, the entire storage space is divided into equal allocation blocks. The bitmap-like Allocation File stores the state of each allocation block. As a general rule, such blocks are assigned to files in continuous groups, which aids in fragmentation reduction. Files may be associated with two sets of data. A data fork stores the actual file's content, while a resource fork stores additional information about it. An extent is a contiguous series of blocks belonging to a fork that is represented by its starting location and number of blocks. Every file and directory in the filesystem is represented in the Catalog File. These records provide the majority of the metadata as well as the first eight extents of each fork. Additional extents are saved in the Extents Overflow File if they are available. Finally, the Attributes File stores additional attributes relating to files and directories. HFS+, for example, permits numerous references to the same file's content, known as hard links. Hard links, unlike conventional files, do not require additional storage space; they reside in the Catalog File as pointers to the original file, which is transferred to the hidden root directory. Although HFS+ is now obsolete, it still offers the benefit of reverse compatibility, allowing access to systems running previous versions of macOS. As a result, it is unlikely to become obsolete very soon.

55

**References**

Tom Nelson July 10, 2021 <https://www.lifewire.com/apple-apfs-file-system-4117093>

<https://www.ufsexplorer.com/articles/macos-file-systems/>