Operating Systems

A brief look at the evolution of computer systems

Paul Krzyzanowski
January 27, 2015

Disclaimer

The goal of this document is to provide a high-level historical perspective on the evolution of computer systems and, with them, some key milestones in the development of operating systems. This document is *not* a complete history of computer systems or operating systems. It's not even remotely thorough. If it was, it would be far longer. While I try to be convey accurate information, it is very likely that I missed more than a few things. Treat this as a presentation of a historical perspective rather than a treatise. Any corrections and enhancements are most welcome.

In the beginning…

There were no operating systems, no programming languages or compilers, no mice and no windows, and the earth was without form, and void; and darkness was upon the face of the deep.

Suppose you built a computer. How would you get it to read and run the programs that you will write? How would it access devices: read input and write output data? A program runs from memory, but one must get the program into memory somehow. How would you then run a different program? Would you have to restart the computer? How much does the program you write have know about your system's hardware: the details of accessing a disk or scanning a keyboard? Can this information be hidden from application programmers?

Questions such as this led to the development of operating systems. The development of operating systems is closely tied to the development of computer systems, how users use them, and what they expect of them. What follows is a quick tour of computing systems through the past seventy or so years (again, much has been omitted; the goal is to give an idea of evolution rather than a thorough historical treatment).

1945: ENIAC, Moore School of Engineering, University of Pennsylvania.

The ENIAC (Electronic Numerical Integrator and Computer) is arguably the first general-purpose electronic computer (the first was really the Colossus, but its very existence was shrouded in secrecy for decades). It was created for a very specific purpose: to compute firing tables for the military. These firing tables were used in combat to find the proper angle of elevation for a gun, taking into account the temperature, wind direction, and type of ammunition. As gun production rolled into high gear during World War II and an ever larger array of guns were manufactured, the army was facing a tremendous backlog of calculations. This led to funding the ENIAC project as an attempt to solve these problems.


Eniac computer

The machine contained:

- an internal memory of 200 decimal digits
- I/O speed of 8,000 decimal digits per minute
- computation speed of 5,000 operations per second
- built with 18,000 vacuum tubes and consumed 100,000 watts (40,000 watts went to the tube filaments and were converted to heat)
- programmed through patch panels and switches
- data read from punched cards

> "Preparing ENIAC for a series of runs was an incredibly involved process. First, detailed instructions had to be written defining the problem and a procedure for solving it. These instructions were programmed by adjusting switches manually and inserting thousands of cables into as many as forty large plug boards. A team of five operators might work several days on

> the external wiring and many more days searching for errors and correcting them."
> — Breakthrough to the Computer Age, Harry Wulforst, Charles Scribner's & Sons Pub., 1982

Can we do better than this?

The experience with the ENIAC led to the realization that there can be an easier way of entering a program into a computer; one that is less labor-intensive than spending several days adjusting switches, plugging in cables, and checking for mistakes.

> "The switches and controls of ENIAC [which are] now arranged to be operated manually, can be positioned by mechanical relays and electromagnetic telephone switches which are instructed by a Teletype tape. [With this arrangement] tapes could be cut for many given problems and reused when needed"
> — Lieut. H. Goldstine

1949: EDSAC and EDVAC

Computers get memory

The EDVAC (Electronic Discrete Variable Automatic Computer) and EDSAC (Electronic Delay Storage Automatic Calculator) were both completed in 1949. The EDVAC was the successor to the ENIAC. The EDSAC was a project headed by Maurice Wilkes at the computing laboratory at Cambridge University and built based on the EDVAC proposal. Both were stored program computers: instructions could be run from the computer's memory rather than be hard-wired. They used John von Neumann's architecture where the computer's program code resides in the same storage as the data.

Mercury delay lines were used for the system's memory. These were long sealed tubes containing mercury. Data was represented as waves in the fluid. It entered via a transducer, propagated through the fluid, and was extracted via a transducer at the far end. At that point the computer could read the data (into vacuum tubes) or cycle it back for recirculation through the mercury tube.

The EDSAC had a whopping 512 35-bit words of memory.

1949 BINAC – a successor to the ENIAC

Programming languages

The first precursor to a programming language appeared: John Mauchly's *Short Order Code*. It featured a simple algebraic interpreter. A programmer could write in pseudocode rather than in the code of the actual machine. This pseudocode could include a set of "subroutines". These weren't subroutines in the sense that we know them. The program looked at each statement, jumped to the required subroutine, executed it, and looped back. The concept of a stack and the return instruction came later.

1951: UNIVAC by Remington

Reusable code

The concepts of code sharing emerge. A standard set of mathematical subroutines was created. These were written on paper and copied (onto the card punch) as needed. Unfortunately, programmers don't always copy very accurately, so debugging was often needed.

John Mauchly estimated that there were perhaps twelve organizations in the world that needed and could afford a UNIVAC. Over 45 were sold.

1952: IBM 701

True reusable code and a real assembler

The 701 was IBM's first commercial scientific computer. It contained 1,024 words of random access memory and used plastic magnetic tape as a storage medium. The computer featured a modular construction; it was built in several units that would be shipped to the customer where it would undergo final connections. This contrasts with earlier machines like the UNIVAC that had to be fully assembled at the customer site.

The 701 brought about:

- The emergence of an assembler-style language: IBM Speed Coding
- SHARE (Society to Help Alleviate Redundant Effort), an IBM user organization, was created. It maintained a repository of common routines. For example, if you needed code for advancing the tape to a certain position and writing a block of data, you could get it from SHARE. These shared I/O routines were the precursor to **device drivers** found in today's operating systems.

1956: The interrupt

The UNIVAC 1103A introduced the **interrupt**. Upon receiving an interrupt, the processor would store the value of its program counter in a fixed memory location and the program counter would be replaced with a preset address, forcing execution to switch to an **interrupt service routine**. To return, the interrupt service routine would branch to the value that was stored in the special memory location. One way in which the interrupt was used was to alllow lower-priority batch processes to run in between war game simulation programs at the Army's Operations Research Office. This was the precursor to time-sharing, which would dominate operating system design for decades to come.

The Operating System emerges

A big time waster in early computing was the time that it took to set up a computer to run a job and then reset it and load the next job when the first program completed. To run a program, a deck of punched cards containing the program was loaded into a machine's memory. This program was the *only thing* that ran on that computer until the program terminated. When the job completed, an operator would load a program to dump memory and would remove the tape, cards, and any printout. After that, the next job would be loaded on the computer.

Batch Processing:

Early operating systems improved system throughput by processing jobs in batches rather than individually. When a job finished, it would branch to a location in the "operating system" that would contain software that would dump the state of the current job, and load the next job. This greatly reduced the need for operator intervention. In the earliest systems, this job transition software was encapsulated in a set of punched cards that were added to the deck of cards that contained the user's program.

Job control languages provided a way for programs to define their resource needs, such as what tapes or card readers they need to use and what their anticipated run time is.

1954–1957: FORTRAN was developed

A high-level language

With high-level languages, programmers no longer had to to understand the architecture of a computer.

Early compilers, including early FORTRAN (FORmula TRANslator) had to fit in 1,024 12-character words and still leave enough room for data. It often took two hours to compile a program because the tape units were kept busy refetching data.

Operating Systems by the late 1950s

By the late 1950s, Operating systems supported:

- Single stream batch processing
- Common, standardized, input/output routines for device access
- Program transition capabilities to reduce the overhead of starting a new job
- Error recovery to clean up after a job terminated abnormally
- Job control languages that allowed users to specify the job definition and resource requirements

Into the 1960s

The goal of operating systems in the early 1960s was to improve throughput. Computers were incredibly expensive and it was important to find ways of utilizing every possible second of CPU time.

Multiprogramming: several programs were kept in primary storage at once and the processor was switched between programs. Multiprogramming works because of the mismatch between I/O (input/output) device speeds and processor speeds. Tens of thousands of cycles could be wasted while a program was waiting for an I/O operation to complete (tens of millions to billions on today's processors). Instead of having a processor sit there and do nothing while an I/O operation was going on, it could switch to running a different program.

Multiprocessing systems: several processors cooperate with one another. Some of these began to emerge in the 1960's.

Transaction processing systems: IBM developed the SABRE airline reservation system for American Airlines in 1960. It consisted of two connected IBM 7090 computers (built with discrete transistors) Travel agents communicated with the central computer via typewriter terminals. Although SABRE was a dedicated system and not a general purpose operating environment, it ushered in new requirements for computing.

With systems such as SABRE:

- User accounts and concurrent access required the creation of protection mechanisms and password storage.
- Computers now had to operate interactively (conversationally). Because of this, human factors became an issue: response time and ease of use.
- With the widespread deployment of disks, access storage organization techniques were developed followed by file systems.

- Data communication links were used for remote access. For the most part, these were telephone lines.

1961: The dawn of minicomputers

Computing for the masses!

Digital Equipment Corporation (DEC) introduces the Programmed Data Processor 1 (PDP–1). It's a veritable bargain at $125,000-$250,000 as any competitive systems at that time began at $1 million. Moreover, it needed no staff of attendants and operators. A lot more companies could now afford computers. In fact, individual departmeents could have their own computer.

1962 Compatible Time-Sharing System (CTSS) from MIT

Time sharing

CTSS ran on the IBM 7094 with special hardware. It proved the value of interactive time sharing systems and the value of sharing data and programs on-line. This system was a precursor to the Multics operating system.

1963 Burroughs Master Control Program (MCP) for the B5000 system

Virtual memory and multiprocessing

MCP was the first operating system written in a high-level language (note that this was *not* a portable operating system; it only ran on the B5000 computer). The B5000 system running MCP gave the user:

- Multiprogramming
- Multiprocessing
- Virtual storage (provides the illusion of a system having more physical memory than it really does)
- Source language debugging capability

1964: IBM System/360

The "system"

IBM bet the company on this one. It created a line of five computers, all compatible, that gave users a migration path as their computing needs grew. It included the largest assortment of simulators and emulators ever assembled for backward compatibility. Card readers, tape drives, printers, and memory devices were redesigned to make them faster (for instance, the System/360 Model 1403 Chain Printer printed 1,100 lines of text per minute).

R&D for the project cost over $1 billion dollars. Within six months after its announcement, IBM booked orders for System/360s that totaled more than three times IBM's entire annual income. The bet paid off big time.

System/360 operating system goal: deliver one operating system for the entire line of computers.

The reality: It proved difficult to create a system that would satisfy users with varying degrees of sophistication. It also didn't seem fair to burden smaller machines with heavy-duty operating systems. A family of operating systems emerged:

Traditionally, a simulator interprets all ope in software while an emulator has hardwa support the execution of instructions. Thes it is common to use both terms to refer to software. Emulation now usually refers to software or hardware that models the inter and behavior of a device but not necessari internal state. The emulated system may b differently internally but provides the sam external interface. API hooks to implemen given hardware function of some device a example. Simulation refers to modeling th internal components of a system.

- PCP/360 — sequential job system
- MFT: multiple job system, fixed number of tasks. The system memory is divided into a fixed number of partitions. Programs are loaded into these partitions and the processor spends some time on each partition
- MVT: multiprogramming with a variable number of tasks. A new job asks for a partition of a certain size, gets it, runs, and then the partition is free again. This later became IBM's MVS system. All memory references are direct memory references at this time (the memory address you ask for is the one you get).

1960s: Disks become mainstream

Random access high-capacity storage

The first disk drive, the IBM 350 disk storage unit, was introduced in 1956. It had a capacity of 3.75 megabytes stored as five million six-bit characters on fifty (!) 24-inch diameter disks. IBM leased it for $3,200 per month.

By the 1960's disk storage finally comes into widespread use. It offers high capacity, almost-random-access-storage. IBM created a family of Disk Operating Systems (DOS, DOS–2314, DOS MP, DOS VS, DOS/VSE) to take advantage of this technology.

1966: Minicomputers get cheaper, more powerful, and really useful

Widespread computing

DEC introduces the PDP–8. It costs a mere $20,000 and becomes used by small businesses, universities, and high schools. Five operating systems are available. Here's a list, paraphrased, from*Introduction to Programming PDP–8 Family Computers, Digital Equipment Corporation, 1969*:

- Paper Tape Programming System — I/O of programs and data is performed manually via a paper tape reader and punch All programming systems begin by loading certain system and utility programs into core via a paper tape.
- DECtape Programming System — I/O of programs and data is performed via a DECtape unit.
- 4K Disk Monitor System — 4K system programs are run under control of the Disk Monitor with disk input and output. You'd use this with a 4K memory system. A DECtape contains over half a million octal words of storage.
- PS/8 Programming System — 8K system programs are run under control of the 8K executive program using either DECtape or disk input/output. System and utility programs include languages (FOCAL, BASIC, FORTRAN-D, assembler), and utilities (PIP for peripheral device control, DDT for debugging, and an editor).
- TSS/8 Time-Sharing System — More than one user is sharing the computer under control of he TSS/8 Monitor, using 4K system programs in a disk environment. Each user has a separate Teletype terminal. It allows device-independent access to as many as 15 I/O devices. The user program may call upon the executive routines for several services, including loading device handlers, assigning names to devices, creating files, and performing line input/output to the Teletype.

Operating Systems for the 1970s

The 1970s were characterized by the following traits:

- Multi-user, multi-tasking reigns supreme.
- Dynamic address translation hardware creates virtual memory that operating systems must handle. Virtual machines can now exist.
- Modular architectures
- Emergence of portable design
- Personal, interactive systems

We also see two developments that will transform computing:

- Data networking. Computers begin to get connected over data communication links.
- Microprocessors. These will make computers small, cheap, and personal. For about 20 years, these will be technological laggards, rediscovering what *real* computers already had many years earlier. By the 1990s, they transform into technology leaders.

1967–1968: The mouse

Introduced a new form of user interaction … dominant until the iPhone & iPad

The mouse was created by Douglas Engelbart at the Stanford Research Institute's Augmentation Research Center (ARC). He and his team at the ARC invented the mouse, the bit-mapped display, graphical user interface concepts, teleconferencing, and hypertext. This was demonstrated in 1968 at the Fall Joint Computer Conference and is known as The Mother of All Demos. This was back in 1968. We have not seen substantial changes to this for 47 years!

1964 and onward: Multics

Multics was an ambitious operating system developed by MIT, General Electric, and Bell Labs. It was designed to be the operating system for GE's GE–645 mainframe but was later targeted to Honeywell machines after Honeywell acquired GE's computer business. It introduced some unique ideas:

- All system memory was mirrored onto the disk and available via the file system
- Dynamic linking: load and add code and data segments to a running process
- Interprocess communication via shared segments
- Multiprocessor support
- On-line reconfiguration of system hardware without downtime

- Hierarchical security model using protection rings
- Hierarchical file system with arbitrary file names
- Symbolic links
- Command processor not part of the operating system
- Written in a high-level language, EPL, a subset of PL/1 with extensions
- I/O redirection to files and programs (“pipes”)

With all its features, it was big and bloated. Performance wsa poor and the compiler was abysmally slow. The frustrations of Multics led the Bell Labs part of the Multics team to create UNIX.

1969: The UNIX Time-Sharing System from Bell Telephone Laboratories

Small OS, toolkit approach

The first edition of UNIX ran on a PDP–7 and was written in assembly language. It emphasized a small and simple kernel and a rich set of utilities (that mostly handled line-oriented text-based data) that could be connected together to perform a variety of tasks. It incorporated a number of ideas from Multics, such as a hierarchical file system, multiple levels of file access permission, input/output redirection, and pipes.

1971: Intel announces the microprocessor

Not a big deal yet, but just wait…

The Intel 4004 contained 2,000 transistors and performed 60,000 operations per second. It was a four bit machine with effectively as much power as the ENIAC. It found a home in some desktop calculators.

c. 1970: DEMOS, Los Alamos

Message passing

This system ran on the CRAY–1 supercomputer and was the first operating system to be based on message-passing.

1972: IBM comes out with VM: the Virtual Machine Operating System

Virtual memory and virtual machines

Goal: Can we build a machine on which we could run different operating systems concurrently to develop them, debug them, and measure their performance?

Developing an operating system is tricky: you need a dedicated computer on which to run the operating systems. With mainframe computers being incredibly expensive (millions of dollars) and taking up a lot of space, even IBM programmers couldn’t get one apiece.

IBM built a modified System/360 (the model 67) that supported address translation. Address translation means that the processor thinks it’s referencing one memory location but it really gets translated into another. Every process can now get its own address space and feel like it owns the machine.

VM was built with a modular approach: resource management and user support were split into separate components. The Control Program (CP) is the part of VM that creates virtual machines on which various operating systems run. A virtual machine contains not only a virtual address space, but also virtual readers, disks, punches, printers, and tape drives. Accessing devices or executing privileged instructions, both of which operating systems do, causes an interrupt to the CP which maps the requested action into a physical action. With VM, one can:

- Test out a new operating system while still doing production work on the old one
- Run multiple copies of the same operating system (VM/CMS runs a copy of CMS, a single-user system, for each user).
- Run special-purpose operating systems for certain tasks

VM started as an internal IBM project to give its engineers the ability to develop and test operating systems but turned out to be useful for IBM customer’s as well. Virtual machines pretty much died off after this until the 1990s but are enjoying a renaissance in the 2000s.

1973: UNIX 4th Edition is published

Portable operating system

This is the first version of UNIX that was mostly written in the C programming language. The system runs on a PDP–11 and comes with an editor, assembler, calculator, electronic mail, compiler, and a bunch of utilities. Since the Bell System was a regulated monopoly and could not be in the computer business, UNIX was practically free to universities (they had to pay for the

> The number of UNIX installations is now 20, and many more are expected. None of has exactly the same complement of hardv software
> — *Preface to the fourth edition.*

documentation and tape). You didn't get support, but you got source. UNIX was an easy system to understand and modify. It was soon ported to different machines.

Three things about UNIX were crucially important:

1. UNIX was mostly written in C. This made it easier to write, debug, enhance, and maintain the software. The approach of using a high-level language for writing an operating system was a relatively novel one. Operating systems were written for specific machines, had to be efficient, and had to access various low-level aspects of the machine. The main compiled high-level languages of the 1950s and 1960s were FORTRAN and COBOL (LISP was also out there but it was interpreted). Both were unsuitable for the task. Previously, Burroughs wrote their operating system for the B5000 in a version of Algol. MULTICS, a large OS project that preceded UNIX, was written in EPL, an early version of PL/I. The language was largely unsuited to operating systems and the compiler was so unoptimized that " a simple PL/I statement might translate into a whole page of code" [ref]. C was a simplified high-level language that was ideal for systems programing. While it incurred some inefficiencies, they were not unreasonable ones.

1. The use of a high level language made it easier to recompile the operating system for a different computer architecture. Traditionally, operating systems were targeted for a specific machine architecture. For example, MULTICS, the predecessor to UNIX, was designed specifically for a GE–645 computer. The architecture and the OS went hand-in-hand. While UNIX was initially written for a PDP–8 and then a PDP–11, the fact that large sections of the operating system were architecture independent and written in a high-level language meant that it was not a daunting task to port the code onto different computer architectures.
2. By 1975 (sixth edition), UNIX was distributed to universities. This exposed a large group of computer science students to the operating system at the source code level. Here was a system they could use, modify, and fully learn. This led to numerous enhancements. Most came from the University of California at Berkeley in the form of the Berkeley Software Distribution (BSD)

1973: Ethernet

Robert Metcalfe invents Ethernet while at Xerox PARC (Palo Alto Research Center). It's not the first data network but it was quickly promoted as a standard and its packet switching approach made it inexpensive and highly scalable. It quickly becomes the dominant networking technology for local area networks.

1973: Xerox Alto

Video display, the desktop, the mouse, ethernet networking, WYSIWYG editing

The Alto was a project developed at Xerox PARC. Although it was not a commercial success and Xerox abandoned the project, it was pivotal in the history of computer systems since it integrated the mouse, ethernet networking, a bitmapped video display, and a detachable key. It also introduced the desktop interaction metaphor on a graphical display.

1974 The Personal Computer Age begins

The first personal computer

Popular Electronics featured the MITS Altair computer on its cover. A kit for this 8080-based machine cost $397. It had no keyboard, video display, disk drive, tape storage, or software. Most systems had 256 bytes of memory (that's *bytes*, not kilo-, or megabytes). Instructions were entered through switches on the front panel.[1]

Intel announces the 8080 microprocessor. larger instruction set than its predecessor, 8008. Better yet, it requires only six additi chips to produce a working computer. It e one million instructions per second and in costs $360. Prices fall to under $100 by la thanks to competition.

A later model, the Altair 8800b, had a serial line to support a video terminal and printer. At the time, a Lear-Siegler ADM–3A, a dumb video terminal with an 80x24 character display, cost $995/kit, $1195/fully assembled). If you couldn't afford that, you might get a used teletype. Thousands upon thousands of people bought these computers. They couldn't do all that much, but it was pretty cool for many to own your very own computer. This led to the Homebrew Computer Club, which led to the creation of more PC and peripheral manufacturers as well as to the creation of software.

Now that there was an obvious market for this, Intel was under attack. By 1975, the Intel 8080 processor was being sold for $179. In 1975, the 6502 processor was introduced for a mere $25. With this low price, it became the processor of choice for a number of personal computers, including the Commodore PET, The Apple II, Atari, and others. Also, a better and cheaper processor that was backward compatible with the Intel 8080 instruction set was introduced by Zilog in 1976 — the Z80. Mainstream personal computers would not see an Intel processor in them until the IBM PC.

1974: Gates and Allen write BASIC for the Altair

PC programming made simple

Now lots of people could write programs on their PC without having to use assembly language. Most people stole this software, which led to Bill Gates' Open Letter to Hobbyists.

c. 1975 CP/M: An operating system for 8-bit Intel-based PCs

The introduction of the BIOS (sort of) into Intel PC architectures

Gary Kildall of Digital Research gives the world CP/M (Control Program for Microcomputers). It runs on Intel 8080 (and on the later 8085 and Zilog Z80) machines. It was not targeted to any specific machine and was designed to be adaptable to any machine with an 8080-family processor and a floppy disk drive. It was a rudimentary operating system: a simple command interpreter, program loader, and file system manager. Only one program ran at a time. When it was done, command transferred to the console command processor code, which prompted the user for the next command. This operating system consists of:

CCP

The Console Command Processor (command interpreter)

BDOS

The "Basic Disk Operating System". This was the program loader and the software that figured out how to manage and interpret the file system on a floppy-disk.

BIOS

This term should sound familiar. The Basic I/O System (BIOS). Since each machine was different, you had to write the low-level functions yourself (get a character from an input device, read a sector from the floppy disk, write a sector, write a character to an output device, …).

The system also came with a few programs, such as an assembler, a line editor, and a program to copy files. This is the sort of software people expected to come with their computers at that time. CP/M was a direct precursor of MS-DOS. MS-DOS was derived from QDOS, the *Quick and Dirty Operating System*, which was essentially a reverse-engineered version of CP/M for the 8086 processor, Intel's first 16-bit CPU. Microsoft bought QDOS from Seattle Computer Products for $50,000. Gary Kildall could have had the IBM contract for the IBM PC operating system, but he botched it. The first IBM PC came with a BASIC interpreter in ROM and an assembler, a line editor, and a program to copy files.

1976: Apple II

A ready-to-use computer

BASIC is built-in. It allows the display of text and graphics in color. This was a "plug & play" machine that remained in production for fifteen years. This machine made personal computers accessible to non-geeks. That was a good chunk of the population.

By 1977, a couple other ready-to-use PCs were sold: Radio Shack's TRS–80 and the Commodore Pet.

1977 DEC introduces the first VAX computer running VMS (the VAX 11/780)

A 32-bit family of minicomputers sharing the same instruction set and virtual memory

The VAX 11/780 was a hugely popular larger minicomputer and one of the earliest computers to use **virtual memory** to manage the [at the time] huge 32-bit address space of the computer (the smaller PDP–11 series was a 16-bit machine).

Like the IBM System/360 (but smaller and cheaper and from a different company), this was to become a series of computers. Despite the range of size and power of different models, they all have the same architecture. VAXes (also called Vaxen) could be networked together and operate in a peer-to-peer relationship (any machine can be a client or a server).

VMS (Virtual Memory System) was the DEC-issued operating system for the VAX, although the VAX soon became a popular platform for UNIX.

VMS was designed to take advantage of the VAX's architecture, incorporating **demand paging**, which enabled the computer to allocate and load a page of memory when the process needed it and map it onto the correct memory location for the process. This avoided having to pre-load an entire program into memory. Securiy was a core facet of the design. It supported privilege checks and account lock0outs. VMS supported 32 priority levels for process scheduling and had support for real-time processes.

1979: Visicalc and WordStar come out

Killer apps

These are the first of the *killer apps* for the personal computer: programs that would justify your purchase of the computer.

Operating Systems for the 1980s

Personal computers are a strong force, but the operating systems are primitive: not much more than a command interpreter, program loader, and device drivers. Even with that, programmers often break convention and access devices directly. Networking, particularly among

workstations, becomes widespread and fast. It's feasible to access files across a network as one would locally. Networked operating systems become interesting.

Microkernels with a message passing structure become hot topics, but never really become mainstream operating systems. Mach, designed as a microkernel, becomes a basis for both Windows NT (and derivatives) as well as OS X but is really too big to be considered a microkernel architecture.

A key mechanism that entered operating systems was *multithreading*, the ability for one process to have multiple concurrent threads of execution. Initially, this was primarily useful for network servers since one process may now service many requests concurrently.

User interfaces start becoming important. Windowing systems get refined.

August 12, 1981: IBM introduces the IBM PC

IBM legitimizes the personal computer

The IBM PC was an open machine based on the Intel 8088 processor (IBM will give you the schematics and parts list). With the most respected brand in computers selling a personal computer, it was no longer a hobbyist's toy but something that could be a serious personal business machine. A lot of people bought these. You couldn't go wrong buying IBM and IBM set a standard that is still followed. Intel and Microsoft (not IBM) were the big winners here because lots of other companies started building clones of the IBM PC but those clones still needed the processor and needed the operating system.

The first IBM PC featured:

- 16K bytes memory (expandable to 256K)
- 2 floppies, each holding 160K bytes
- 2 displays available: a color and a monochrome
- PC-DOS from Microsoft. This was essentially a CP/M clone. Bill Gates agreed to provide an OS to IBM even though he had nothing to give at the time. He cleverly negotiated the rights to sell the operating system (probably anticipating the PC clones that would come out in the near future).
- A price of $2495 with 16K bytes RAM and one floppy. Add $1000 to get 256K bytes RAM

1983 Microsoft begins work on MS-Windows

It's slow, buggy, and generally an embarrassment. People stick with MS-DOS.

1984 Apple Macintosh comes out

Personal computers for everyone

The Mac introduces the mass market to the mouse and windows. The command line is now dead as far as the average consumer is concerned. Popular for its user-friendly interface. The point-and-click mouse-based interface is derived from the Xerox Star system created at Xerox PARC (Palo Alto Research Center).

c. 1985: Networked file systems on workstations

Remote resources that look like local ones

SUN's NFS allows users to *mount* file systems from other machines onto their own machines. Apollo's Domain system incorporates the entire network of machines into its file name space. Now you can access data that lives on other machines in just the same way as you access data on your machine. A bunch of companies start selling networked personal workstations for the enterprise. Sun wins.

1986: Mach

Microkernels

Mach is a microkernel system that was designed to allow the emulation of a variety of other operating systems over it (such as various variants of UNIX). It allows transparent access to network resources, can exploit parallelism, and can support a large address space. The principles of Mach become a basis for Microsoft's Windows NT and for Apple's OS X.

1980's Amoeba

A microkernel with autonomous components

The system is a distributed collection of process servers (processor pools), file servers, compute servers, and others.

Amoeba is built with a microkernel architecture. The Amoeba microkernel runs on all machines in the system The fundamental concept under Amoeba is the object (a collection of data upon which certain well-defined operations may be performed). These objects are named and created by capabilities. Amoeba demonstrated a number of interesting concepts but was never adopted commercially.

Late 1980's: Plan 9 from Bell Labs

UNIX is getting old; let's take the good ideas, dump the bad, and start over

Built with the realization that previous efforts of building distributed networks of systems and getting them appear as one uniform entity weren't entirely successful.

Goal: Build a distributed, scalable system that appears as one time-sharing system that can support thousands of users, terabytes of files, and gigabytes of memory. Plan 9 is composed of a number of separate components: CPU servers, terminals (with processing), file servers, and networks.

Plan 9 never gained commercial traction or even much adoption within Bell Labs. Although it had great ideas, the user community didn't care a whole lot about new operating systems; they were no longer satisfied when presented with an operating system and a few programs. They want to run their favorite applications. Backward compatibility was important.

1990 Microsoft Windows 3.0 comes out

Microsoft Windows 1.0 first came out in 1985, followed by Windows 2.0 in 1987. These were essentially just graphical shells over MS-DOS, which is a single user, single-tasking system. However, Windows provided users with drop-down menus, scroll bars on on-screen windows, and dialog boxes. Windows 1.0 required a PC with a minimum of 256 KB of memory. Neither of these releases were hugely popular as most applications still ran under MS-DOS.

In 1990, Microsoft introduced Windows 3.0. This was the first hugely popular version of Windows, although many of the DOS underpinnings were still clearly visible. Performance improved, icons were better developed, and the system fully supported Intel's new 386 processor. The 386 processor was the first Intel processor to provide support for virtual memory, which now allowed Windows to multitask multiple instances of MS-DOS programs, giving each program its own virtual address space. Equally importantly, Microsoft introduced a Windows Software Development Kit (SDK), which provided developers with a standard API for accessing system functions, including graphics, ushering in a huge wave of application development for the platform.

1992 The first Windows virus comes out

The virus, named WinVir, was discovered. An infected program would search for other executable files and edit those files to insert copies of itself in them. Then it would remove itself from the original file to restore that file to its original condition. Viruses became an endemic problem largely because operating systems users generally ran with administrative privileges in systems such as Windows, allowing programs that they execute to have free access to all files in the system.

1991 GNU/Linux

A free Unix-like operating system becomes available

UNIX, while achieving great popularity, was only freely available to universities. The Bell System was a regulated monopoly and could not sell computer products. After the breakup of the Bell System in 1984, AT&T wanted to get into the computer business and free university licensing of the UNIX operating system was revoked. Minix was created by Andrew Tanenbaum as a small Unix-like operating system. Unfotrunately, like UNIX in the past, it was freely available only to universities and research.

Linux started as a kernel originally written by Linus Torvalds and was complemented with a wide range of GNU user-level tools (gcc, make, emacs, and much of the other stuff that one expects in a Unix distribution). It evolved over time. Although it has not achieved widespread popularity in PC desktops (where Windows and OS X dominate), it runs on many servers as well as embedded systems. In the latter category, it serves as the underpinnings for Google's Android and Chrome OS operating systems, the TiVo digital video recorder, various set-top boxes, automotive computers, and industrial controllers.

1993 Windows NT

The design of VMS was led by David Cutler. When the project was canceled in 1988, Microsoft hired Cutler and around 20 other DEC employees from the VMS team to work on a next generation operating system for Windwos that would rival UNIX. This led to the creation of Windows NT. NT is, in many ways, a successor to VMS and shares much terminology and concepts in common.

Windows NT was also inspired by Mach's microkernel architecture. NT is not a microkernel but enables OS **emulation subsystems** to run as user-level server processes. This enables it to implement backward compatibility support with other operating systems, including DOS, OS/2, POSIX (UNIX interface), and 16-bit Windows.

"NT" stood for New Technology. It was designed to be a better Windows, offering networking, per-object based processes, and file protection mechanisms. Unlike VMS, it was written almost entirely in C and designed to be portable across different processor and hardware architectures. MS-DOS and earlier editions of Microsoft Windows were written only for the Intel x86 family of processors. NT introduced a hardware abstraction layer to provide an abstract interface to the underlying hardware. The system was fully 32-bit while Windows was written with 16-bit architectures in mind. At the high levels of the operating system, API modules provided the option to support a variety of system call interfaces, although only Win32 was ultimately supported. The system could also handle a variety of installable file system modules, with NTFS being the core file system.

NT was arguably the last modern operating system that was written from scratch. Every successive version of Windows has been based on an evolution of Windows NT.

1993: Mosaic

The web browser is born

Mosaic, the Netscape Navigator (1994), and Internet Explorer (1995) usher in the web. The web browser becomes the killer app for the mid–1990s and onward. At first, the web is just an information navigation tool but it very quickly becomes an applications and service delivery platform where the web browser serves as a user interface to services that reside on remote Internet-accessible servers. It is almost inconceivable to image life without the web.

The browser, of course, is not an operating system, but it let many to reexamine the role of the operating system for consumer computing and imagine a world where all services are provided via a web browser.

The 2000s

c. 1997 and another try in 2009: The Internet appliance

If all the good stuff is on the web, how much do you really need in a PC and operating system?

The complexity and cost of managing bloated PCs (and the requisite software and disks) led to the idea of *thin clients* or the *appliance computer* (take it out of the box, plug it in, and it works) that will run a very limited set of software (such as a web browser) All the interesting stuff will be out on the network.

The first attempt to do this didn't succeed. In July 2009, Google announced Google Chrome OS. The goal of Chrome OS is to create a lightweight system with a minimal user interface running a browser. All the applications are on the web. The jury is still out on whether this will be a success.

2007: iOS

Multitouch UI

Apple's iPhone makes the multi-touch user interface mainstream. The user interaction model includes a virtual keyboard and support for tap, swipe, pinch and un-pinch gestures to zoom and unzoom to support direct, finger-based, manipulation of on-screen objects. The concept of multi-touch interfaces goes back to the 1970s, with interfaces using pinch-to-zoom gestures being created in the early 1980s. The underlying operating system, named **iOS**, is derived from OS X but with changes to support a different windowing system, restrictions on background processes to enable better application performance and improved battery life, and mobile services such as push notification and the ability to notify apps when a location changes.

2008: Android OS

A free [2] OS platform for mobile phones

Google develops a variant of Linux that is targeted to mobile phones. The Linux system and its libraries are stripped down of excess baggage, standard libraries for accessing components such as a GPS, accelerometer, telephony, and other components found on mobile phones are added, and a graphical UI framework is put in place. All applications are written in Java run on a Java-like virtual machine (Dalvik). This ensures that the applications are not dependent on any specific processor platform.

2009 The Internet of Things (IoT)

All sorts of appliances and devices talking with each other

The term, *Internet of Things* was coined by Kevin Ashton in 2009 but the ideas go much further back in time. As microcontrollers became smaller, cheaper, and more power efficient and wireless networking became ubiquitous, it became clear that all manner of objects could be controlled or could report their status. This include things such as

- light bulbs (e.g., the Philips Hue connected bulb)
- fitness devices (e.g., the Fitbit activity wristband)
- home alarms, heating, AC systems (e.g., the Nest thermostat and Nest Protect smoke detectors)
- hot water heaters
- washing machines, ovens, and other appliances
- shipping containers
- vending machines
- digital signage, from billboards to shelf tags

and many more. A key development is not just the ability to control devices from an iPhone but to enable intelligent machine-to-machine (*M2M*) communication. To enable this requires standardized APIs and network protocols. Given that many of these devices are small and simple, installing a bloated operating system such as Linux is overkill. However, security can be an important concern and needs to be incorporated at the most fundamental levels of the system while still providing convenience and expected functionality. Because these devices are not likely to get software updates and patches as often as PCs (or, in most cases, never), they are at risk of harboring known security vulnerabilities. The research and industrial community is still in the process of developing suitable operating environments and communication protocols for these systems.

References (partial)

- Digital Equipment Corporation, *Digital PDP–8 Handbook Series*. © 1970
- Harrison, William, *IBM Corporation*, private correspondence
- Ritchie, Dennis, *Bell Telephone Laboratories*, private correspondence
- Tanenbaum, Andrew S., *Modern Operating Systems*. Prentice Hall © 1992
- Thompson, K. and Ritchie, D. M., *UNIX Programmer's Manual, Fourth Edition*. Bell Telephone Laboratories, © 1972, 1973
- Wilkes, Maurice, *Memoirs of a Computer Pioneer*. MIT Press, © 1985
- Wulforst, Harry, *Breakthrough to the Computer Age*. Charles Scribner's Sons, © 1982
- The Xerox Alto Computer, Byte Magazine, September 1981, pp. 56–58
- and many others whose contents litter the recesses of my mind but whose sources I no longer remember. I will update this reference list over time.

Some web sources:

- Interrupts, Mark Smotherman, 2008
- [The UNIVAC 1102, 1103, and 1104][], The Unisys History Newsletter, George Gray.
- ENIAC
- IBM Archives on the IBM 701
- IBM Archives: System/360 Announcement
- IBM Exhibits: IBM Mainframes
- VM operating system, Wikipedia
- The Unofficial CP/M Web Site: binaries, source code, manuals, emulators.
- Intel Museum: Intel 4004
- Federeico Faggin's information on the Intel 4004 and 8080. He is the creator of the 4004 processor and founder of Zilog, which built the competing Z80 processor.
- Doug Engelbart Institute, the father of the mouse.
- The UNIX Time-Sharing System, Dennis M. Ritchie and Ken Thompson, Bell Laboratories, 1974
- The Evolution of the Unix Time-sharing System, Dennis M. Ritchie, Bell Laboratories, 1979
- IBM Archives: IBM Personal Computer
- Multics page at MIT
- Multics Simplified I/O Redirection (pipes)
- PL/I on Multics, multicians.org
- The Multics Operating System, Wikipedia article
- Windows NT and VMS: The Rest of the Story, Mark Russinovich, WindowsIT Pro, 1998.
- Plan 9 from Bell Labs
- Google Android
- Bruce Schneier, The Internet of Things Is Wildly Insecure — And Often Unpatchable . Wired, January 6, 2014

This document is updated from its original version of September 11, 2010.