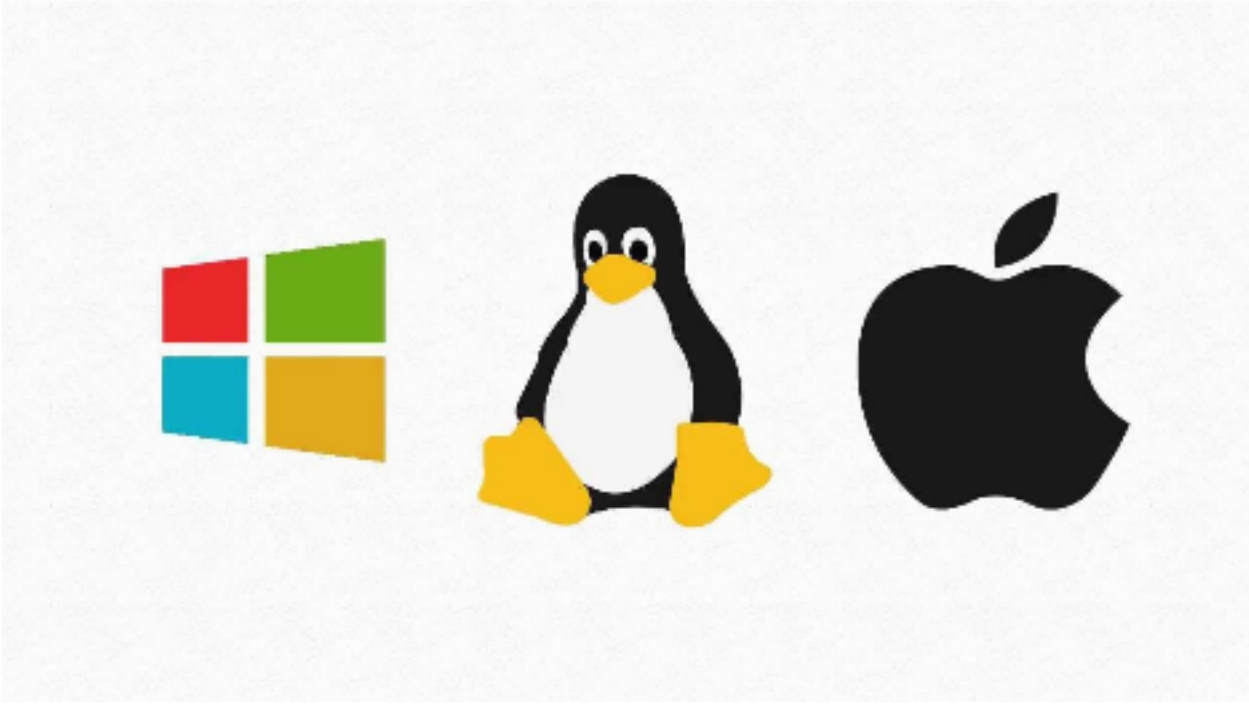


Deep Dive into Operating System Kernels and Evolution

Research Conducted At:

Montana State University
Operating Systems CSCI460 Spring 2020

Authors:

Christian Marquardt
Michael Ressler
Khbindar Arumugam
Eric Kempf

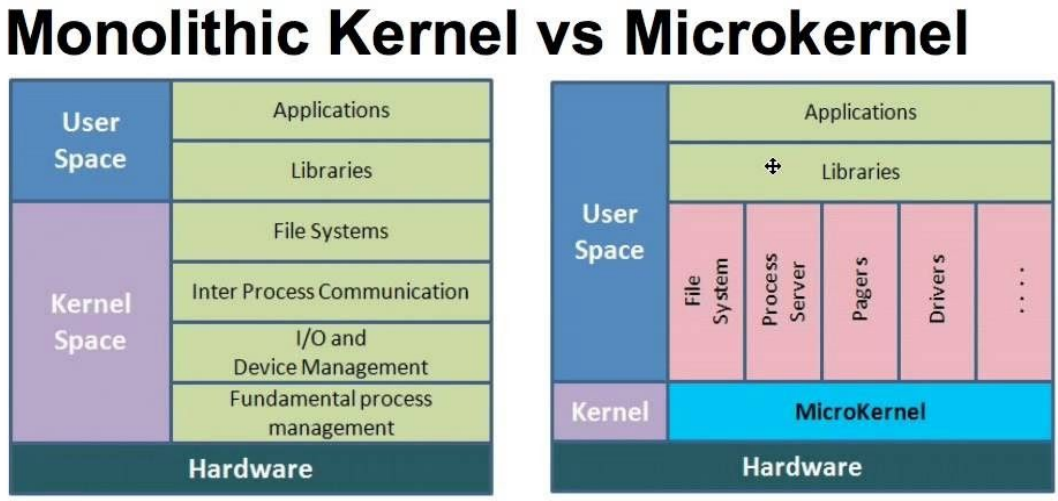
Table of Contents

1. Introduction

In 1991, when Linus Torvalds created the kernel, it revolutionized the technological world. Developers and hobbyists alike were able to have closer interactions between software and hardware components with extreme precision. One of the key aspects when designing an operating system is performance. When it comes to design, certain trade-offs must be made between efficiency, security, accessibility, and much more. Defining these tradeoffs have proven to be difficult as the demand for these features has increased to an unfathomable amount as time has progressed. During this phase, there have been immeasurable amounts of progress built on the initial kernel from private groups to large-scale corporations that have built and designed an operating system with a specific type of kernel in mind to meet their needs. This paper will delve into the kernel design and performance of the three big giants in the computer industry. These giants are Linux, MacOS, and Windows and discover similarities, differences, and key trade-offs of their kernel and how it operates.

1.1 Two Types of Kernels

When defining what a kernel is, it is impossible to hear the words monolithic and micro. These two words are what a kernel can be categorized. The microkernel version focuses on managing system resources with two completely separate address spaces. These address spaces are known as the user and kernel space. An example would be when a user service fails it does not affect anything within the kernel space. This enables the kernel to have a reduced size which ultimately leads to a smaller operating system. A monolithic kernel has a management system that focuses primarily on the application layer of the system and the hardware. [8] Some resources the monolithic kernel provides are management services such as file and memory configurations while also conducting CPU scheduling through syscalls. One of the main drawbacks for the monolithic kernel is that everything is conducted in the same address space. This means if one service fails then the entire system fails.



2. Linux Kernel

2.1 Introduction and History to Linux

Linux is an operating system distribution created around the Linux kernel. During the early 1990s, Linux was developed by Linus Torvalds (Finnish software engineer) and the Free Software Foundation (FSF). The design of Linux was almost identical to MINIX which was a UNIX operating system. Linus was motivated to create Linux because of MINIX's inability to do terminal emulations. In the beginning stages, Linux was created on MINIX using GNU C compilers and had the ability to do terminal emulations. Since Linus knew he could advance the system to the next level, he worked on the system and created multiple versions. The first official version was version 0.02, it was capable of running bash shell which was a traditional program that gave text-only user interface for operating systems that were similar to UNIX. This version also featured GCC (the GNU C compiler). [13] In 1994, version 1.0 was released and this was considered the core of the operating system. Around the same time, Richard Stallman (American software developer) and the FSF were working on creating an open-source operating system named GNU which was similar to the UNIX operating system. The utilities used in their operating system were later included in the Linux kernel in order to complete the system. [14]

Ever since Linux was initially released, many people have made tremendous changes to it. Between 2007 - 2008, the number of Linux kernel source code added was 4,300 lines, removed was 1,800 lines, and modified was 1,500 lines. For the past few years, the rate at which Linux released was 2 ¾ months. Knowing the releasing period of the kernel makes it easier for users to wait for new features and decide on certain things. New releases of the kernel are modified versions of the previous releases. Preferred features from the previous versions are usually reproduced in the new versions. New versions of Linux are created because of evolution. When developers react to stimuli in the world, changes are made to the kernel in order to satisfy the needs of the users. After changes are made to the kernel, developers run the kernel to test its condition. Unit tests are one method used to test if the external features have broken.

Linux operating systems are not produced by one organization or person, since it is open-source software, different organizations/people work on different parts of the system. 25% of these developers are not sponsored nor paid for working on the kernels. This group mostly consists of amateurs (who usually call themselves hobbyists). The remaining 75% of developers are paid and sponsored by companies because the companies want to have a record of what the users want, how the kernel works, and what's the next step/how to grow from here. Linux systems can be assembled by anyone. However, people prefer to purchase a fully assembled system due to the amount of time and work involved in compiling the system. This is where Linux distributions come in, also known as distros of Linux. A Linux distribution is basically when a Linux kernel and a collection of software come together to create an operating system. Linux distribution technically does all the work for their users, making it easier for people to purchase and use

systems that are already compiled. Linux distributions combine all codes from an open-source project to create a single operating system that can be used right away. These distributions decide on the default desktop environment, browser, and other software for their users. Based on what a person wants, the choice of distribution differs. There are hundreds of Linux distributions that focus on specific users or systems. RedHat Enterprise Linux, Fedora, Ubuntu, Debian, and Linux Mint are a few examples of the most popular Linux distributions available. [15]

2.2 What Linux has to Offer

There are several reasons for the importance of Linux:

Free and Open Source

- ➢ Linux is available for everyone and it is free.
- ➢ Linux can be used by businesses to reduce their IT budget and save lots of money.
- ➢ Anyone can contribute, modify, enhance, and distribute the code.
- ➢ People have the freedom and resources to build their own kernel.
- ➢ Users have access to all basic software, educational software, and professional software.

Security and Privacy

- ➢ By installing Linux, viruses, and malware on systems can be prevented. Compared to Windows, Linux does a good job of protecting a system from viruses.
- ➢ Not every user can access system settings, Linux requires a password from the administrator. In order to prevent multiple users from making changes to the system settings and configurations, Linux asks users to be logged in as the root if they want changes to be made to the system.
- ➢ Linux allows users to browse the internet without fearing viruses.
- ➢ Since Linux is open-source software, many developers have amended the source code to reduce the number of flaws.
- ➢ Linux guarantees privacy since most of the user's data is not collected and stored. This is true when using its distributions and software.

Stability

- ➢ Linux is a reliable system that barely crashes.
- ➢ The system does not have to be rebooted often like Windows. Therefore, Linux has a greater number of servers running on the internet.
- ➢ Regardless of the number of years, Linux OS is capable of running fast. It barely slows down.

Revive older computer systems

- ➢ Through Linux, old and outdated computer systems can be used as a firewall, router, backup server or file server, and more.
- ➢ Compared to other operating systems, Linux is the best option to repurpose old hardware.
- ➢ Due to the small footprint of Linux, it is possible to run on older hardware or on embedded systems

Performance, Compatibility, and Flexibility

- ➢ Linux ensures high performance on multiple networks and workstations.
- ➢ A large number of users can use the system at the same time.
- ➢ It works efficiently.
- ➢ Linux is compatible with many file formats.
- ➢ There is flexibility on Linux. If there is only a need to install certain components it is possible.
- ➢ In order to make sure there is no major loss and there is a backup, Linux files can be kept under multiple partitions.

Runs on any hardware

- ➢ Linux works on networking devices, phones, personal computers, and supercomputers.
- ➢ Even though initially Linux was developed on PC hardware using Intel processors, over the years, it has been transferred to more hardware platforms.
- ➢ Depending on specific hardware requirements, users can customize Linux installation.
- ➢ Since users can pick the modules they wish to install, it makes it possible for Linux to be installed on old hardware as well.
- ➢ Linux makes it possible for all hardware resources to be used.
- ➢ A lightweight Linux system can be installed into an old and slow Windows system to enhance the life of the system.

Easy to maintain and use

- ➢ It is easy to maintain the Linux OS because users can update the OS and all software installed.
- ➢ The central software repository is used to update systems and keep them safe. All variants of Linux have their own central software repository.
- ➢ Linux offers regular updates and the system does not have to reboot in order to be updated.
- ➢ The update can be done automatically or periodically by a few clicks.
- ➢ When compared to a Windows system update, a Linux system update is easier. All third-party software doesn't have to be updated individually.
- ➢ Linux is user-friendly and has a good graphical user interface (GUI). Almost every functionality on Windows can be found in Linux.

- ➢ The graphical user interface has been developed to accommodate typical users' wants. Users can do anything they want even without them knowing any commands.
- ➢ If there is a need to run applications that work only on Windows, Windows emulator Wine can be installed to make those applications run on a Linux system.

Customization

- ➢ A key feature that differentiates Linux from other operating systems is customization.
- ➢ Users have the flexibility to customize their system based on their preferences and requirements. Features can be added or deleted depending on their need since it is an open-source operating system.
- ➢ There are plenty of wallpapers, desktop icons, and panel options to choose from. Wallpapers and icon themes can be installed in the system.
- ➢ A command-line interface with many shells is offered by the Linux OS. In order to automate routine maintenance and other tasks, systems administrators can use a powerful command-line interface and write shell scripts.

Software updates and Various Distributions

- ➢ Even though Linux goes through a huge number of software updates, these software updates don't take as much time to update as other operating systems. Updates in Linux are easier and can be done without any concerns.
- ➢ Since there are many Linux distributions available, it gives the users a range of options to choose from.
    - For beginners, Ubuntu or Linux Mint is recommended and for good programmers, Debian or Fedora is suggested.
- ➢ Linux distributions combine all codes from an open-source project to create a single operating system that can be used right away.
- ➢ Some distributions of Linux are Fedora, Ubuntu, Arch Linux, Debian, Linux Mint, and many more.

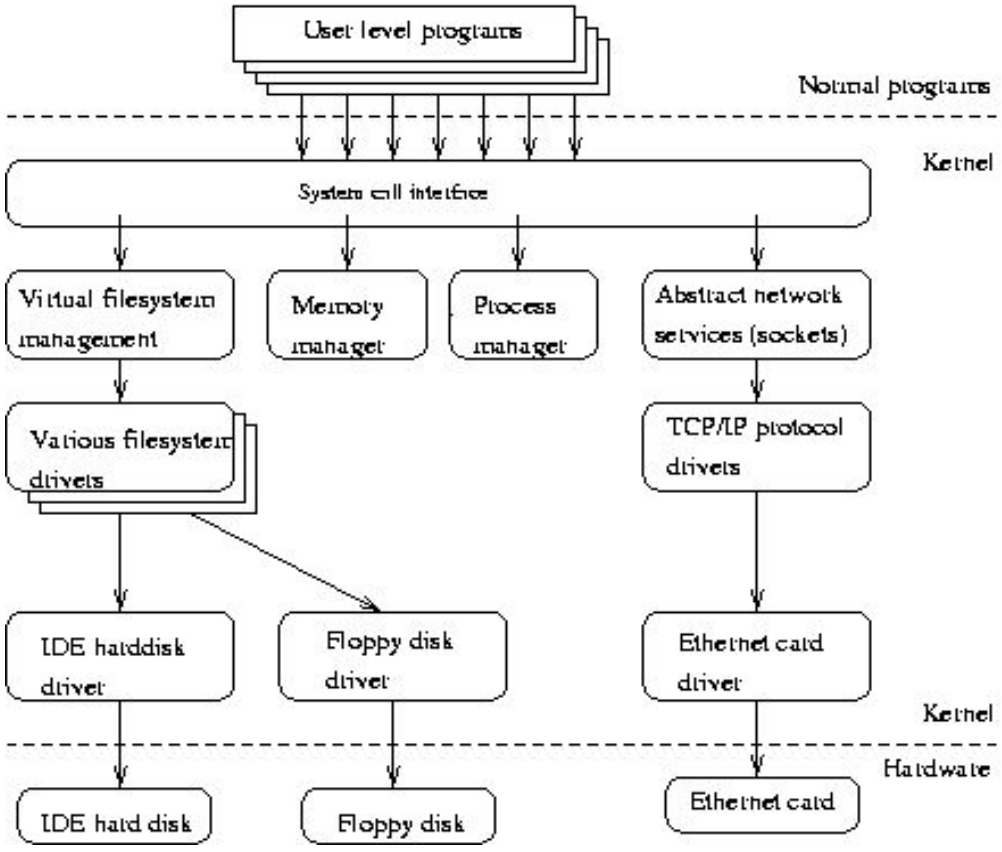
Education

- ➢ Linux can be used by students to learn:
    - How the software works, before modifying and extending the code to suit their needs.
    - The internals of an OS and the software.
- ➢ Teaching students about software can generate new software and help innovation depending on local needs.
- ➢ Linux does not have to be for programmers only, users can contribute to Linux by assisting in documentation, translation, and testing.
- ➢ Since it is free, it can be a great educational tool for schools and colleges to teach students. [16][17][18][19]

2.3 General structure/design

Linux is a 'Unix-style' operating system, so we know it's kernel is monolithic by design. This essentially details that the device drivers operate in the kernel-space rather than the user-space as in microkernels. The Linux kernel is built up of a set of subsystems seen in *Figure 1* below such as memory management, process scheduling, network, and so forth. The kernel contains many tools for system and application programs to use, where this keeps the users from accessing the hardware directly. We can see this in *Figure 1* where user-level programs must go through the kernel first by using system call interfaces. Some of these system calls we have seen before from our previous programming assignments using exec() and fork() calls.

Figure 1:



Source [1]

The memory management subsystem of a Linux kernel assigns memory area and swaps space areas to processes. Linux also supports virtual memory allowing usable memory to grow. Another important subsystem would be process management which handles processes and implements multitasking by switching a process on the processor. The kernel also contains its own set of threads used to concurrently run kernel tasks and only exist in the kernel space. [2] Looking at independent processes, when one is created, the syscall will return to the next instruction for both the parent and child process. This allows an execve() syscall to replace that

current program with a new one with a new stack, heap, and data allocation. This allows for a new program to run. Processes run with privileges determining its capabilities, i.e. admin (0) can have a lot more access.

The Linux kernel is preemptive, which wasn't the case in kernel version 2.5.4 and older. In these older versions, the scheduler couldn't suspend a task even with a lower priority if it was in kernel mode. However, with preemption, the kernel has the ability to context switch. This allows tasks to be preempted for higher-priority tasks when needed, by suspending the current task on the processor. This prevents tasks from taking up resources from others, and makes the kernel decide when to interrupt/suspend a task. Having a preemptive kernel is a key difference between Linux and Unix kernels.

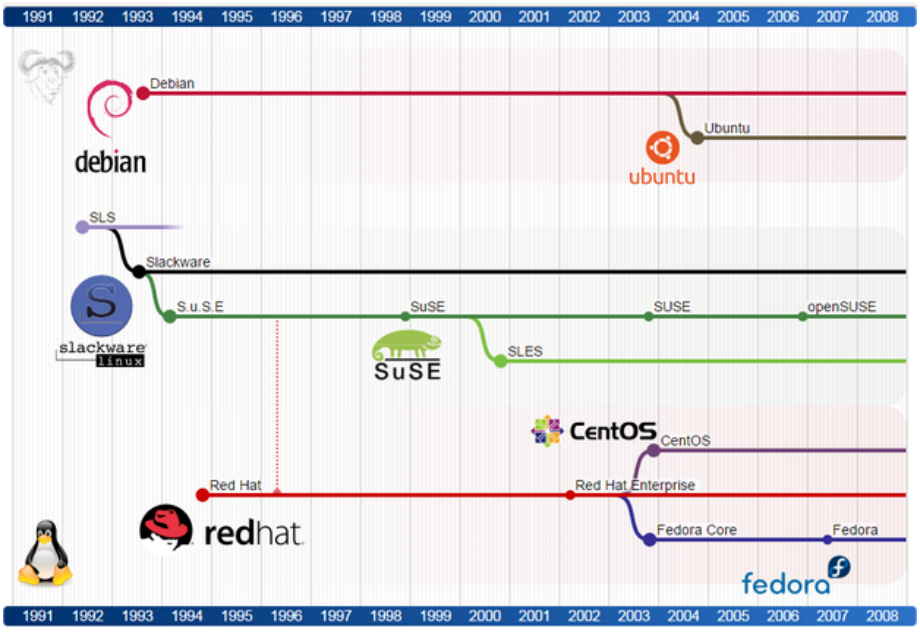
Looking at the Linux Kernel's process scheduler, it is designed to use scheduling classes that are modular with a "Completely Fair Scheduler" (CFS) as the default class. A CFS algorithm uses a red-black tree as its data structure. This implementation allows tasks to get a fair share of time to the processor. Other algorithms are implemented as these classes, each designed for certain parameters. For instance, the "First-In-First-Out" scheduling algorithm policy is designed for special time-critical applications. Most recently, the "earliest deadline first" algorithm class has the highest precedence over their implemented scheduling classes. [3]

2.4 Linux Distributions

The outcome of Linux being open-source software has led to the rise of distributions tuned to the users' needs. Some of the well-established "distros" include Debian, Ubuntu, Mint, Red Hat, CentOS, Fedora, and many more. All these distros are based on the Linux kernel, however, differ from small modifications and what kernel version they use. For instance, a distro may choose an older Linux kernel version to modify, since newer releases can be more unstable. Users looking for stability over new features may favor a distro like this one described.

There are a plethora of distros to choose from, each tailored to the users' needs and wants. For instance, Ubuntu is a very popular choice among Linux distros, for it is designed for beginners and users transitioning to a Linux operating system. Many distros are community-driven and can be derived from other distros. For example, the Linux Mint distro is based on Ubuntu and has made modifications to their communities' wants. *Figure 2* below demonstrates a basic timeline of some popular Linux distributions and how other distros are created from modifications to it's parent distro.

Figure 2:



Source [20]

3. MacOS Kernel

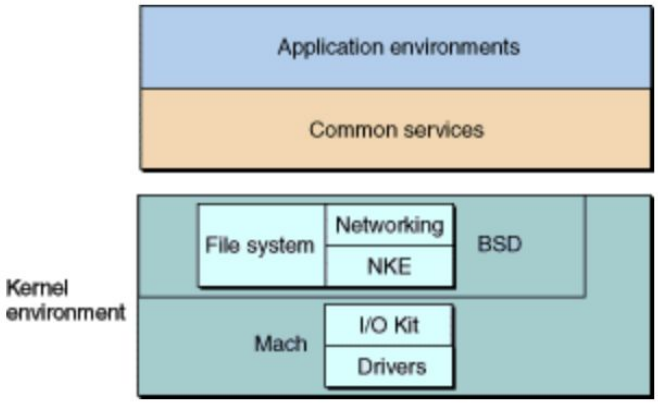
The MacOS kernel first got its start with Steve Jobs and with the computer company known as NeXT. They were able to use and develop the Carnegie Mellon University's Mach kernel that Apple bought in 1985. It is known as XNU, which stands for “XNU is Not Linux.” This consisted of C++ components for distinct drivers and the Mach kernel with FreeBSD operating system features. All of these features are encapsulated and improved upon under the operating system known as Darwin. It is the same name Apple uses to this day for their kernel. [4]

3.1 Hybrid Kernel

Apple had the goal of approaching their kernel with a balance between a monolithic and microkernel. This type of kernel is created by having a separation between subsystems such as the kernel itself, distinct device drivers, and an initial near-minimum amount of software to provide protection between critical areas of the kernel and the user-space. This would later fall into the category of a hybrid kernel. [5] These types of kernels are designed to provide a healthy medium of flexibility that a micro-kernel has, and the performance a monolithic kernel provides. The idea of flexibility comes from the idea that when one part of the kernel stops, another part can start it again. It has been disputed whether having a hybrid kernel is a more stable approach to the design of the kernel compared to its Linux counterpart.

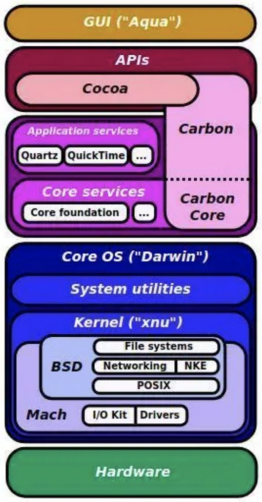
3.2 Mach and BSD Architecture

To understand how the kernel works, we must take a deeper look into its environment. As stated earlier, the MacOS kernel environment consists of two major layers known as Mach and BSD. Mach is a microkernel that focuses on managing processor resources such as CPU usage and memory protection and passes all of that to other OS layers. Some notable components in a Mach microkernel are IPCs (untyped interprocess communications), RPC (remote procedure calls), and the support of pagers. As said, the kernel in Darwin's operating system is not just a microkernel as it also has unique characteristics that a monolithic kernel has. This layer is called FreeBSD, which branched from its predecessor BSD (Berkeley Software Distribution). FreeBSD provides the identity for the OS by providing services such as syscalls, many POSIX components such as pthreads. [9] These layers work together to perform crucial operations such as helping a chosen API to match a specific portion that the kernel needs in order to function efficiently and properly.



3.3 Not Completely Closed-Source?

While many assume that the operating system MacOS uses has always been closed-source it has been open since creation. Many other portions under Darwin too are open-source such as the coding language developed by Apple called Swift, their command-line toolkit used by developers, and even the rendering engine behind Safari. The common misconception behind this is because API's more, notably the UI API's are under a closed source license that is within the Carbon and Cocoa APIs such as CoreServices. Other applications such as Quicktime, Quartz, and Metal too are licensed as closed-source. [5] The idea behind this is that if you were to make these features open-source it could lead to loopholes and malicious attempts on the software which could potentially exploit many if not every IOS or MacOS user. This also helps Apple profit



substantially since it is under their license and cannot be changed or improved upon by any developer outside of Apple.

3.4: Security Features

With MacOS being as closed-source as it is, it has had the reputation of being relatively secure by default. This functionality is developed with internal software features that focus on getting in the way of free and open-source applications loading up on the MacOS machine that could come with potential malware in hopes of exploiting secure and critical areas. Security has coverage from operations within the operating system to the boot-up process. While it may seem like the MacOS is an impenetrable object when it comes to malicious attempts, there have been successful attacks on the OS. These types of attacks are done by downloading third-party applications or fraudulent emails. One of the most detrimental attacks on the MacOS entailed an application known as MacDownloader, which stated there were viruses on your computer and they needed a scan and removal. [7] It originally had the goal of attacking the US defense industry and hid under an Adobe Flash update. Users unknown to this malware would knowingly enter their admin password which in turn lets the software lift any sensitive data that was on the machine and send them out. Apple would later implement a cybersecurity feature known as System Integrity Protection in which the kernel would enforce certain parameters with the goal of protecting processes even with root privileged execution.

3.5: The Future and Accessibility Concerns

When MacOS Catalina was out, and the software updated, Apple declared that in the future the removal of kernel extensions would come into fruition and applications would need re-evaluation if they wanted to continue to work in the MacOS environment. Kernel extensions provide applications the ability to reach into kernel space or also known as plug-ins. This part removal is in hopes of increasing optimization and security for the machine. While this change is only one instance and sounds like a much-needed update, flexibility on the user-end has increasingly become smaller. Developers alike will have less area to develop software that utilizes the kernel and separates them farther from the kernel than they have ever before. While this setback can either be seen as good or bad, it is important to understand both sides and the tradeoffs. To increase accessibility there have been 3rd party projects that have branched from Darwin such as one named PureDarwin, which has the goal of making Darwin have increased use and the ability to add more software than its predecessor. [6]

4. Windows Kernel

4.1 A brief history of Windows development

The first version of Windows was released in 1985. Before this, Microsoft had developed a disk operating system known as MS-DOS, but users had to interact with it via the command prompt. Originally Windows acted as a graphical user interface for MS-DOS, allowing the user to interact with the operating system using a mouse and opening actual windows. By 1993 Microsoft had released the Windows NT kernel. While originally marketed towards businesses as a high-performance alternative to what was already on the market, NT would lay the foundation for future development of the Windows kernel.

The release of Windows NT came with support for many virtualization features. Most importantly, it split the virtual memory system into four different layers, utilizing a hybrid kernel architecture, unlike the monolithic design that Linux uses. These layers consisted of the kernel, executive, supervisor, and user layers. This change allowed for duties that normally fell onto the kernel to be split up among the others. The kernel layer was now primarily in charge of keeping track of synchronous threads and processes, in addition to lower-level machine functions such as interrupt service routines (ISRs). This meant things such as IO and virtual memory management (objects, heaps, handles, etc.) were pushed onto the executive layer's list of responsibilities. Another important element introduced was the scheduler. The scheduler's main function was to assign priority to various threads in the system to determine which should be allowed to use the processor's resources at a given time. However, at this time the scheduler could only handle one queue of threads per processor which turned out to be a bottleneck that was addressed in later versions of Windows.

Windows 7 introduced the Dynamic Fair Share Scheduler (similar to the Completely Fair Scheduler on Linux) to the OS. Mainly designed for terminals, this scheduler improved upon the ability to judge how much computing power a process would use and made sure a single large workload didn't hog resources that other threads needed. In addition, Windows 7 also added functionality to support more than 64 processors at once. It did this by introducing the concept of processor groups, which create a block of processors, allowing them to be scheduled as a single cohesive unit. However, there was still a bottleneck in the form of the single dispatcher database lock, which was given to a single thread at a time, limiting the efficiency of the processor group system. Eventually, this was fixed by changing to a per object lock-based system, allowing for more processes to run at once. This led to performance boosts of up to 290% in some cases and is still considered to be one of the biggest performance boosts ever seen on Windows from the implementation of a single feature.[10] Windows 10 went on to improve the concept of processor groups even further with the addition of CPU sets, which allow a process to take control of a group of processors at the same time, preventing any other process from accessing the resources there.

4.2 Security on Windows

It is often said that Windows is less secure or stable than MacOS or Linux. While this may be true in some aspects (third party drivers, etc.), it is not true when we are talking about the kernel itself. Microsoft is constantly pushing out updates and patches focused on security on their OS. Just earlier this year they released a form of kernel data protection (KDP), which relies on virtualization to protect data on the kernel. It does this by splitting up the kernel into two sections, a normal kernel known as VTL0, and a more secure kernel known as VTL1.[12] By restricting memory access between the two sub kernels and setting sections of the memory to be read-only, KDP effectively limits the amount of readily available data to outside sources.

Microsoft is also developing a security feature they are calling hardware-enforced stack protection. This technology utilizes advances in newer Intel CPUs to create something known as a “shadow stack”. This is essentially a copy of a program’s stack, which the CPU runs through without actually running the program. When the program runs, it can look for differences between the shadow stack and normal program to see if any malicious code is trying to execute. Although this technology has not yet been pushed out to the main versions of Windows, Microsoft claims it should be able to detect and prevent common attacks on the kernel, such as buffer overflows and null pointers. These security features illustrate how much time is put into keeping Windows secure. They also show what is to come in the future for the Windows kernel in regards to security features.

4.3 Comparing Windows to other kernels

When comparing the Windows kernel to something like UNIX, we can see many differences in how they got their start. For example, Windows was built using a 32-bit program address space with support for multiprocessors, whereas UNIX used a 16-bit program address space with a uniprocessor design. Another key difference is that Windows uses the same kernel for a multitude of different architectures, such as x86, x84, ARM, and ARM64. This allows the operating system to be very versatile and run on many different types of devices with different specifications for each. It does this through the use of refactoring, or essentially modifying the dynamically linked library (DLL) on a per case basis, allowing it to scale it’s resources dynamically to whatever hardware is available. Windows takes this a step further with the use of API sets, which allow the use of a DLL to be separated from where it is actually located. This allows for the sharing of DLLs for different devices without breaking the functionality of the DLL.

This emphasis on versatility and reuse of code is a key difference we see when comparing Windows to MacOS and Linux. MacOS is designed to run on a select number of devices, all of which are developed by Apple, so there is little need for them to implement it in a way that allows for code sharing. Since Linux is open source, it gets its versatility from the creation and

release of new distributions. What makes Windows unique is its ability to use the same kernel DLLs between vastly different architectures.

5. Conclusion

The technology that operating systems and specifically kernels employ has come a long way from where it started three decades ago. The debate over which OS is the best OS has raged on ever since operating systems were created. The truth is, in today's day and age you can't go wrong with an operating system. MacOS has the benefit of using hardware and software designed together for synergy. Windows gains its power from the reuse of kernel code in many different implementations. Linux allows its users to choose from a vast array of distributions for whatever suits the user best. All three operating systems have made massive improvements throughout their evolution, in terms of performance as well as security.

For this project we sought to investigate further into the inner workings of the most popular operating systems in use today. We examined how they got their start, and how early design choices led them down the path to where they are now. We compared these choices between the kernels, and though we didn't make an ultimate decision on which operating system is the best, we feel there is enough information here for the reader to make an educated decision for themselves. Whether that is to go for a hybrid kernel like Windows and MacOs, or take the dive into a monolithic kernel like Linux is up to you.

Resources

Linux Structure Resources

[1]: Linux kernel overview image
Wirzenius, Lars. “2.2. Important Parts of the Kernel.” *Important Parts of the Kernel*, 2004, tldp.org/LDP/sag/html/kernel-parts.html.

[2]: Linux kernel threads
Corbet. "Kernel threads made easy [LWN.net]". Jan 6, 2004. https://lwn.net/Articles/65178/

[3]: Deadline Task Scheduling - Linux
The kernel development community. “Deadline Task Scheduling.” *Deadline Task Scheduling - The Linux Kernel Documentation*, www.kernel.org/doc/html/latest/scheduler/sched-deadline.html.

[20]: Linux Distributions timeline image
Wordpress. “Linux Distributions.” *Cognitive Waves*, 27 Mar. 2020, cognitivewaves.wordpress.com/linux-distributions/.

MacOS Resources

[4]: Darwin History
Scaife, Jamie. *A Look at PureDarwin - an OS Based on the Open Source Core of MacOS*, 27 Nov. 2019, www.jamieweb.net/blog/a-look-at-puredarwin/.

[5]: Differences into Linux and Mac Kernel
William, and Mahesh. “Linux Kernel Vs. Mac Kernel.” *Linux Tutorials, FOSS Reviews, Security News*, 6 Dec. 2019, www.linuxandubuntu.com/home/difference-between-linux-kernel-mac-kernel.

[6]: PureDarwin Project
PD-Devs, PureDarwin. *PureDarwin*, 2017, www.puredarwin.org/.

[7]: MacOS Kernel Security Features
Stavniychuk, Darina. “Popular Mac Viruses, Malware and Security Flaws.” *MacPaw*, MacPaw, 11 Jan. 2019, macpaw.com/how-to/known-mac-viruses-malware-security-flaws.

[8]: Deep Dive into Hybrid Kernels
Dude, Posted bySome. “Monolithic Kernels, Microkernels, and Everything In Between.” *Some Dude Says*, 15 Feb. 2020, somedudesays.com/2020/02/monolithic-kernels-microkernels-and-everything-in-between/.

[9]: MacOS Kernel Architecture and Mach/BSD
Apple, Developer. “Kernel Programming Guide.” *Kernel Architecture Overview*, 8 Aug. 2013, developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/Architecture/Architecture.html.

Windows Resources

[10]: Windows Kernel
Pulapaka, H. (2020, October 25). One Windows Kernel. Retrieved November 13, 2020, from https://techcommunity.microsoft.com/t5/windows-kernel-internals/one-windows-kernel/ba-p/267142

[11]: Linux vs Windows
Lithmee. (2018, November 26). What is the Difference Between Windows Kernel and Linux Kernel. Retrieved November 14, 2020, from https://pediaa.com/what-is-the-difference-between-windows-kernel-and-linux-kernel/

[12]: Windows Security
Introducing Kernel Data Protection, a new platform security technology for preventing data corruption. (2020, July 08). Retrieved November 14, 2020, from https://www.microsoft.com/security/blog/2020/07/08/introducing-kernel-data-protection-a-new-platform-security-technology-for-preventing-data-corruption/

Linux History Resources

[13]: History of Linux
History of Linux - Complete History of the Linux Operating System, history-computer.com/ModernComputer/Software/Linux.html.

[14]: Linux
Encyclopædia Britannica, Encyclopædia Britannica, Inc., www.britannica.com/technology/Linux.

[15]: Linux Distro
Hoffman, Chris. "What Is a Linux Distro, and How Are They Different from One Another?" How, How-To Geek, 23 Sept. 2016, www.howtogeek.com/132624/htg-explains-whats-a-linux-distro-and-how-are-they-different/.

[16]: Advantages of Linux
"Advantage of Linux: Top 18 Important Advantages Of Linux." EDUCBA, 5 Aug. 2020, www.educba.com/advantage-of-linux/.

[17]: Reasons to use Linux
S Sathyanarayanan, et al. "Ten Reasons Why We Should Use Linux." Open Source For You, 31 Mar. 2020, www.opensourceforu.com/2020/03/reasons-to-use-linux/.

[18]: History of Linux
DigitalOcean. "History of Linux." *DigitalOcean*, DigitalOcean, 12 Nov. 2020, www.digitalocean.com/community/tutorials/brief-history-of-linux.

[19]: Background about Linux
"What Is Linux? The History and Background of the World's Most Popular Open Source Operating System." *Linux Training Academy*, www.linuxtrainingacademy.com/what-is-linux/.