Muhammad Choudhury

Chowdhury Jawaad

Evaluation of Two Modern Operating Systems: macOS Vs. Windows

COP 4610 Computer Operating Systems

November 28th, 2020

Table of Contents

Introduction

MacOS is the Unix-Based Operating System created and maintained by Apple Inc. which runs on all of Apple computers. It was initially released on March 24th, 2001 and is currently the second most popular desktop OS after Microsoft Windows. The latest version is MacOS Catalina (version 10.15) which was released on October 7th, 2019. (Gerbarg 5)

Apple's distinguished OS runs on both desktops like the iMac as well as on laptops such as MacBooks. There are several annual updates to MacOS which are mainly security patches and also add in more functionalities even to older Mac computers which are notable for their longevity. The OS was designed with the goal of attracting and satisfying all types of users such as general consumers, professional users who need multimedia editing, and power users whose needs require a lot of computing power. (Gerbarg 3)

The other operating system that we will be looking into is the family of Windows operating systems. Windows is by far the most popular choice for personal computers, it is estimated that more than 90 percent of all personal computers are running Windows OS, the other 10 percent includes Linux and Mac operating systems. Windows has a larger user base because the operating system was designed for attracting all types of users, from the general consumers, professionals in the industry to the International Space Station. NASA officials who took care of the computers aboard the ISS relied on both Linux-based and Windows systems throughout the station's history. Let's take a brief look at the history of the Windows operating system, from the very first one to the current version. (Vangie Beal pg1-3)

<u>Windows 1.0 -2.0</u>

This was introduced in 1985, and it is the first graphical personal computer operating environment that was developed by Microsoft. This version allowed users to use point and click to access the windows, instead of typing MS-DOS into the terminal.

Windows 3.0 - 3.1

This version was released in May 1900, and it provided the standard look of Microsoft for many years to come. This computer performed better and had advanced graphics with 16 colours that was designed for the 386 processor. This version included program manager, print manager, file manager, and had some games like Minesweeper, and Solitaire.

Windows 95

This version of Windows was released in 1995, and it included some major updates to the operating system. This had a better and new version of the user interface, and included important internal improvements, such as supporting 32-bit applications. Another new implementation in the operating system was computer recognizing and configuring installed hardware, meaning plug and play.

Windows 98

This version of Windows was released in 1998, and it supported many new technologies, such as AGP, MMX, FAT32, DVD, ACPI, and USB. This main update to this version was the active desktop, which integrated the web browser with the operating system. This creates a consistent view for the user since there is no difference between accessing some documents locally, or on a web server somewhere around the world.

Windows ME - Millennium Edition

This version was an updated version of Windows 98, and also had some features that will be present in windows 2000. Windows ME was a transitional computer, it had the common features and applications, but also some new technologies.

Windows NT 31. - 4.0

Windows NT stood for new technology, and this version was available from 1993 to 1996. This 32-bit operating system that can do multitasking. There were two versions of Windows NT, the Windows NT Server, which is used as a server in networks, and Windows NT Workstation, which is a stand-alone or client workstation.

Windows 2000

Released in 2000, this operating system was used for business desktop and laptop systems in order to run software applications. Multitasking and accessing the directors and the internet was made more simpler. Easy access to files, printers, and network resources. There were 4 versions of windows 2000, the Professional, Server, Advanced Server, and datacenter Server.

Windows XP

This version of Windows was released in 2001. The operating system is built on the windows 2000 kernel, so it is a more stable and reliable user interface environment. This also supports two versions, there are the Home and Professional versions, and this XP is Microsoft's best-selling product.

Windows Vista

Vista was released in 2006, and this operating system offered more advanced security measures, was more reliable, and was easy to use. Performance and manageability of the operating system was greatly improved as well. Vista simplified the desktop configuration management, allowing reduction in the cost of keeping systems updated.

<u>Windows 7</u>

Windows 7 was released in 2009, and this marked the 25 year anniversary for Microsoft in producing the Windows operating system. Windows 7 is improved upon multi-touch support, internet explorer 8, it has improved performance and start-up time.

<u>Windows 8</u>

Windows 8 was released in 2012, and the user interface was completely redesigned. Many users were not happy with the new layout, but this was needed because this operating system layout was created with touchscreen in mind. This new interface worked well with tablets and other hand-held devices.

<u>Windows 10</u>

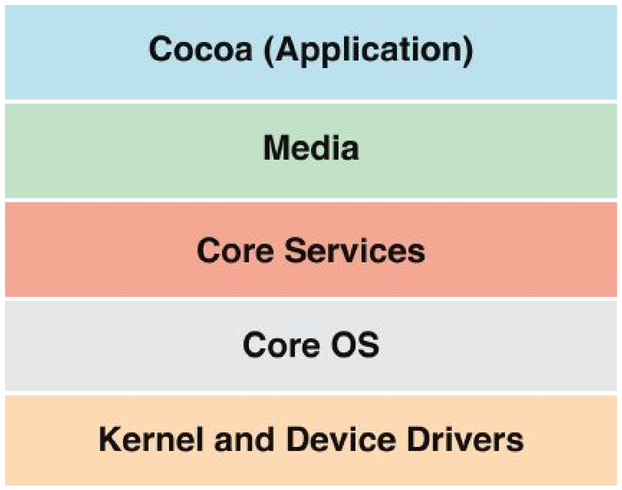
The current version of Windows is the Windows 10, this was released in 2015, and it is still the version that is used as of 2020. This version of windows also features Microsoft Edge, and a new browser. Microsoft has no plans of creating Windows 11, and plans on supporting and updating Windows 10.

These two Operating Systems are the most well-known and widely used OSs on the face of this planet and possibly throughout human history. There are benefits and drawbacks to using each OS along with differences in the technical aspects in each of them which we will explain throughout this project report. Although there is no "right" answer as to which system is better, we will do our best to analyze every detail and then come up with an educated determination as to which OS is better to use. (Wonglimpiyarat 2)

OS Architecture

Apple's MacOS has a layered architecture. The major segments of the OS can be shown in the figure below where the layers that are shown consist of technologies that are more specialized in descending order. The Cocoa layer manages user applications, user events, and the application behaviour. (Apple 17)

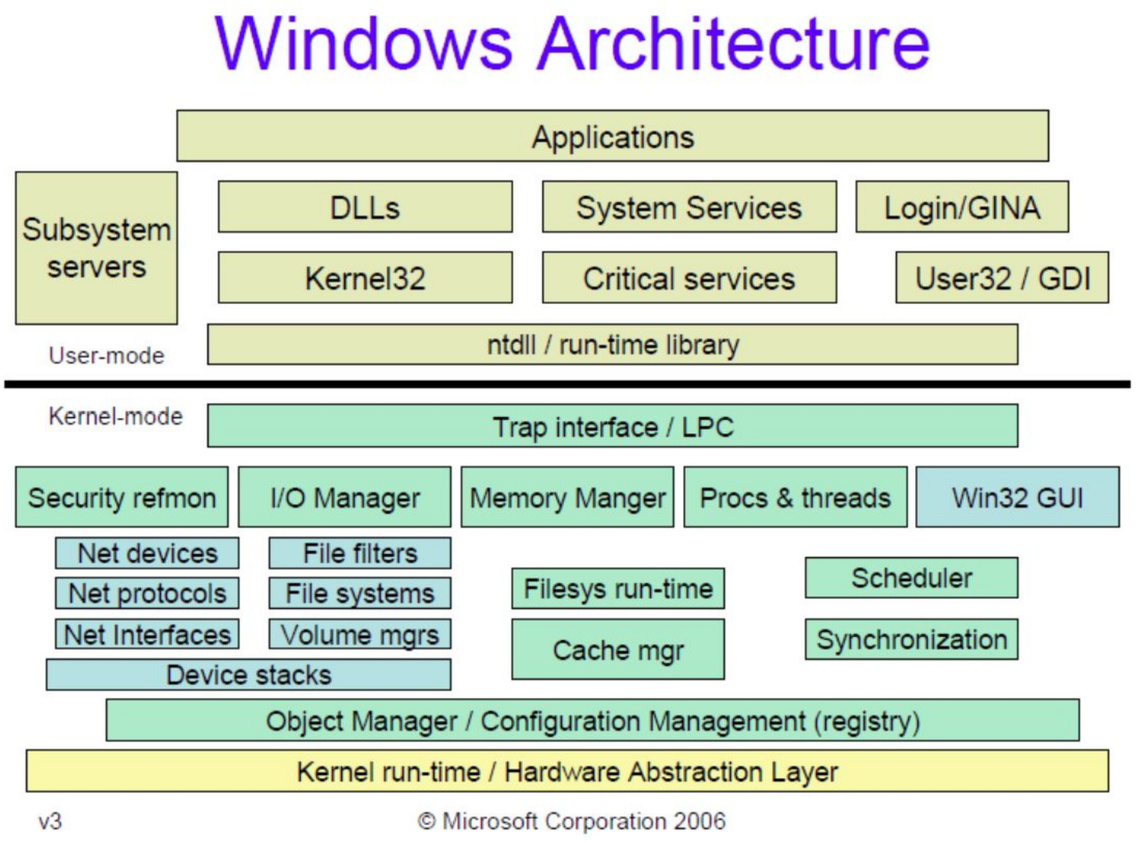
The Media layer is in control of the editing, playing, and recording of all audio/visual media and the rendering of both 2D and 3D graphics. The Core Services layer includes fundamental services and functionalities like network communication and data formatting. The Core OS layer controls all interfaces related to networking and hardware devices such as the ones running on the CPU and GPU. (Apple 17)



(Apple 15)

The Kernel and Devices layer, also called Darwin, includes the XNU kernel which is a hybrid kernel combining the Mach microkernel and the FreeBSD monolithic kernel and the IO kit framework for drivers. The layer provides support for security, file systems, extensions to kernel, programming languages, and device drivers. It also handles the interprocess communication with the other layers of the OS architecture. (Apple 17)

Now looking at the Windows NT, which is a line of operating systems developed by Microsoft. The architecture of the operating system in Windows NT has a layered characteristic. The design of the architecture consists of two main parts, there is the user mode and kernel mode. The NT operating system is a preemptive, reentrant system, and it is designed to work with uniprocessor and multiprocessor SMP based computers. (Sulung, Putra pg 2-3)



(Sulung 4)

In the figure above we can distinguish between the two types of NT architecture, the User-mode and Kernel-mode, with many different modules within both these layers. The user mode interface of the applications and the operating system kernel functions is called an "environment subsystem", and there are three types of them, the Win32 subsystem, an OS/2 subsystem, and a POSIX subsystem. Unlike the user mode, the Kernel mode has full access to the hardware and system resources of the computer. In this mode, there is a certain level of control the operating system has over the user. The kernel mode can stop user mode services and applications from accessing critical areas of the operating system. User mode processes must ask kernel mode to perform any operations on their behalf. (Sulung, Putra pg 3-4)

Threading

Threading is supported in MacOS. Every single application starts out with a single thread that will run the applications “main” function. The program will then spawn additional threads that all have their own call stack and will be scheduled separately from other threads by the kernel. All threads in an application will share the same space in virtual memory as well as the access rights to resources. Each thread must have memory allocated to it in both the kernel and userspace or else the thread cannot be created. The OS uses the Cocoa framework, which is Apple’s native object-oriented API, for multithreading applications to use locks and semaphores to make sure that all the threads are synchronized and executed correctly. The “NSThread" command will have to be called by the kernel anytime a new thread is spawned so that the OS keeps track of all threads. (Gerbarg 34)

Windows support multiple threads, thread is the entity within a process that is scheduled for execution. Every thread in a process shares a virtual address space and system resources. A process starts with a single thread, but additional threads can be created from the other threads. A thread is able to create a window, and a thread that creates a window also owns all of the associated message queue. So, the thread needs a loop to process all of the messages in the message queue. In order to avoid a deadlock, it is important to use a function like MsgWaitForMultipleObjects for processing the messages. The AttachThreadInput function allows a set of threads to share the same input, this way the threads can activate another thread's window. Threads can also be terminated, if a thread is terminated any resources owned by the thread, such as windows and hooks, are freed. Also termination of a thread causes thread exit code to be set, thread ject is signaled, and if the thread is the only active thread in the process, then the process is also terminated. (Karl Bridge pg 4-5)

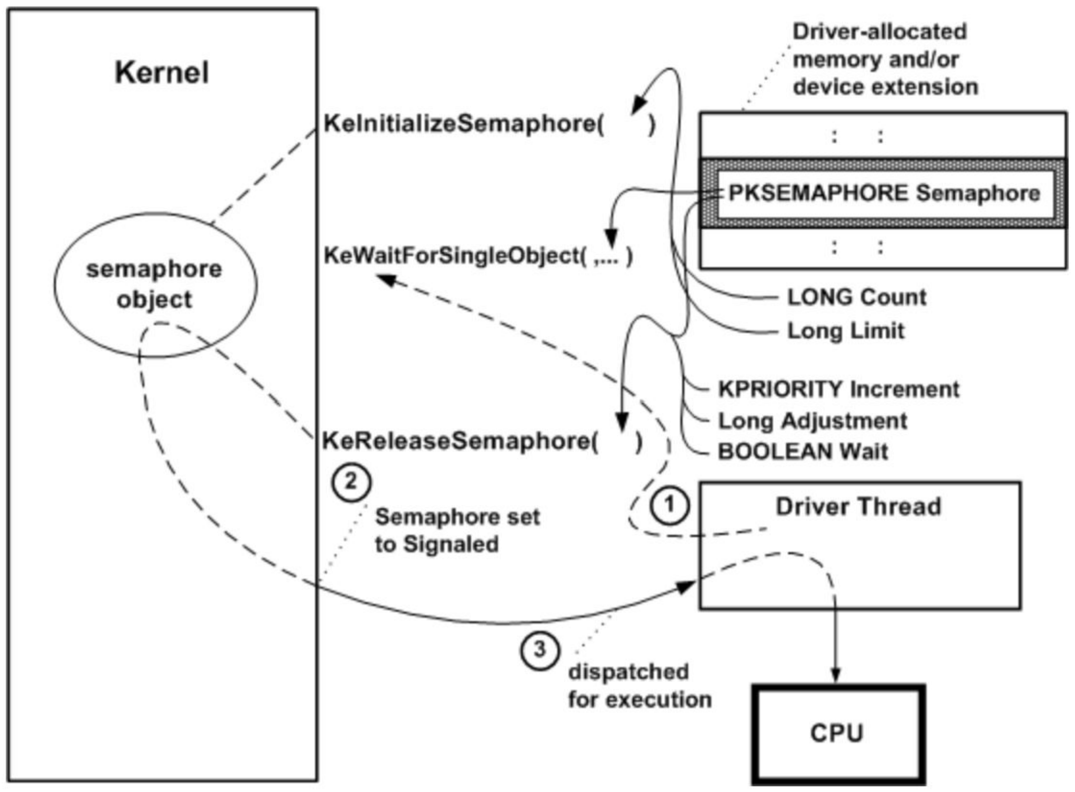
Threading and multithreading are supported in macOS and WIndows. In particular, every single process must start with a single thread and then can split off into many threads depending on the needs of the process. Other similarities are that every fork and child process that is created is recorded in each system and also the threads have their own allocated spaces in virtual address space and their own allocated resources. One differentiation is that Windows doesn't automatically give a thread rights to physical memory until it is referenced as opposed to macOS which will give it to the thread(s) if they need it. Terminating threads are accomplished in Windows by calling libraries instead of native APIs. (Wonglimpiyarat 5)

Synchronization Techniques

Semaphores and locks are utilized in MacOS as synchronization techniques or primitives. Counting semaphores follow Mesa semantics, meaning that a sleeping thread will be ready for execution-only until at least the semaphore becomes positive. The semaphore also is designed to not be decreased by other incoming threads until that recently awoken thread will go for execution. MacOS also uses several types of locks, which allow for a process to not have its critical resources accessed by another thread. (Apple 43)

There are many synchronization techniques for threads, some techniques include: compare and swap, mutual exclusion and threads, semaphores and threads, space location locks, and object locks. For Windows, any driver can be used as a semaphore object inorder to synchronize operations between driver-created threads and other driver routines.

Highest-level drivers run in the context of whichever thread requesting an I/O operation, and a semaphore may be used to protect the resource shared among the dispatch routines. Any driver that uses a semaphore object has to use Kelnitializesemapore before it can release the semaphore. The figure below shows how a driver with a thread can use semaphore objects. (Apple 43)



(Carol 43)

Another technique we will be looking at are spin locks. Spin locks are kernel-defined mechanisms, they are used to protect shared data or resources from simultaneous access. A lot of components use spin locks, for example a driver. A driver can use one or more executive spin locks. Spin locks are classified as a low-level synchronization mechanism which is mainly used for shared memory multiprocessors. When a thread requests a spin lock, which is currently being held up by another thread, then there is a second thread that spins in a loop to test if the lock is available. (Carol Mars pg 4)

Technique name	Description
Spinlock	The system will simply run around in a loop until the thread's time quantum expires and then the next thread will execute. Not very efficient technique.
Mutex	Is placed among the waiting threads and will "sleep" until the finished thread holding the lock wakes it up so that no other process can gain the resource. More efficient than spinlocks.
Read-Write Lock	Multiple readers can view the lock at any given time while only one writer can actually write to it. Efficient for sharing purposes.

(Apple 17)

Windows and macOS also have many of the same synchronization techniques that allow for multiple threads to act with each other in a comprehensive and proper manner so the critical sections are accessed at different times. Both Operating Systems use spinlocks, mutexes, and semaphores as synchronization methods as they are very efficient and rarely lead to the program crashing because the critical section was being used by too many threads. Semaphore objects are created differently in each OS as Windows mostly uses drivers to create them whereas in macOS this is the kernels job to do. There are different function calls for specific synchronization methods in each OS as well but they ultimately behave in the same manner and have the same function. (Wonglimpiyarat 7)

Scheduling Algorithms

The MultiLevel Feedback Queue is the primary scheduling algorithm for the MacOS operating system. This MLFQ algorithm contains several queues, which each have their own Round Robin algorithm, in which threads travel up and down between the four priority levels, shown in the table below, for several reasons. One instance of a thread being demoted to a lower queue is when the set time quantum of that queue is exceeded by that thread. This especially occurs with threads that are CPU-bound and therefore require a lot of computation time. These types of threads are assigned to the lower priority bands so that it doesn't cause starvation for other threads, especially the important I/O bound threads which are generally more important since it requires interaction with the user. (Cowan 55)

Priority Band	Description
Real-time threads	Highest priority threads; the goal is to get a fraction of total clock cycles
Kernel mode only	Allocated for threads made in the kernel and therefore have higher priority than user-space threads
System high priority	Higher priority than normal threads
Normal	Lowest priority threads

In the Windows operating system, threads are scheduled to run based on their scheduling priority. Every thread has a certain priority level, and it ranges from 0 (lowest) to 31 (highest). The OS treats every thread as equal, and the system assigns time slices in a round-robin fashion to all of the threads with highest priority. If none of those threads are ready to run, then the system assigns time slices in a round-robin fashion to the threads with the next highest priority. The priority of each thread is determined by the class of its process and the priority level of the threads within the priority class of its process. (Karl bridge ph 4-5)

Windows and macOS both use the advanced and efficient MLFQ scheduling algorithm. Each queue has its own unique Round Robin scheduling and time quantum and also processes travel up and down between queues depending on many factors. They are also preemptive meaning that if a process from a higher queue comes in then it will preempt a currently executing process from a lower queue. macOS looks for more opportunities than Windows to find a proper mix of CPU-bound and I/O-bound processes. The main difference is the actual priority levels. Windows has 32 priority levels whereas macOS has four large specialized priority bands with several queues in them and the number of queues can be changed depending on how busy the system is. (Cowan 4)

Memory Management

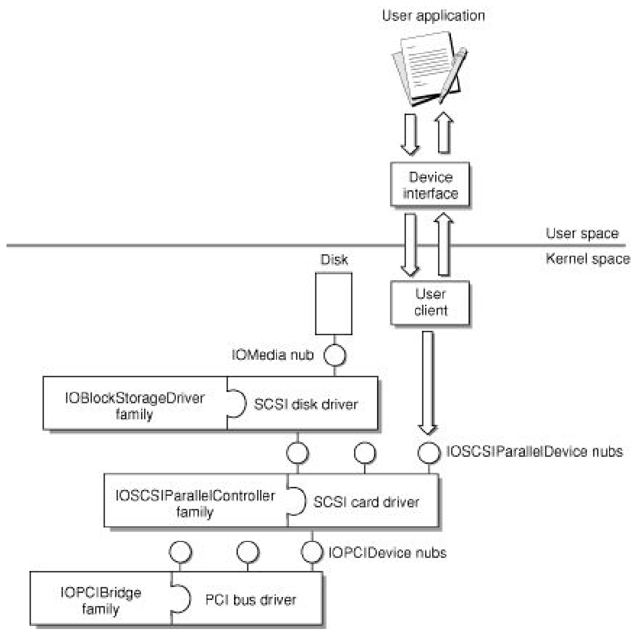
Virtual Memory is used greatly in the MacOS when it comes to the aspect of managing and organizing memory. The VM consists of address spaces and minimizing those contents at the higher level and also consists of memory objects with their own owners and protections at the lower level. If a VM object needs to be duplicated for any reason, a “shadow object” is created which is manipulated and changed by the user or the kernel whereas the original object is kept the same. UPLs, or Universal Page List is a data structure that alters the aspects of pages like its caching, permission, mapping, etc. any time there is a change to a program in the user space like making a Word document private. (Apple 32)

For Windows memory management is taken care by the memory manager, which implements virtual memory and provides a set of services such as memory mapped files, large memory support, and support for cache manager. Each process on 32-bit Microsoft Windows has its own virtual address space that enables addressing up to 4 gigabytes of memory. Each process on 64-bit Windows has a virtual address space of 8 terabytes. All threads of a process can access its virtual address space. However, threads cannot access memory that belongs to another process, which protects a process from being corrupted by another process. (Ted Hudek pg 5)

Like every major Operating System, macOS and Windows both use virtual memory as it expands what you can do in your daily computing session since it appears that you are using more RAM than you currently have. For Windows however, the memory for each process is private and whatever data is in each process cannot be accessed by another process unless it's shared. This can be beneficial so that passwords and other sensitive data isn’t leaked or accessed by another process ensuring that all of the user's precious information is more protected. Although overall, both systems have their own page mapping tables which translates where programs in the virtual address space to the physical address space. (Cowan 6)

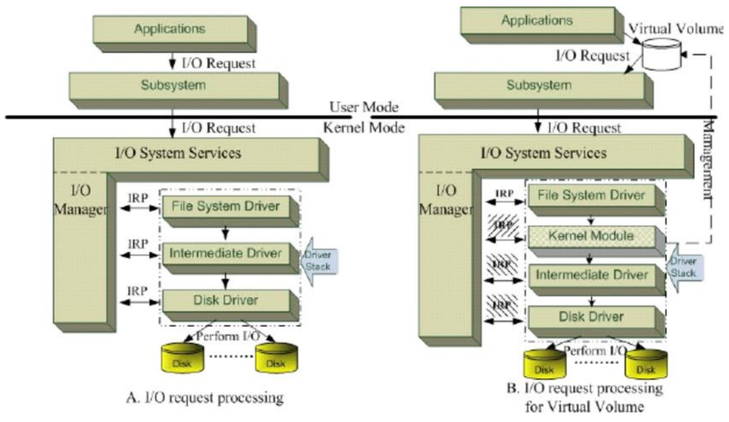
I/O Management

The input/output architecture of the MacOS mimics the hierarchy seen in the actual software. There exists the "family" class containing high-level classification of many devices like the storage families (disks and tapes), protocol families (external drives and flash drives), and the human interface family (mouse and keyboard). This is useful because all the drivers in a family can share the same features like the SCSI controller being able to scan the SCSI bus. A "driver" manages a specific device or bus. Most drivers act in a client-provider symbiosis where the driver must know about all the relevant families that are associated with other drivers so that the communication is sent and received efficiently. There is also the "nub" which acts as a connection point for a driver and also can act as a bridge between two drivers and controls power management and arbitration. (Gerbarg 56)



(Apple 27)

Every operating system has a certain I/O model that is used for handling the flow of data to and from peripheral devices. For Windows kernel-model the I/O manager manages the communication between applications and the device drivers. Since devices operate at speeds that may not match the operating system, communication between the operating system and device drivers are done through I/O requests packets (IRPs). These packets are passed from the operating system to specific drivers and from one river to the next. The Windows I/O system has a layered driver model called stacks, and the IRPs go from one driver to another in the same stack for communication. Windows the I/O model is an asynchronous I/O. Some features of the asynchronous I/O include having a consistent interface to all kernel-mode drivers, including lowest-level, intermediate, and file system drivers. I/O operations are layered.The I/O manager defines a set of standard routines, some required and others optional, that drivers can support. IRPs are used to convey traditional I/O requests, the I/O manager works with the PnP and power manager to snd the IRPs containing PnP power requests. (Tom Chase pg 3)



(Microsoft 45)

Device drivers are used in both of these Operating Systems because they provide the software connection between Input/Output devices and the computer. It is within the kernel's domain to manage communication between the different devices and drivers and this can be overridden by the user at any time. Windows utilizes stacks as a way for communication to occur; a wired mouse would need to talk to a USB hub and then the hub controller and then to the rest of the computer through the PCI bus. macOS has a more specific hierarchy as it ranges from families to drivers and then nubs where they all have their own jobs. (Tom Chase pg 5)

Evaluation

MacOS is beloved by billions of users throughout the world for various reasons. Whether one is a content creator editing videos or one is a developer, it is world-renown software for a reason It is also not a perfect Operating System and has a few flaws of its own that must be noticed.

A great attribute of macOS is that they have built their OS on top of previously existing Unix-like OSs like FreeBSD and Mach and have taken the best qualities out of all of them and have further built upon them. This allows the system to be very safe and secure whenever users are going about the Internet including when downloading anything which results in macOS being less likely to get viruses than its competitors. Another advantage is that it has a layered architecture meaning that the kernel will control much of the central functions in a computer like CPU scheduling whereas the Cocoa layer will control all user applications and app behaviour.

There are also some disadvantages to macOS like the fact that it has four priority bands for its CPU scheduling which isn't good when taking into account how many threads there may be in a computer whereas Windows has 32. macOS is also rather limited overall in comparison to other OSs when it comes to what one can customize and manipulate like files or applications mostly due to its Aqua graphical implementation which is designed to make everything look nice. The I/O management system also is not perfect as everything is separated into categories in which not all devices will automatically communicate with each other unless two drivers are within the same family or the kernel allows them to do so.

Windows is the most popular desktop Operating System in the whole world for a reason. It is preferred by almost all companies and organizations due to its years of efficiency and also its progress of technology in all of these years. And of course it has its own drawbacks which lead to the rise of Apple in the desktop OS market.

Probably the biggest appeal of Microsoft's OS is that it can run on far more desktops and laptops from various other computer companies like HP and Dell as opposed to how macOS can really only run on devices made by Apple. Windows is also known for its multithreading features which give specific rights to threads like memory access rights and also for hyperthreading which is why Windows is used more for workstations and powerhouse desktops. It also has a great I/O management system that allows for devices to communicate with each other and the computer efficiently as possible without too many system blocks in the way.

There are some limitations of the Windows OS such as the fact that it doesn’t use specific synchronization techniques for certain scenarios; it will only use whatever semaphore is easiest to implement which isn't always the best semaphore to use at that moment. Also it doesn’t have the best way of solving deadlocks as it just goes over a deadlock and hopes to remedy it later on which can often lead to the system crashing or freezing. Also there are several different versions of Windows that come out every few years which can lead to some problems especially since older computers can’t update to the latest Windows 10 as opposed to how old MacBooks can still update to the latest patches.

There are many ways in which to expand either OS. For macOS, I would probably make the system more flexible so that it can run various applications and programs which can run on Windows but not on macOS. This would be done by overhauling the Cocoa layer which controls all applications and user interface and making it able to run just about any applications that can run on other OSs. And the best thing that Windows can do is to create more native applications so that users can do more on that OS instead of having to look for some third-party application off the internet. macOS has programs like QuickTime and GarageBand which enhance macOS as a whole and this is what I think Windows needs to do.

References

1)

Wonglimpiyarat, Jarunee. “Technology Strategies and Standard Competition — Comparative Innovation Cases of Apple and Microsoft.” *Elsevier*, Journal of High Technology Management Research, 19 Jan. 2012, www.dl.edi-info.ir/Technology%20strategies%20and%20standard%20competition%20%E2%80%94%20Comparative%20innovation.pdf

2)

Stallings, William. “The Windows Operating System .” *Rossano*, Prentice Hall, 2008, rossano.pro.br/fatec/cursos/soii/Windows.pdf

3)

Solomon, David A. “The Windows NT Kernel Architecture.” *Computer*, Expert Seminars, Oct. 1998, d1wqtxts1xzle7.cloudfront.net/61330850/0072228420191125-35640-cvxwqa.pdf?1574699391=&response-content-disposition=inline%3B+filename%3D00722284_pdf.pdf&Expires=1606610822&Signature=W02oOYrcCFN1OLkiiFh2z1kmLJ6e9UIhFQ4fz2o4zSIxePHFSzVjvmZL9OArGwv~R85ASrUYroCmwQbs~fD5Z3v3bekMNDZ6CNQCbHUoYpmI-Ic6MSwIzddbNU79d~YXU09qsYs4bTUH6ZsguSb344lu2uzagXvBj45xx3dvf8xBc

XSvhlRbuUaF6jT7NTeDqFYjrHDe24Ka1aC4YHbUEJcIIzx3mdjjjXsrGK8RQtPMf6S5AaY3UPq~e~ruZUnlA7kBjifG2BBJArbedq0yG7noUHRCWkIRe36tzSGRu6LDOZFB16HMkKFjxkhLQyUypzoFmbIwVeyr1U0Zm47Bag__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

4)

Cowan, Crispin. “User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems.” *CiteSeerX*, IEEE, 2012, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.679.1398&rep=rep1&type=pdf

5)

Gerbarg, Louis G. “Advanced Synchronization in Mac OS X: Extending Unix to SMP and Real-Time.” *USENIX*, BDSCon, 5 Dec. 2001, www.usenix.org/legacy/event/bsdcon02/full_papers/gerbarg/gerbarg_html/

6)

Case, Andrew, and Ryan D Maggio. “Memory Analysis of MacOS Page Queues.” *Science Direct*, DFRWS USA, July 2020, www.sciencedirect.com/science/article/pii/S2666281720302535.

7)

Sulung, Putra. “Windows Architecture.” Medium, Medium, 21 Aug. 2018, medium.com/@putrasulung2108/windows-architecture-d2b022f136d3.

8)

Karl-Bridge-Microsoft. "Multiple Threads - Win32 Apps." *Win32 Apps | Microsoft Docs*, docs.microsoft.com/en-us/windows/win32/procthread/multiple-threads.

9)

Carol-Mars-Microsoft. "Semaphore Objects" | *Microsoft Docs*, docs.microsoft.com/en-us/windows-hardware/drivers/kernel/semaphore-objects

10)

John-Rice-Microsoft. "Introduction to Spin Locks" | *Microsoft Docs*, docs.microsoft.com/en-us/windows-hardware/drivers/kernel/introduction-to-spin-locks

11)

"Kernel Programming Guide." *Kernel Architecture Overview*, Apple, 8 Aug. 2013, developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgrammi ng/Architecture/Architecture.html.

12)

Karl-Bridge-Microsoft. "Scheduling Priorities - Win32 Apps." Microsoft Docs, 31 May 2018, docs.microsoft.com/en-us/windows/win32/procthread/scheduling-priorities.

13)

Mclean Byron. “About Memory Management - Win32 Apps.” Microsoft Docs, 31 May 2018, docs.microsoft.com/en-us/windows/win32/memory/about-memory-management.

14)

Ted Hudek. “Overview of the Windows I/O Model - Windows Drivers.” *Microsoft Docs*, docs.microsoft.com/en-us/windows-hardware/drivers/kernel/overview-of-the-windows-i-model

15)

Tom Chase. “Windows Kernel-Mode I/O Manager - Windows Drivers.” Microsoft Docs, 17 Oct. 2018,

docs.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-i-o-manager.