OS Comparisons & Beyond

Benjamin Brewster

What We'll Cover

- A brief history of some of the major Operating Systems that make up our current landscape
- Compare and contrast a few Operating Systems
- Even wax a bit philosophical about the past and future

Short History of DOS - Business-Driven

- The setup: Bill Gates and Paul Allen were writing and selling Microsoft BASIC, a language for development, to early computer hobbyists
- IBM was just finishing up prototypes for their new PC and approached Microsoft to see if they had an OS suitable for the hardware in ~12/80
- Microsoft didn't have an OS, so they quickly licensed 86-DOS from Seattle Computer Products (written by 24-year old Tim Paterson)
- Microsoft presented 86-DOS as "Microsoft DOS" to IBM, who accepted it; after a re-write, IBM and MS jointly retained copyright
- Microsoft purchases the rights to 86-DOS in 7/81 from SCP; SCP later sues, claiming Microsoft hid their deal with IBM to get it cheap

Short History of DOS - Business Driven

- IBM begins sales of its PCs with PC-DOS in 8/81
- Microsoft, however, begins sales of MS-DOS (the same thing as PC-DOS without IBM's specific drivers for its own PC) to other OEMs
- Eventually, Microsoft gains exclusive rights to DOS, and sells to all OEMS, securing their place in the market
- The last version of MS-DOS was 6.0, released 3/93 to huge sales and success

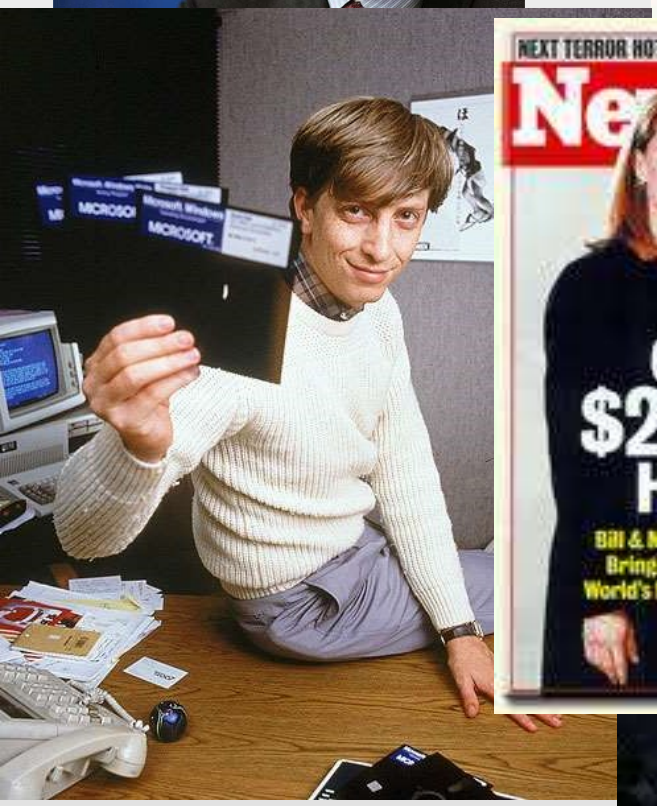
Short History of Windows - To The Future

- Windows 1.0 was released November 20, 1985, and was not well received - it was mainly an overlay over DOS
  - Licensing requirements for Microsoft, which had created applications for Apple, enforced limits: windows could not overlap on screen!
  - Included Paint, Write, a terminal, MS-DOS, and Reversi, among others
- Apple and Microsoft began legal battles with Windows 2.03 and 3.0; the judge dropped all but 10 of Apple's 189 claims of infringement, and most of the 10 left were over uncopyrightable ideas

Short History of Windows - To The Future

- Windows 3 (released May 1990) was wildly successful, selling around 10 million copies even before 3.1 came out; support ended in 2001
- Support for 32-bit software began with Windows 95, released 8/24/95
- Followed by: OS/2, NT, 98, 2000, ME, XP, Server 2003, XP 64, Home Server, Vista, Server 2008, 7, Server 2008 R2, Home Server 2011, Thin PC, 8, Server 2012, 10, Server 2016

Short History of Windows - The Wealthiest



Given away more than 33 Billion today, more than anyone in history

- Forbes list of billionaires in 2016:
  1. Bill Gates, 75 B
  2. Amancio Ortega, 67 B
  3. Warren Buffett, 60 B
  4. Carlos Slim, 50 B
  5. Jeff Bezos, 45 B
  6. Mark Zuckerberg, 44 B
  7. Larry Ellison, 43 B
  8. Michael Bloomberg, 40 B
  9. Charles Koch, 39 B
  10. David Koch, 39 B

Short History of macOS - Inspired

- Steve's Wozniak and Jobs found Apply Computer Inc. on 1/3/77, immediately beginning to make computers of Wozniak's design
- Apple I sells 200 units, before Apple is founded, but the Apple II, launched in April 1977 goes on to sell millions well into the 1980s
- Apple IPO on 12/12/80 generates more capital than any IPO since Ford in 1956, instantly creates more millionaires than any company in history
- Apple visits Xerox PARC, comes back with ideas for Apple's first GUI-based computer, the Lisa, which introduces the words mouse, desktop, and icon; Lisa costs $10K in 1983 ($23.8K in 2016) and fails

Short History of macOS - Inspired

- Subsequent computers are successful running early versions of the Mac OS, though Jobs and Wozniak both leave in 1985
- Jobs founds NeXT, which produces the UNIX-based NeXTstep OS on the computers it sells; this OS uses the Mach 2.5 kernel and subsystems from BSD 4.3
- In 1996, Apple buys NeXT, as a result of Jobs influence, beating out rival BeOS, which sees Jobs return as CEO to Apple!
- Microsoft donates $150 million to Apple to call an end to the decades-long legal war between the companies
- NeXTstep becomes the foundation of OSX, which launches in 2001 as version 10.0 “Cheetah”
- OSX morphs into macOS version 10.12 “Sierra” in 2016

Short History of iOS - Revolutionary

- When Jobs returns to Apple, he brings with him the WebObjects Application server from NeXT, which turns into the Apple Store
- Internally at Apple, OS X is forked into iOS: it's still Mach and BSD underneath it all
- Capacitive sensors are added to the screen, and an entirely new user paradigm is created: touch, hiding the file system, and a simple web browser
- iOS version 1 (initially called "iPhone OS") launches with the iPhone in January 2007; App Store doesn't launch until July 2008
- Apple unit sales in 2016 across their three major divisions:
  - iPhone: 211.88 million
  - iPad: 45.59 million
  - Mac: 18.48 million

Mobile devices (iPhone, iPad) make up 93% of Apple's unit sales in 2016

Sales figures: https://www.statista.com/statistics/382302/unit-sales-of-apple-by-product-category

Short History of Android - Playing Catch-Up

- Android, Inc. was founded in Palo Alto in October 2003
- Initial smartphone plans were to compete with Symbian and Windows Mobile using a custom fork of Linux
- In June 2005, Google purchased Android, Inc. for at least $50 million
- The earliest prototypes of Android resembled BlackBerry's OS with a full QWERTY keyboard layout; after Apple's announcement and rollout of the iPhone in 2007, those visual layouts and keyboard plans were scrapped
- The first phone with Android was the HTC Dream, released on 10/22/2008

Short History of Android - Victory

- Fast forward to today, Android is not just the dominant mobile OS, it's the dominant OS in all sales of devices worldwide
- In 2015 sales, according to Gartner Research, Android's market share is unrivaled:
  - **Android:** 1.3 billion devices (54%)
  - **iOS/OS X combined:** 297 million devices (12.3%)
  - **Windows:** 283 million (11.7%)
  - **All others:** ~520 million (21.6%)

OS Internal Architecture - Booting with BIOS

- Before we can talk about the procedures being followed in the kernel, we need to understand the first stages of how modern OSs boot

- In ye olden days (pre-2014) a switched-on computer would first load the **BIOS** (Basic Input and Output System), which:
  - Performed hardware initialization, including testing
  - Loaded a full-featured boot loader, which may simply load an OS itself from mass memory device, or provide the user a selection of OSs to boot from
  - Provided an abstracted, consistent method of getting data to and from input and output devices

OS Internal Architecture - Booting with UEFI

- Modern PCs, when switched on, now first load the **UEFI** (Unified Extensible Firmware Interface) firmware
- The UEFI is a mini-OS that provides many more features:
  - Boot from large disks (up to 8 ZiB) with a modern GUID Partition Table (GPT)
  - Full-featured pre-boot environment, including a mouse and keyboard-driven GUI, audio, network access, hardware testing, and IT management
  - Booting the OS of your choice from all registered systems
  - Optionally, preventing unsigned drivers from being used in the booting of an OS, which prevents firmware-level rootkits (and possibly locking hardware to a specific OS)

OS Internal Architecture - Booting with UEFI

- Modern PCs, when switched on, now first load the **UEFI** (Unified Extensible Firmware Interface) firmware
- The UEFI is a mini-OS that provides many more features:
  - Boot from large disks (up to 8 ZiB) with a modern GUID Partition Table (GPT)
  - Full-featured pre-[illegible]vironment, including a mouse and keyboard-driven [illegible]ement

One megabyte	= 1,000,000 bytes	= 1000^2 bytes
One gigabyte	= 1,000,000,000 bytes	= 1000^3 bytes
One terabyte	= 1,000,000,000,000 bytes	= 1000^4 bytes
One petabyte	= 1,000,000,000,000,000 bytes	= 1000^5 bytes
One exabyte	= 1,000,000,000,000,000,000 bytes	= 1000^6 bytes
One zettabyte	= 1,000,000,000,000,000,000,000 bytes	= 1000^7 bytes
	= 1,000,000,000,000 gigabytes	
One zebibyte	= 1,180,591,620,717,411,303,424 bytes	= 1024^7 bytes

[illegible]e booting of an [illegible]ing hardware to

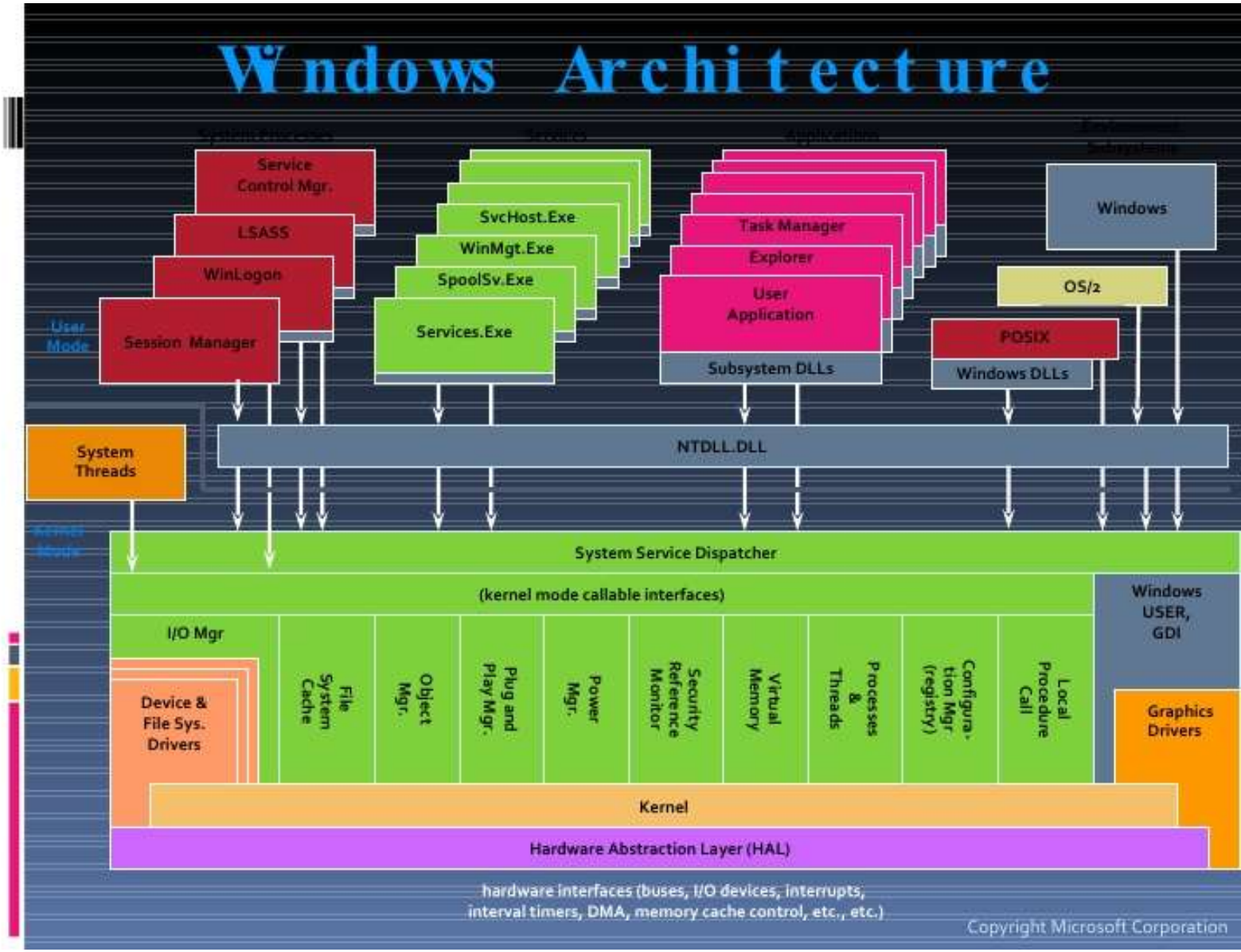
Windows Internal Architecture

- User programs can access the OS by using the Windows API, of which there are many versions: Win32, Win64, etc.; backwards compatibility is a major focus, allowing many of these APIs to be accessible at the same time
- The user has deep access to low-level aspects of the GUI, which can cause simple programs to be large if API libraries such as .NET or DirectX are not used
- A large set of services run on top of the kernel providing access to various features of Windows to user programs

Windows Internal Architecture

- The Registry: a hierarchical database of configuration settings for Windows and Applications
  - Used by the OS itself (kernel), drivers, services (what Windows calls daemons), the security system (SAM), and UI
  - Applications can use the registry if they wish, but could also use standard configuration files in the file system
- Security Account Manager
  - An encrypted database file that stores the hashed passwords and local accounts
  - Cannot be messed with while Windows is running (though in-memory passwords can be dumped); can be edited when Windows is not running (during a Linux boot off a live CD, for example);

Windows Internal Architecture



Windows Boot Procedure

- Power On
- UEFI program executed
- The Windows bootloader Winload.exe loads basic drivers required to read data from disk
- Kernel is initialized
- Registry and non-boot drivers are loaded and started
- Winlogon.exe starts, requiring the user to login
- Upon successful login, explorer.exe is started
- Desktop window manager (DWM) is started

macOS Internal Architecture - Kernel Dev Origins

- The UNIX concept and implementation of **pipes**, which allows data to be moved between many small interacting programs, also causes/implements **blocking system calls**, which causes data to be moved in staggered pieces around the system
- Thus, the implementation of pipes as memory buffers, which copy so much data around, does not scale well when fast speeds and low latency are desired
- So the natural response in the 1980s was to write the **kernel** and core functionality as a single large, unwieldly program to prevent copying
- These large kernels are bug-prone; small interacting pieces are *much* easier to test and verify independently!

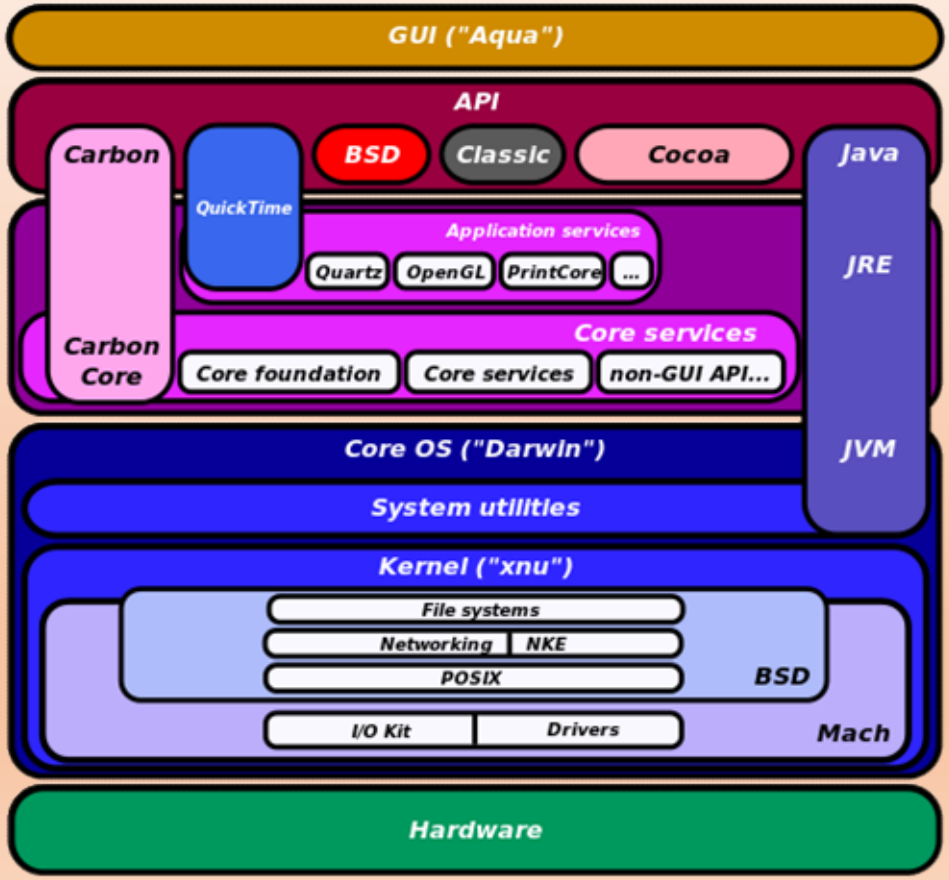
macOS Internal Architecture - Kernel Dev Outcome

- New theories of kernel design were tested out in the Aleph kernel developed at University of Rochester in 1975, where the kernel was shrunk to handle only access to hardware, including memory: thus, it uses a form of shared memory to copy information around between easily tested subsystem programs
- New CPUs developed in the early 80s offer support for a **Memory Management Unit** (MMU), which allows for virtual memory to be implemented, tracking the pages of memory in use by processes
- This allows memory that is told to be “copied” to instead be transparently *referenced virtually*, dramatically reducing the amount of data being copied by the kernel; this is called **copy-on-write** *(i.e., copy only if you’re going to make a modification)*
- These new, smaller kernels are called **microkernels**

macOS Internal Architecture

- macOS uses the Mach microkernel paired with BSD programs to provide system call access
- User programs can access macOS by using the standard, tried-and-true UNIX API system calls we've been studying, in addition to other API libraries that access additional functionality of the OS
- Drivers are abstracted as one form of "kernel extensions", which are all loaded when the OS boots - thus trusted, low-level code can be added to the kernel by third-parties

macOS Internal Architecture



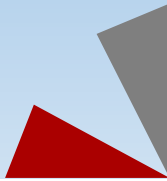
macOS Boot Process

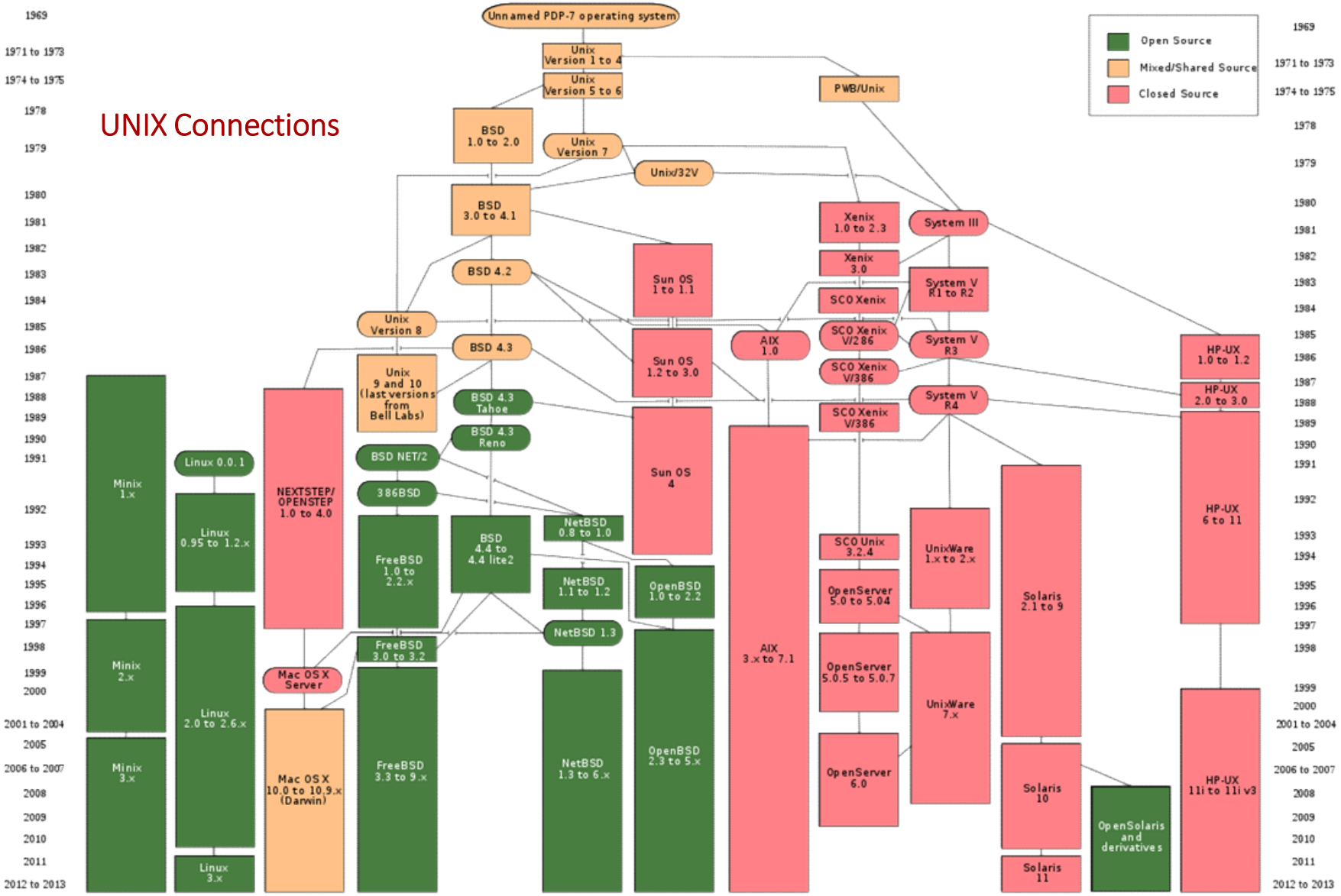
- Power On
- UEFI program executed
- Control passed to the BootX bootloader
- BootX loads previously cached list of kernel extensions, including drivers
- init process in Mach microkernel is started; control has left the firmware
- Mach and BSD data structures are initialized
- I/O starts up
- Kernel starts virtual memory management tracking routines
- Kernel starts GUI and other daemons/services

The Importance Of UNIX

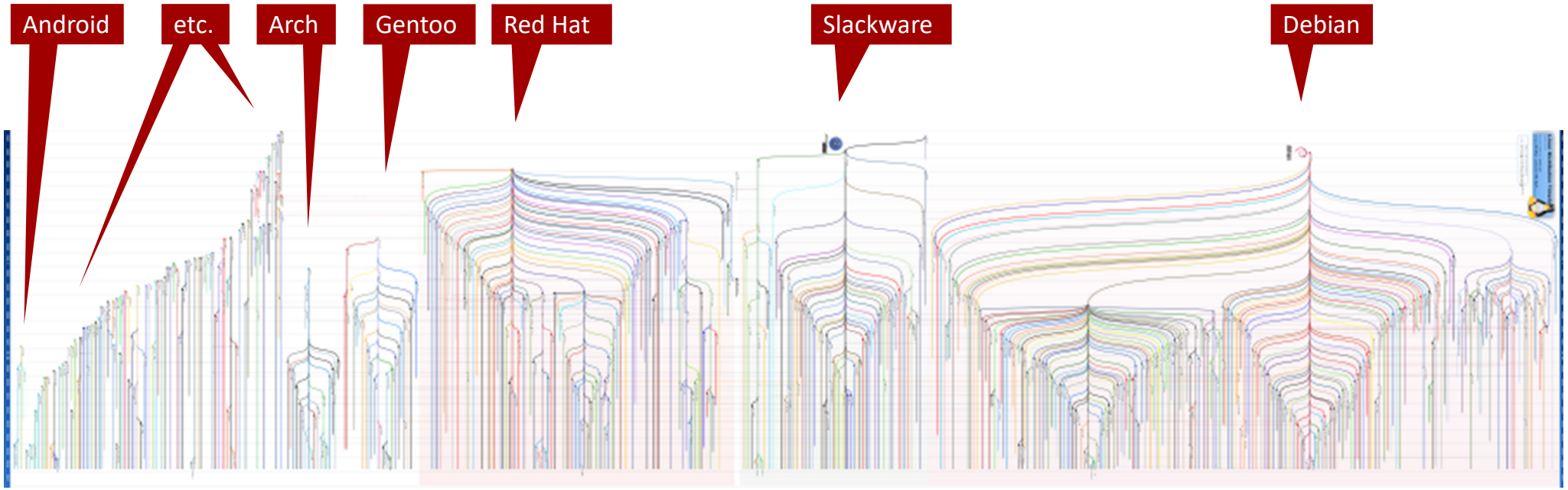
- UNIX is run on a vast majority of devices around the world
- It has generated fortunes, saved lives, and been an agent for positive change for millions and millions of people
- Today, you can even run a bash shell in Windows 10

- Let’s take a look back at the historical connections between UNIX, Linux, and all the variants





Linux Connections - Go Check This Out



A Few Early Pioneers

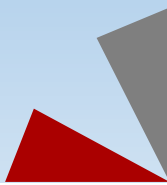
- Tommy Flowers designed Colossus (1944), the world's first programmable electronic computer to help decrypt German wartime messages
- J. Presper Eckert & John Mauchly designed and built the ENIAC (1946), the first all electronic, Turing-complete computer, and the UNIVAC I (1951), the first commercially available computer
- Margaret Hamilton coined the phrase "software engineering", instrumental in testing and timing-critical human interaction with computers; led dev of on-board software used in Apollo missions
- One of the first OSs with a software-based paged virtual memory was THE, designed by a team led by Edsger Dijkstra way back in 1962

Edsger Dijkstra (1930 – 2002) Contributions

- “Dijkstra’s Algorithm”
- Semaphores, including the Dining Philosopher Problem, deadlock, and other synchronization issues
- Operating system design, including abstraction layers
- Compiler design (he wouldn’t shave his beard until he had created the first ALGOL 60 compiler, then decided to keep it)
- Software engineering, including the paper, “A Case against the GO TO Statement”, helping shape programming as a discipline, not an ad hoc craft
- Distributed computing
- Formal specs and verification, and how to simplify them; CS cast as math

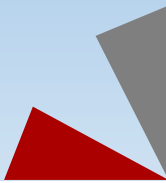
THE's layers

- Dijkstra's THE system had these layers all the way back in 1968:
  - Layer 0: Scheduler
  - Layer 1: Memory pager
  - Layer 2: Communication between OS and terminal
  - Layer 3: Managed IO between attached devices
  - Layer 4: User Programs
  - Layer 5: The User (Dijkstra says, "not implemented by us")
- What will you, dear student, do with the implementation of yourself?



Edsger Dijkstra Quotes

- "Brainpower is by far our scarcest resource.“
- "The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague.“
- "Simplicity is prerequisite for reliability.“
- "Programming is one of the most difficult branches of applied mathematics; the poorer mathematicians had better remain pure mathematicians."



Edsger Dijkstra Quotes

There may also be political impediments [to becoming an exceptional programmer]. Even if we know how to educate tomorrow's professional programmer, it is not certain that the society we are living in will allow us to do so. The first effect of teaching a methodology — rather than disseminating knowledge — is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence. In a society in which the educational system is used as an instrument for the establishment of a homogenized culture, in which the cream is prevented from rising to the top, the education of competent programmers could be politically impalatable.

Let me conclude. Automatic computers have now been with us for a quarter of a century. They have had a great impact on our society in their capacity of tools, but in that capacity their influence will be but a ripple on the surface of our culture, compared with the much more profound influence they will have in their capacity of intellectual challenge without precedent in the cultural history of mankind.

ACM Turing Lecture 1972

Edsger Dijkstra Quotes

There may also be political impediments [to becoming an exceptional programmer]. Even if we know how to educate tomorrow's professional programmer, it is not certain that the society we are living in will allow us to do so. **The first effect of teaching a methodology — rather than disseminating knowledge** — is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence. In a society in which the educational system is used as an instrument for the establishment of a homogenized culture, in which the cream is prevented from rising to the top, the education of competent programmers could be politically impalatable.

Let me conclude. Automatic computers have now been with us for a quarter of a century. They have had a great impact on our society in their capacity of tools, but in that capacity their influence will be but a ripple on the surface of our culture, compared with the much more profound influence they will have in their capacity of intellectual challenge without precedent in the cultural history of mankind.

ACM Turing Lecture 1972

We are teaching you the theories so that you can acquire any skill

Edsger Dijkstra Quotes

There may also be political impediments [to becoming an exceptional programmer]. Even if we know how to educate tomorrow's professional programmer, it is not certain that the society we are living in will allow us to do so. The first effect of teaching a methodology — rather than disseminating knowledge — is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence. **In a society in which the educational system is used as an instrument for the establishment of a homogenized culture, in which the cream is prevented from rising to the top, the education of competent programmers could be politically impalatable.**

Let me conclude. Automatic computers have now been with us for a quarter of a century. They have had a great impact on our society in their capacity of tools, but in that capacity their influence will be but a ripple on the surface of our culture, compared with the much more profound influence they will have in their capacity of intellectual challenge without precedent in the cultural history of mankind.

ACM Turing Lecture 1972

We will reward winners and victors; life doesn't hand out participation trophies

Edsger Dijkstra Quotes

There may also be political impediments [to becoming an exceptional programmer]. Even if we know how to educate tomorrow's professional programmer, it is not certain that the society we are living in will allow us to do so. The first effect of teaching a methodology — rather than disseminating knowledge — is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence. In a society in which the educational system is used as an instrument for the establishment of a homogenized culture, in which the cream is prevented from rising to the top, the education of competent programmers could be politically impalatable.

Let me conclude. Automatic computers have now been with us for a quarter of a century. **They have had a great impact on our society in their capacity of tools, but** in that capacity their influence will be but a ripple on the surface of our culture, compared with the **much more profound influence they will have in their capacity of intellectual challenge** without precedent in the cultural history of mankind.

ACM Turing Lecture 1972

Go do something useful!

Improve life, don't just live it!