Operating system

An **operating system** (**OS**) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer – from cellular phones and video game consoles to web servers and supercomputers.

The dominant general-purpose[3] desktop operating system is Microsoft Windows with a market share of around 76.45%. macOS by Apple Inc. is in second place (17.72%), and the varieties of Linux are collectively in third place (1.73%).[4] In the mobile sector (including smartphones and tablets), Android's share is up to 72% in the year 2020.[5] According to third quarter 2016 data, Android's share on smartphones is dominant with 87.5 percent with also a growth rate of 10.3 percent per year, followed by Apple's iOS with 12.1 percent with per year decrease in market share of 5.2 percent, while other operating systems amount to just 0.3 percent.[6] Linux distributions are dominant in the server and supercomputing sectors. Other specialized classes of operating systems (special-purpose operating systems)[3][7], such as embedded and real-time systems, exist for many applications. Security-focused operating systems also exist. Some operating systems have low system requirements (e.g. light-weight Linux distribution). Others may have higher system requirements.

Some operating systems require installation or may come pre-installed with purchased computers (OEM-installation), whereas others may run directly from media (i.e. live cd) or flash memory (i.e. USB stick).

Contents

Types of operating systems

Single-tasking and multi-tasking

A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by time-sharing, where the available processor time is divided between multiple processes. These processes are each interrupted repeatedly in time slices by a task-scheduling subsystem of the operating system. Multi-tasking may be characterized in preemptive and co-operative types. In preemptive multitasking, the operating system slices the CPU time and dedicates a slot to each of the programs. Unix-like operating systems, such as Solaris and Linux—as well as non-Unix-like, such as AmigaOS—support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking; 32-bit versions of both Windows NT and Win9x used preemptive multi-tasking.

Single- and multi-user

Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem.[8] A multi-user operating system extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users.

Distributed

A distributed operating system manages a group of distinct, networked computers and makes them appear to be a single computer, as all computations are distributed (divided amongst the constituent computers).[9]

Templated

In the distributed and cloud computing context of an OS, *templating* refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in virtualization and cloud computing management, and is common in large server warehouses.[10]

Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines with less autonomy (e.g. PDAs). They are very compact and extremely efficient by design, and are able to operate with a limited amount of resources. Windows CE and Minix 3 are some examples of embedded operating systems.

Real-time

A real-time operating system is an operating system that guarantees to process events or data by a specific moment in time. A real-time operating system may be single- or multi-tasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved. Such an event-driven system switches between tasks based on their priorities or external events, whereas time-sharing operating systems switch tasks based on clock interrupts.

Library

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries and composed with the application and configuration code to construct a unikernel: a specialized, single address space, machine image that can be deployed to cloud or embedded environments.

History

Early computers were built to perform a series of single tasks, like a calculator. Basic operating system features were developed in the 1950s, such as resident monitor functions that could automatically run different programs in succession to speed up processing. Operating systems did not exist in their modern and more complex forms until the early 1960s.[11] Hardware features were added, that enabled use of runtime libraries, interrupts, and parallel processing. When personal computers became popular in the 1980s, operating systems were made for them similar in concept to those used on larger computers.

In the 1940s, the earliest electronic digital systems had no operating systems. Electronic systems of this time were programmed on rows of mechanical switches or by jumper wires on plugboards. These were special-purpose systems that, for example, generated ballistics tables for the military or controlled the printing of payroll checks from data on punched paper cards. After programmable general-purpose computers were invented, machine languages (consisting of strings of the binary digits 0 and 1 on punched paper tape) were introduced that sped up the programming process (Stern, 1981).

In the early 1950s, a computer could execute only one program at a time. Each user had sole use of the computer for a limited period and would arrive at a scheduled time with their program and data on punched paper cards or punched tape. The program would be loaded into the machine, and the machine would be set to work until the program completed or crashed. Programs could generally be debugged via a front panel using toggle switches and panel lights. It is said that Alan Turing was a master of this on the early Manchester Mark 1 machine, and he was already deriving the primitive conception of an operating system from the principles of the universal Turing machine.[11]

OS/360 was used on most IBM mainframe computers beginning in 1966, including computers used by the Apollo program.

Later machines came with libraries of programs, which would be linked to a user's program to assist in operations such as input and output and compiling (generating machine code from human-readable symbolic code). This was the genesis of the modern-day operating system. However, machines still ran a single job at a time. At Cambridge University in England, the job queue was at one time a washing line (clothesline) from which tapes were hung with different colored clothes-pegs to indicate job priority.

An improvement was the Atlas Supervisor. Introduced with the Manchester Atlas in 1962, it is considered by many to be the first recognisable modern operating system.[12] Brinch Hansen described it as "the most significant breakthrough in the history of operating systems."[13]

Mainframes

Through the 1950s, many major features were pioneered in the field of operating systems on mainframe computers, including batch processing, input/output interrupting, buffering, multitasking, spooling, runtime libraries, link-loading, and programs for sorting records in files. These features were included or not included in application software at the option of application programmers, rather than in a separate operating system used by all applications. In 1959, the SHARE Operating System was released as an integrated utility for the IBM 704, and later in the 709 and 7090 mainframes, although it was quickly supplanted by IBSYS/IBJOB on the 709, 7090 and 7094.

During the 1960s, IBM's OS/360 introduced the concept of a single OS spanning an entire product line, which was crucial for the success of the System/360 machines. IBM's current mainframe operating systems are distant descendants of this original system and modern machines are backwards-compatible with applications

written for OS/360.

OS/360 also pioneered the concept that the operating system keeps track of all of the system resources that are used, including program and data space allocation in main memory and file space in secondary storage, and file locking during updates. When a process is terminated for any reason, all of these resources are re-claimed by the operating system.

The alternative CP-67 system for the S/360-67 started a whole line of IBM operating systems focused on the concept of virtual machines. Other operating systems used on IBM S/360 series mainframes included systems developed by IBM: DOS/360[a] (Disk Operating System), TSS/360 (Time Sharing System), TOS/360 (Tape Operating System), BOS/360 (Basic Operating System), and ACP (Airline Control Program), as well as a few non-IBM systems: MTS (Michigan Terminal System), MUSIC (Multi-User System for Interactive Computing), and ORVYL (Stanford Timesharing System).

Control Data Corporation developed the SCOPE operating system in the 1960s, for batch processing. In cooperation with the University of Minnesota, the Kronos and later the NOS operating systems were developed during the 1970s, which supported simultaneous batch and timesharing use. Like many commercial timesharing systems, its interface was an extension of the Dartmouth BASIC operating systems, one of the pioneering efforts in timesharing and programming languages. In the late 1970s, Control Data and the University of Illinois developed the PLATO operating system, which used plasma panel displays and long-distance time sharing networks. Plato was remarkably innovative for its time, featuring real-time chat, and multi-user graphical games.

In 1961, Burroughs Corporation introduced the B5000 with the MCP (Master Control Program) operating system. The B5000 was a stack machine designed to exclusively support high-level languages with no machine language or assembler; indeed, the MCP was the first OS to be written exclusively in a high-level language (ESPOL, a dialect of ALGOL). MCP also introduced many other ground-breaking innovations, such as being the first commercial implementation of virtual memory. During development of the AS/400, IBM made an approach to Burroughs to license MCP to run on the AS/400 hardware. This proposal was declined by Burroughs management to protect its existing hardware production. MCP is still in use today in the Unisys company's ClearPath/MCP line of computers.

UNIVAC, the first commercial computer manufacturer, produced a series of EXEC operating systems. Like all early main-frame systems, this batch-oriented system managed magnetic drums, disks, card readers and line printers. In the 1970s, UNIVAC produced the Real-Time Basic (RTB) system to support large-scale time sharing, also patterned after the Dartmouth BC system.

General Electric and MIT developed General Electric Comprehensive Operating Supervisor (GECOS), which introduced the concept of ringed security privilege levels. After acquisition by Honeywell it was renamed General Comprehensive Operating System (GCOS).

Digital Equipment Corporation developed many operating systems for its various computer lines, including TOPS-10 and TOPS-20 time sharing systems for the 36-bit PDP-10 class systems. Before the widespread use of UNIX, TOPS-10 was a particularly popular system in universities, and in the early ARPANET community. RT-11 was a single-user real-time OS for the PDP-11 class minicomputer, and RSX-11 was the corresponding multi-user OS.

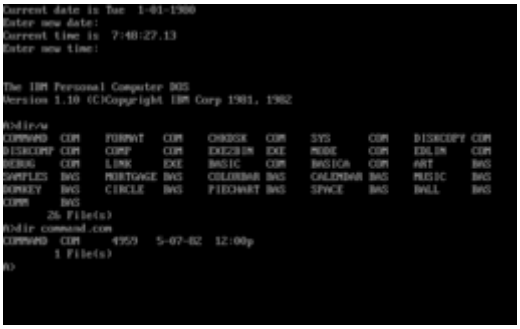
From the late 1960s through the late 1970s, several hardware capabilities evolved that allowed similar or ported software to run on more than one system. Early systems had utilized microprogramming to implement features on their systems in order to permit different underlying computer architectures to appear to be the same as others in a series. In fact, most 360s after the 360/40 (except the 360/165 and 360/168) were microprogrammed implementations.

The enormous investment in software for these systems made since the 1960s caused most of the original computer manufacturers to continue to develop compatible operating systems along with the hardware. Notable supported mainframe operating systems include:

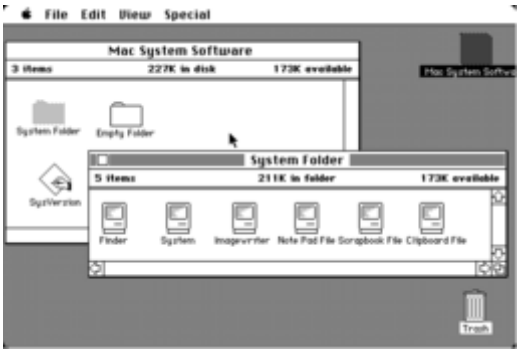
- Burroughs MCP – B5000, 1961 to Unisys Clearpath/MCP, present
- IBM OS/360 – IBM System/360, 1966 to IBM z/OS, present
- IBM CP-67 – IBM System/360, 1967 to IBM z/VM
- UNIVAC EXEC 8 – UNIVAC 1108, 1967, to OS 2200 Unisys Clearpath Dorado, present

Microcomputers

The first microcomputers did not have the capacity or need for the elaborate operating systems that had been developed for mainframes and minis; minimalistic operating systems were developed, often loaded from ROM and known as *monitors*. One notable early disk operating system was CP/M, which was supported on many early microcomputers and was closely imitated by Microsoft's MS-DOS, which became widely popular as the operating system chosen for the IBM PC (IBM's version of it was called IBM DOS or PC DOS). In the 1980s, Apple Computer Inc. (now Apple Inc.) abandoned its popular Apple II series of microcomputers to introduce the Apple Macintosh computer with an innovative graphical user interface (GUI) to the Mac OS.



PC DOS was an early personal computer OS that featured a command line interface.



Mac OS by Apple Computer became the first widespread OS to feature a graphical user interface. Many of its features such as windows and icons would later become commonplace in GUIs.

The introduction of the Intel 80386 CPU chip in October 1985,[14] with 32-bit architecture and paging capabilities, provided personal computers with the ability to run multitasking operating systems like those of earlier minicomputers and mainframes. Microsoft responded to this progress by hiring Dave Cutler, who had developed the VMS operating system for Digital Equipment Corporation. He would lead the development of the Windows NT operating system, which continues to serve as the basis for Microsoft's operating systems line. Steve Jobs, a co-founder of Apple Inc., started NeXT Computer Inc., which developed the NEXTSTEP operating system. NEXTSTEP would later be acquired by Apple Inc. and used, along with code from FreeBSD as the core of Mac OS X (macOS after latest name change).

The GNU Project was started by activist and programmer Richard Stallman with the goal of creating a complete free software replacement to the proprietary UNIX operating system. While the project was highly successful in duplicating the functionality of various parts of UNIX, development of the GNU Hurd kernel proved to be unproductive. In 1991, Finnish computer science student Linus Torvalds, with cooperation from volunteers collaborating over the Internet, released the first version of the Linux kernel. It was soon merged with the GNU user space components and system software to form a complete operating system. Since then, the combination of the two major components has usually been referred to as simply "Linux" by the software industry, a naming convention that Stallman and the Free Software Foundation remain opposed to, preferring the name GNU/Linux. The Berkeley Software Distribution, known as BSD, is the UNIX derivative

distributed by the University of California, Berkeley, starting in the 1970s. Freely distributed and ported to many minicomputers, it eventually also gained a following for use on PCs, mainly as FreeBSD, NetBSD and OpenBSD.

Examples

Unix and Unix-like operating systems

Evolution of Unix systems

Unix was originally written in assembly language.[15] Ken Thompson wrote B, mainly based on BCPL, based on his experience in the MULTICS project. B was replaced by C, and Unix, rewritten in C, developed into a large, complex family of inter-related operating systems which have been influential in every modern operating system (see History).

The *Unix-like* family is a diverse group of operating systems, with several major sub-categories including System V, BSD, and Linux. The name "UNIX" is a trademark of The Open Group which licenses it for use with any operating system that has been shown to conform to their definitions. "UNIX-like" is commonly used to refer to the large set of operating systems which resemble the original UNIX.

Unix-like systems run on a wide variety of computer architectures. They are used heavily for servers in business, as well as workstations in academic and engineering environments. Free UNIX variants, such as Linux and BSD, are popular in these areas.

Four operating systems are certified by The Open Group (holder of the Unix trademark) as Unix. HP's HP-UX and IBM's AIX are both descendants of the original System V Unix and are designed to run only on their respective vendor's hardware. In contrast, Sun Microsystems's Solaris can run on multiple types of hardware, including x86 and Sparc servers, and PCs. Apple's macOS, a replacement for Apple's earlier (non-Unix) Mac OS, is a hybrid kernel-based BSD variant derived from NeXTSTEP, Mach, and FreeBSD.

Unix interoperability was sought by establishing the POSIX standard. The POSIX standard can be applied to any operating system, although it was originally created for various Unix variants.

BSD and its descendants

The first server for the World Wide Web ran on NeXTSTEP, based on BSD.

A subgroup of the Unix family is the Berkeley Software Distribution family, which includes FreeBSD, NetBSD, and OpenBSD. These operating systems are most commonly found on webservers, although they can also function as a personal computer OS. The Internet owes much of its existence to BSD, as many of the protocols now commonly used by computers to connect, send and receive data over a network were widely implemented and refined in BSD. The World Wide Web was also first demonstrated on a number of computers running an OS based on BSD called NeXTSTEP.

In 1974, University of California, Berkeley installed its first Unix system. Over time, students and staff in the computer science department there began adding new programs to make

things easier, such as text editors. When Berkeley received new VAX computers in 1978 with Unix installed, the school's undergraduates modified Unix even more in order to take advantage of the computer's hardware possibilities. The Defense Advanced Research Projects Agency of the US Department of Defense took interest, and decided to fund the project. Many schools, corporations, and government organizations took notice and started to use Berkeley's version of Unix instead of the official one distributed by AT&T.

Steve Jobs, upon leaving Apple Inc. in 1985, formed NeXT Inc., a company that manufactured high-end computers running on a variation of BSD called NeXTSTEP. One of these computers was used by Tim Berners-Lee as the first webserver to create the World Wide Web.

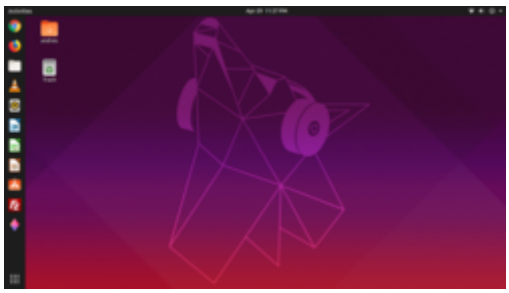
Developers like Keith Bostic encouraged the project to replace any non-free code that originated with Bell Labs. Once this was done, however, AT&T sued. After two years of legal disputes, the BSD project spawned a number of free derivatives, such as NetBSD and FreeBSD (both in 1993), and OpenBSD (from NetBSD in 1995).

macOS

macOS (formerly "Mac OS X" and later "OS X") is a line of open core graphical operating systems developed, marketed, and sold by Apple Inc., the latest of which is pre-loaded on all currently shipping Macintosh computers. macOS is the successor to the original classic Mac OS, which had been Apple's primary operating system since 1984. Unlike its predecessor, macOS is a UNIX operating system built on technology that had been developed at NeXT through the second half of the 1980s and up until Apple purchased the company in early 1997. The operating system was first released in 1999 as Mac OS X Server 1.0, followed in March 2001 by a client version (Mac OS X v10.0 "Cheetah"). Since then, six more distinct "client" and "server" editions of macOS have been released, until the two were merged in OS X 10.7 "Lion".

Prior to its merging with macOS, the server edition – macOS Server – was architecturally identical to its desktop counterpart and usually ran on Apple's line of Macintosh server hardware. macOS Server included work group management and administration software tools that provide simplified access to key network services, including a mail transfer agent, a Samba server, an LDAP server, a domain name server, and others. With Mac OS X v10.7 Lion, all server aspects of Mac OS X Server have been integrated into the client version and the product re-branded as "OS X" (dropping "Mac" from the name). The server tools are now offered as an application.[16]

Linux



Ubuntu, desktop Linux distribution

The Linux kernel originated in 1991, as a project of Linus Torvalds, while a university student in Finland. He posted information about his project on a newsgroup for computer students and programmers, and received support and assistance from volunteers who succeeded in creating a complete and functional kernel.

Linux is Unix-like, but was developed without any Unix code, unlike BSD and its variants. Because of its open license model, the Linux kernel code is available for study and modification, which resulted in its use on a wide range of computing machinery from supercomputers to smart-watches. Although estimates suggest that Linux is used on only 1.82% of all "desktop" (or laptop) PCs,[21] it has been widely adopted for use in servers[22] and embedded systems[23] such as cell phones. Linux has superseded Unix on many platforms and is used on most supercomputers including the top 385.[24] Many of the same computers are also on Green500 (but in different order), and Linux runs on the top 10. Linux is also commonly used on other small energy-efficient computers,

such as smartphones and smartwatches. The Linux kernel is used in some popular distributions, such as Red Hat, Debian, Ubuntu, Linux Mint and Google's Android, Chrome OS, and Chromium OS.

Linux, a unix-like operating system was first time released on September 17, 1991, by Linus Torvalds.[17][18][19] Picture of Tux the penguin, mascot of Linux.[20]

Microsoft Windows

Microsoft Windows is a family of proprietary operating systems designed by Microsoft Corporation and primarily targeted to Intel architecture based computers, with an estimated 88.9 percent total usage share on Web connected computers.[21][25][26][27] The latest version is Windows 10.

In 2011, Windows 7 overtook Windows XP as most common version in use.[28][29][30]

Microsoft Windows was first released in 1985, as an operating environment running on top of MS-DOS, which was the standard operating system shipped on most Intel architecture personal computers at the time. In 1995, Windows 95 was released which only used MS-DOS as a bootstrap. For backwards compatibility, Win9x could run real-mode MS-DOS[31][32] and 16-bit Windows 3.x[33] drivers. Windows ME, released in 2000, was the last version in the Win9x family. Later versions have all been based on the Windows NT kernel. Current client versions of Windows run on IA-32, x86-64 and 32-bit ARM microprocessors.[34] In addition Itanium is still supported in older server version Windows Server 2008 R2. In the past, Windows NT supported additional architectures.

Server editions of Windows are widely used. In recent years, Microsoft has expended significant capital in an effort to promote the use of Windows as a server operating system. However, Windows' usage on servers is not as widespread as on personal computers as Windows competes against Linux and BSD for server market share.[35][36]

ReactOS is a Windows-alternative operating system, which is being developed on the principles of Windows – without using any of Microsoft's code.

Other

There have been many operating systems that were significant in their day but are no longer so, such as AmigaOS; OS/2 from IBM and Microsoft; classic Mac OS, the non-Unix precursor to Apple's macOS; BeOS; XTS-300; RISC OS; MorphOS; Haiku; BareMetal and FreeMint. Some are still used in niche markets and continue to be developed as minority platforms for enthusiast communities and specialist applications. OpenVMS, formerly from DEC, is still under active development by VMS Software Inc. Yet other operating systems are used almost exclusively in academia, for operating systems education or to do research on operating system concepts. A typical example of a system that fulfills both roles is MINIX, while for example Singularity is used purely for research. Another example is the Oberon System designed at ETH Zürich by Niklaus Wirth, Jürg Gutknecht and a group of students at the former Computer Systems Institute in the 1980s. It was used mainly for research, teaching, and daily work in Wirth's group.

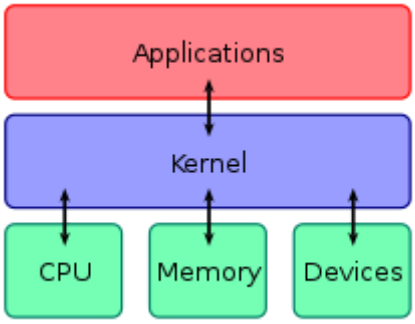
Other operating systems have failed to win significant market share, but have introduced innovations that have influenced mainstream operating systems, not least Bell Labs' Plan 9.

Components

The components of an operating system all exist in order to make the different parts of a computer work together. All user software needs to go through the operating system in order to use any of the hardware, whether it be as simple as a mouse or keyboard or as complex as an Internet component.

Kernel

With the aid of the firmware and device drivers, the kernel provides the most basic level of control over all of the computer's hardware devices. It manages memory access for programs in the RAM, it determines which programs get access to which hardware resources, it sets up or resets the CPU's operating states for optimal operation at all times, and it organizes the data for long-term non-volatile storage with file systems on such media as disks, tapes, flash memory, etc.



A kernel connects the application software to the hardware of a computer.

Program execution

The operating system provides an interface between an application program and the computer hardware, so that an application program can interact with the hardware only by obeying rules and procedures programmed into the operating system. The operating system is also a set of services which simplify development and execution of application programs. Executing an application program involves the creation of a process by the operating system kernel which assigns memory space and other resources, establishes a priority for the process in multi-tasking systems, loads program binary code into memory, and initiates execution of the application program which then interacts with the user and with hardware devices.

Interrupts

Interrupts are central to operating systems, as they provide an efficient way for the operating system to interact with and react to its environment. The alternative – having the operating system "watch" the various sources of input for events (polling) that require action – can be found in older systems with very small stacks (50 or 60 bytes) but is unusual in modern systems with large stacks. Interrupt-based programming is directly supported by most modern CPUs. Interrupts provide a computer with a way of automatically saving local register contexts, and running specific code in response to events. Even very basic computers support hardware interrupts, and allow the programmer to specify code which may be run when that event takes place.

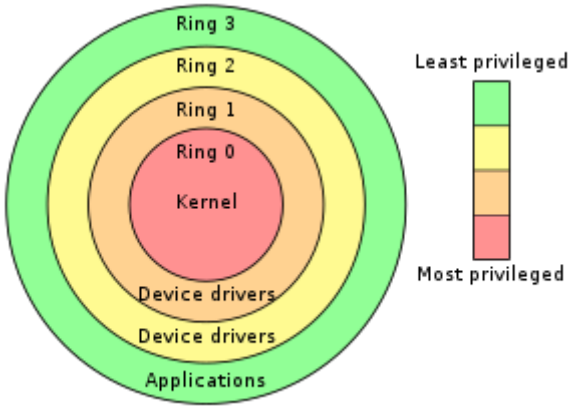
When an interrupt is received, the computer's hardware automatically suspends whatever program is currently running, saves its status, and runs computer code previously associated with the interrupt; this is analogous to placing a bookmark in a book in response to a phone call. In modern operating systems, interrupts are handled by the operating system's kernel. Interrupts may come from either the computer's hardware or the running program.

When a hardware device triggers an interrupt, the operating system's kernel decides how to deal with this event, generally by running some processing code. The amount of code being run depends on the priority of the interrupt (for example: a person usually responds to a smoke detector alarm before answering the phone). The processing of hardware interrupts is a task that is usually delegated to software called a device driver, which may be part of the operating system's kernel, part of another program, or both. Device drivers may then relay information to a running program by various means.

A program may also trigger an interrupt to the operating system. If a program wishes to access hardware, for example, it may interrupt the operating system's kernel, which causes control to be passed back to the kernel. The kernel then processes the request. If a program wishes additional resources (or wishes to shed resources) such as memory, it triggers an interrupt to get the kernel's attention.

Modes

Modern microprocessors (CPU or MPU) support multiple modes of operation. CPUs with this capability offer at least two modes: user mode and supervisor mode. In general terms, supervisor mode operation allows unrestricted access to all machine resources, including all MPU instructions. User mode operation sets limits on instruction use and typically disallows direct access to machine resources. CPUs might have other modes similar to user mode as well, such as the virtual modes in order to emulate older processor types, such as 16-bit processors on a 32-bit one, or 32-bit processors on a 64-bit one.



Privilege rings for the x86 microprocessor architecture available in protected mode. Operating systems determine which processes run in each mode.

At power-on or reset, the system begins in supervisor mode. Once an operating system kernel has been loaded and started, the boundary between user mode and supervisor mode (also known as kernel mode) can be established.

Supervisor mode is used by the kernel for low level tasks that need unrestricted access to hardware, such as controlling how memory is accessed, and communicating with devices such as disk drives and video display devices. User mode, in contrast, is used for almost everything else. Application programs, such as word processors and database managers, operate within user mode, and can only access machine resources by turning control over to the kernel, a process which causes a switch to supervisor mode. Typically, the transfer of control to the kernel is achieved by executing a software interrupt instruction, such as the Motorola 68000 TRAP instruction. The software interrupt causes the microprocessor to switch from user mode to supervisor mode and begin executing code that allows the kernel to take control.

In user mode, programs usually have access to a restricted set of microprocessor instructions, and generally cannot execute any instructions that could potentially cause disruption to the system's operation. In supervisor mode, instruction execution restrictions are typically removed, allowing the kernel unrestricted access to all machine resources.

The term "user mode resource" generally refers to one or more CPU registers, which contain information that the running program isn't allowed to alter. Attempts to alter these resources generally causes a switch to supervisor mode, where the operating system can deal with the illegal operation the program was attempting, for example, by forcibly terminating ("killing") the program).

Memory management

Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs. This ensures that a program does not interfere with memory already in use by another program. Since programs time share, each program must have independent access to memory.

Cooperative memory management, used by many early operating systems, assumes that all programs make voluntary use of the kernel's memory manager, and do not exceed their allocated memory. This system of memory management is almost never seen any more, since programs often contain bugs which can cause them to exceed their allocated memory. If a program fails, it may cause memory used by one or more other programs to be affected or overwritten. Malicious programs or viruses may purposefully alter another program's memory, or may affect the operation of the operating system itself. With cooperative memory management, it takes only one misbehaved program to crash the system.

Memory protection enables the kernel to limit a process' access to the computer's memory. Various methods of memory protection exist, including memory segmentation and paging. All methods require some level of hardware support (such as the 80286 MMU), which doesn't exist in all computers.

In both segmentation and paging, certain protected mode registers specify to the CPU what memory address it should allow a running program to access. Attempts to access other addresses trigger an interrupt which cause the CPU to re-enter supervisor mode, placing the kernel in charge. This is called a segmentation violation or Seg-V for short, and since it is both difficult to assign a meaningful result to such an operation, and because it is usually a sign of a misbehaving program, the kernel generally resorts to terminating the offending program, and reports the error.

Windows versions 3.1 through ME had some level of memory protection, but programs could easily circumvent the need to use it. A general protection fault would be produced, indicating a segmentation violation had occurred; however, the system would often crash anyway.

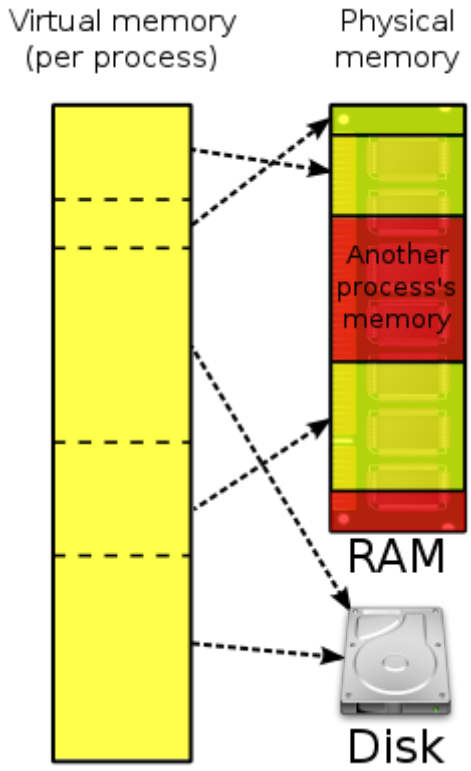
Virtual memory

The use of virtual memory addressing (such as paging or segmentation) means that the kernel can choose what memory each program may use at any given time, allowing the operating system to use the same memory locations for multiple tasks.

If a program tries to access memory that isn't in its current range of accessible memory, but nonetheless has been allocated to it, the kernel is interrupted in the same way as it would if the program were to exceed its allocated memory. (See section on memory management.) Under UNIX this kind of interrupt is referred to as a page fault.

When the kernel detects a page fault it generally adjusts the virtual memory range of the program which triggered it, granting it access to the memory requested. This gives the kernel discretionary power over where a particular application's memory is stored, or even whether or not it has actually been allocated yet.

In modern operating systems, memory which is accessed less frequently can be temporarily stored on disk or other media to make that space available for use by other programs. This is called swapping, as an area of memory can be used by multiple programs, and what that memory area contains can be swapped or exchanged on demand.



Many operating systems can "trick" programs into using memory scattered around the hard disk and RAM as if it is one continuous chunk of memory, called virtual memory.

"Virtual memory" provides the programmer or the user with the perception that there is a much larger amount of RAM in the computer than is really there.[37]

Multitasking

Multitasking refers to the running of multiple independent computer programs on the same computer; giving the appearance that it is performing the tasks at the same time. Since most computers can do at most one or two things at one time, this is generally done via time-sharing, which means that each program uses a share of the computer's time to execute.

An operating system kernel contains a scheduling program which determines how much time each process spends executing, and in which order execution control should be passed to programs. Control is passed to a process by the kernel, which allows the program access to the CPU and memory. Later, control is returned to the kernel through some mechanism, so that another program may be allowed to use the CPU. This so-called passing of control between the kernel and applications is called a context switch.

An early model which governed the allocation of time to programs was called cooperative multitasking. In this model, when control is passed to a program by the kernel, it may execute for as long as it wants before explicitly returning control to the kernel. This means that a malicious or malfunctioning program may not only prevent any other programs from using the CPU, but it can hang the entire system if it enters an infinite loop.

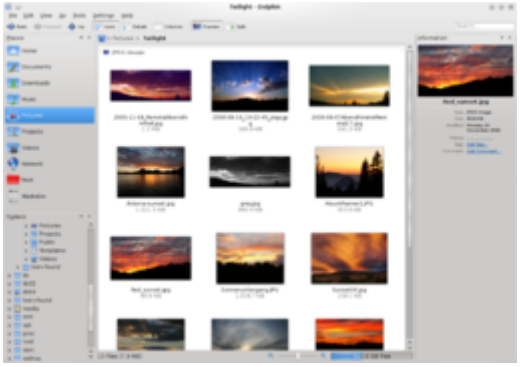
Modern operating systems extend the concepts of application preemption to device drivers and kernel code, so that the operating system has preemptive control over internal run-times as well.

The philosophy governing preemptive multitasking is that of ensuring that all programs are given regular time on the CPU. This implies that all programs must be limited in how much time they are allowed to spend on the CPU without being interrupted. To accomplish this, modern operating system kernels make use of a timed interrupt. A protected mode timer is set by the kernel which triggers a return to supervisor mode after the specified time has elapsed. (See above sections on Interrupts and Dual Mode Operation.)

On many single user operating systems cooperative multitasking is perfectly adequate, as home computers generally run a small number of well tested programs. The AmigaOS is an exception, having preemptive multitasking from its first version. Windows NT was the first version of Microsoft Windows which enforced preemptive multitasking, but it didn't reach the home user market until Windows XP (since Windows NT was targeted at professionals).

Disk access and file systems

Access to data stored on disks is a central feature of all operating systems. Computers store data on disks using files, which are structured in specific ways in order to allow for faster access, higher reliability, and to make better use of the drive's available space. The specific way in which files are stored on a disk is called a file system, and enables files to have names and attributes. It also allows them to be stored in a hierarchy of directories or folders arranged in a directory tree.



File systems allow users and programs to organize and sort files on a computer, often through the use of directories (or "folders").

Early operating systems generally supported a single type of disk drive and only one kind of file system. Early file systems were limited in their capacity, speed, and in the kinds of file

names and directory structures they could use. These limitations often reflected limitations in the operating systems they were designed for, making it very difficult for an operating system to support more than one file system.

While many simpler operating systems support a limited range of options for accessing storage systems, operating systems like UNIX and Linux support a technology known as a virtual file system or VFS. An operating system such as UNIX supports a wide array of storage devices, regardless of their design or file systems, allowing them to be accessed through a common application programming interface (API). This makes it unnecessary for programs to have any knowledge about the device they are accessing. A VFS allows the operating system to provide programs with access to an unlimited number of devices with an infinite variety of file systems installed on them, through the use of specific device drivers and file system drivers.

A connected storage device, such as a hard drive, is accessed through a device driver. The device driver understands the specific language of the drive and is able to translate that language into a standard language used by the operating system to access all disk drives. On UNIX, this is the language of block devices.

When the kernel has an appropriate device driver in place, it can then access the contents of the disk drive in raw format, which may contain one or more file systems. A file system driver is used to translate the commands used to access each specific file system into a standard set of commands that the operating system can use to talk to all file systems. Programs can then deal with these file systems on the basis of filenames, and directories/folders, contained within a hierarchical structure. They can create, delete, open, and close files, as well as gather various information about them, including access permissions, size, free space, and creation and modification dates.

Various differences between file systems make supporting all file systems difficult. Allowed characters in file names, case sensitivity, and the presence of various kinds of file attributes makes the implementation of a single interface for every file system a daunting task. Operating systems tend to recommend using (and so support natively) file systems specifically designed for them; for example, NTFS in Windows and ext3 and ReiserFS in Linux. However, in practice, third party drivers are usually available to give support for the most widely used file systems in most general-purpose operating systems (for example, NTFS is available in Linux through NTFS-3g, and ext2/3 and ReiserFS are available in Windows through third-party software).

Support for file systems is highly varied among modern operating systems, although there are several common file systems which almost all operating systems include support and drivers for. Operating systems vary on file system support and on the disk formats they may be installed on. Under Windows, each file system is usually limited in application to certain media; for example, CDs must use ISO 9660 or UDF, and as of Windows Vista, NTFS is the only file system which the operating system can be installed on. It is possible to install Linux onto many types of file systems. Unlike other operating systems, Linux and UNIX allow any file system to be used regardless of the media it is stored in, whether it is a hard drive, a disc (CD, DVD...), a USB flash drive, or even contained within a file located on another file system.

Device drivers

A device driver is a specific type of computer software developed to allow interaction with hardware devices. Typically this constitutes an interface for communicating with the device, through the specific computer bus or communications subsystem that the hardware is connected to, providing commands to and/or receiving data from the device, and on the other end, the requisite interfaces to the operating system and software applications. It is a specialized hardware-dependent computer program which is also operating system specific that enables another program, typically an operating system or applications software package or computer program running under the operating system kernel, to interact transparently with a hardware device, and usually provides the requisite interrupt handling necessary for any necessary asynchronous time-dependent hardware interfacing needs.

The key design goal of device drivers is abstraction. Every model of hardware (even within the same class of device) is different. Newer models also are released by manufacturers that provide more reliable or better performance and these newer models are often controlled differently. Computers and their operating systems cannot be expected to know how to control every device, both now and in the future. To solve this problem, operating systems essentially dictate how every type of device should be controlled. The function of the device driver is then to translate these operating system mandated function calls into device specific calls. In theory a new device, which is controlled in a new manner, should function correctly if a suitable driver is available. This new driver ensures that the device appears to operate as usual from the operating system's point of view.

Under versions of Windows before Vista and versions of Linux before 2.6, all driver execution was co-operative, meaning that if a driver entered an infinite loop it would freeze the system. More recent revisions of these operating systems incorporate kernel preemption, where the kernel interrupts the driver to give it tasks, and then separates itself from the process until it receives a response from the device driver, or gives it more tasks to do.

Networking

Currently most operating systems support a variety of networking protocols, hardware, and applications for using them. This means that computers running dissimilar operating systems can participate in a common network for sharing resources such as computing, files, printers, and scanners using either wired or wireless connections. Networks can essentially allow a computer's operating system to access the resources of a remote computer to support the same functions as it could if those resources were connected directly to the local computer. This includes everything from simple communication, to using networked file systems or even sharing another computer's graphics or sound hardware. Some network services allow the resources of a computer to be accessed transparently, such as SSH which allows networked users direct access to a computer's command line interface.

Client/server networking allows a program on a computer, called a client, to connect via a network to another computer, called a server. Servers offer (or host) various services to other network computers and users. These services are usually provided through ports or numbered access points beyond the server's IP address. Each port number is usually associated with a maximum of one running program, which is responsible for handling requests to that port. A daemon, being a user program, can in turn access the local hardware resources of that computer by passing requests to the operating system kernel.

Many operating systems support one or more vendor-specific or open networking protocols as well, for example, SNA on IBM systems, DECnet on systems from Digital Equipment Corporation, and Microsoft-specific protocols (SMB) on Windows. Specific protocols for specific tasks may also be supported such as NFS for file access. Protocols like ESound, or esd can be easily extended over the network to provide sound from local applications, on a remote system's sound hardware.

Security

A computer being secure depends on a number of technologies working properly. A modern operating system provides access to a number of resources, which are available to software running on the system, and to external devices like networks via the kernel.[38]

The operating system must be capable of distinguishing between requests which should be allowed to be processed, and others which should not be processed. While some systems may simply distinguish between "privileged" and "non-privileged", systems commonly have a form of requester *identity*, such as a user name. To establish identity there may be a process of *authentication*. Often a username must be quoted, and each username may have a password. Other methods of authentication, such as magnetic cards or biometric data,

might be used instead. In some cases, especially connections from the network, resources may be accessed with no authentication at all (such as reading files over a network share). Also covered by the concept of requester **identity** is *authorization*; the particular services and resources accessible by the requester once logged into a system are tied to either the requester's user account or to the variously configured groups of users to which the requester belongs.

In addition to the allow or disallow model of security, a system with a high level of security also offers auditing options. These would allow tracking of requests for access to resources (such as, "who has been reading this file?"). Internal security, or security from an already running program is only possible if all possibly harmful requests must be carried out through interrupts to the operating system kernel. If programs can directly access hardware and resources, they cannot be secured.

External security involves a request from outside the computer, such as a login at a connected console or some kind of network connection. External requests are often passed through device drivers to the operating system's kernel, where they can be passed onto applications, or carried out directly. Security of operating systems has long been a concern because of highly sensitive data held on computers, both of a commercial and military nature. The United States Government Department of Defense (DoD) created the *Trusted Computer System Evaluation Criteria* (TCSEC) which is a standard that sets basic requirements for assessing the effectiveness of security. This became of vital importance to operating system makers, because the TCSEC was used to evaluate, classify and select trusted operating systems being considered for the processing, storage and retrieval of sensitive or classified information.

Network services include offerings such as file sharing, print services, email, web sites, and file transfer protocols (FTP), most of which can have compromised security. At the front line of security are hardware devices known as firewalls or intrusion detection/prevention systems. At the operating system level, there are a number of software firewalls available, as well as intrusion detection/prevention systems. Most modern operating systems include a software firewall, which is enabled by default. A software firewall can be configured to allow or deny network traffic to or from a service or application running on the operating system. Therefore, one can install and be running an insecure service, such as Telnet or FTP, and not have to be threatened by a security breach because the firewall would deny all traffic trying to connect to the service on that port.

An alternative strategy, and the only sandbox strategy available in systems that do not meet the Popek and Goldberg virtualization requirements, is where the operating system is not running user programs as native code, but instead either emulates a processor or provides a host for a p-code based system such as Java.

Internal security is especially relevant for multi-user systems; it allows each user of the system to have private files that the other users cannot tamper with or read. Internal security is also vital if auditing is to be of any use, since a program can potentially bypass the operating system, inclusive of bypassing auditing.

User interface

Every computer that is to be operated by an individual requires a user interface. The user interface is usually referred to as a shell and is essential if human interaction is to be supported. The user interface views the directory structure and requests services from the operating system that will acquire data from input hardware devices, such as a keyboard, mouse or credit card reader, and requests operating system services to display prompts, status messages and such on output hardware devices, such as a video monitor or printer. The two most common forms of a user interface have historically been the command-line interface, where computer commands are typed out line-by-line, and the graphical user interface, where a visual environment (most commonly a WIMP) is present.

Graphical user interfaces

A screenshot of the KDE Plasma 5 graphical user interface. Programs take the form of images on the screen, and the files, folders (directories), and applications take the form of icons and symbols. A mouse is used to navigate the computer.

A screenshot of the Bash command line. Each command is typed out after the 'prompt', and then its output appears below, working its way down the screen. The current command prompt is at the bottom.

Most of the modern computer systems support graphical user interfaces (GUI), and often include them. In some computer systems, such as the original implementation of the classic Mac OS, the GUI is integrated into the kernel.

While technically a graphical user interface is not an operating system service, incorporating support for one into the operating system kernel can allow the GUI to be more responsive by reducing the number of context switches required for the GUI to perform its output functions. Other operating systems are modular, separating the graphics subsystem from the kernel and the Operating System. In the 1980s UNIX, VMS and many others had operating systems that were built this way. Linux and macOS are also built this way. Modern releases of Microsoft Windows such as Windows Vista implement a graphics subsystem that is mostly in user-space; however the graphics drawing routines of versions between Windows NT 4.0 and Windows Server 2003 exist mostly in kernel space. Windows 9x had very little distinction between the interface and the kernel.

Many computer operating systems allow the user to install or create any user interface they desire. The X Window System in conjunction with GNOME or KDE Plasma 5 is a commonly found setup on most Unix and Unix-like (BSD, Linux, Solaris) systems. A number of Windows shell replacements have been released for Microsoft Windows, which offer alternatives to the included Windows shell, but the shell itself cannot be separated from Windows.

Numerous Unix-based GUIs have existed over time, most derived from X11. Competition among the various vendors of Unix (HP, IBM, Sun) led to much fragmentation, though an effort to standardize in the 1990s to COSE and CDE failed for various reasons, and were eventually eclipsed by the widespread adoption of GNOME and K Desktop Environment. Prior to free software-based toolkits and desktop environments, Motif was the prevalent toolkit/desktop combination (and was the basis upon which CDE was developed).

Graphical user interfaces evolve over time. For example, Windows has modified its user interface almost every time a new major version of Windows is released, and the Mac OS GUI changed dramatically with the introduction of Mac OS X in 1999.[39]

Real-time operating systems

A real-time operating system (RTOS) is an operating system intended for applications with fixed deadlines (real-time computing). Such applications include some small embedded systems, automobile engine controllers, industrial robots, spacecraft, industrial control, and some large-scale computing systems.

An early example of a large-scale real-time operating system was Transaction Processing Facility developed by American Airlines and IBM for the Sabre Airline Reservations System.

Embedded systems that have fixed deadlines use a real-time operating system such as VxWorks, PikeOS, eCos, QNX, MontaVista Linux and RTLinux. Windows CE is a real-time operating system that shares similar APIs to desktop Windows but shares none of desktop Windows' codebase.[40] Symbian OS also has an RTOS kernel (EKA2) starting with version 8.0b.

Some embedded systems use operating systems such as Palm OS, BSD, and Linux, although such operating systems do not support real-time computing.

Operating system development as a hobby

A hobby operating system may be classified as one whose code has not been directly derived from an existing operating system, and has few users and active developers.

In some cases, hobby development is in support of a "homebrew" computing device, for example, a simple single-board computer powered by a 6502 microprocessor. Or, development may be for an architecture already in widespread use. Operating system development may come from entirely new concepts, or may commence by modeling an existing operating system. In either case, the hobbyist is his/her own developer, or may interact with a small and sometimes unstructured group of individuals who have like interests.

Examples of a hobby operating system include Syllable and TempleOS.

Diversity of operating systems and portability

Application software is generally written for use on a specific operating system, and sometimes even for specific hardware. When porting the application to run on another OS, the functionality required by that application may be implemented differently by that OS (the names of functions, meaning of arguments, etc.) requiring the application to be adapted, changed, or otherwise maintained.

Unix was the first operating system not written in assembly language, making it very portable to systems different from its native PDP-11.[41]

This cost in supporting operating systems diversity can be avoided by instead writing applications against software platforms such as Java or Qt. These abstractions have already borne the cost of adaptation to specific operating systems and their system libraries.

Another approach is for operating system vendors to adopt standards. For example, POSIX and OS abstraction layers provide commonalities that reduce porting costs.

Market share

See also

- Comparison of operating systems
- Crash (computing)
- Hypervisor
- Interruptible operating system
- List of important publications in operating systems

- List of operating systems
- List of pioneers in computer science
- Live CD
- Glossary of operating systems terms
- Microcontroller
- Mobile device
- Mobile operating system
- Network operating system
- Object-oriented operating system
- Operating System Projects
- System Commander
- System image
- Timeline of operating systems

Notes

a. A combination of DOS/360 and emulation software was known as Compatibility Operating System (COS).

References

1. Stallings (2005). *Operating Systems, Internals and Design Principles*. Pearson: Prentice Hall. p. 6.
2. Dhotre, I.A. (2009). *Operating Systems*. Technical Publications. p. 1.
3. "VII. Special-Purpose Systems - Operating System Concepts, Seventh Edition [Book]" (https://www.oreilly.com/library/view/operating-system-concepts/9780471694663/pt07.html). *www.oreilly.com*.
4. "Desktop Operating System Market Share Worldwide" (https://gs.statcounter.com/os-market-share/desktop/worldwide/). *StatCounter Global Stats*. Retrieved 31 October 2020.
5. "Mobile & Tablet Operating System Market Share Worldwide" (https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/). *StatCounter Global Stats*. Retrieved 31 October 2020.
6. "Strategy Analytics: Android Captures Record 88 Percent Share of Global Smartphone Shipments in Q3 2016" (http://www.businesswire.com/news/home/20161102006440/en/Strategy-Analytics-Android-Captures-Record-88-Percent). 2 November 2016. Archived (https://web.archive.org/web/20161105223332/http://www.businesswire.com/news/home/20161102006440/en/Strategy-Analytics-Android-Captures-Record-88-Percent) from the original on 5 November 2016.
7. "Special-Purpose Operating Systems - RWTH AACHEN UNIVERSITY Institute for Automation of Complex Power Systems - English" (https://www.acs.eonerc.rwth-aachen.de/cms/E-ON-ERC-ACS/Studium/Lehrveranstaltungen/~lrhs/Spezial-Betriebssysteme/?lidx=1). *www.acs.eonerc.rwth-aachen.de*.
8. Lorch, Jacob R., and Alan Jay Smith. "Reducing processor power consumption by improving processor time management in a single-user operating system." Proceedings of the 2nd annual international conference on Mobile computing and networking. ACM, 1996.
9. Mishra, B.; Singh, N.; Singh, R. (2014). "Master-slave group based model for co-ordinator selection, an improvement of bully algorithm". *International Conference on Parallel, Distributed and Grid Computing (PDGC)*. pp. 457–460. doi:10.1109/PDGC.2014.7030789 (https://doi.org/10.1109%2FPDGC.2014.7030789). ISBN 978-1-4799-7682-9. S2CID 13887160 (https://api.semanticscholar.org/CorpusID:13887160).

10. Gagne, Silberschatz Galvin (2012). *Operating Systems Concepts*. New York: Wiley. p. 716. ISBN 978-1118063330.
11. Hansen, Per Brinch, ed. (2001). *Classic Operating Systems* (https://books.google.com/books?id=-PDPBvIPYBkC&pg=PP1). Springer. pp. 4–7. ISBN 0-387-95113-X.
12. Lavington, Simon (1998). *A History of Manchester Computers* (2nd ed.). Swindon: The British Computer Society. pp. 50–52. ISBN 978-1-902505-01-5.
13. Brinch Hansen, Per (2000). *Classic Operating Systems: From Batch Processing to Distributed Systems*. Springer-Verlag.
14. "Intel® Microprocessor Quick Reference Guide - Year" (http://www.intel.com/pressroom/kits/quickrefyr.htm#1985). *www.intel.com*. Archived (https://web.archive.org/web/20160425001839/http://www.intel.com/pressroom/kits/quickrefyr.htm#1985) from the original on 25 April 2016. Retrieved 24 April 2016.
15. Ritchie, Dennis. "Unix Manual, first edition" (https://web.archive.org/web/20080518013206/http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html). Lucent Technologies. Archived from the original (http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html) on 18 May 2008. Retrieved 22 November 2012.
16. "OS X Mountain Lion – Move your Mac even further ahead" (https://www.apple.com/macosx/lion/). Apple. Archived (https://web.archive.org/web/20110523090205/http://www.apple.com/macosx/lion/) from the original on 23 May 2011. Retrieved 7 August 2012.
17. "Twenty Years of Linux according to Linus Torvalds" (https://www.zdnet.com/article/twenty-years-of-linux-according-to-linus-torvalds/). ZDNet. April 13, 2011. Archived (https://web.archive.org/web/20160919232940/http://www.zdnet.com/article/twenty-years-of-linux-according-to-linus-torvalds/) from the original on September 19, 2016. Retrieved September 19, 2016.
18. Linus Benedict Torvalds (5 October 1991). "Free minix-like kernel sources for 386-AT" (https://groups.google.com/group/comp.os.minix/msg/2194d253268b0a1b?pli=1). Newsgroup: comp.os.minix (news:comp.os.minix). Retrieved 30 September 2011.
19. "What Is Linux: An Overview of the Linux Operating System" (https://medium.com/@theinfovalley097/what-is-linux-an-overview-of-the-linux-operating-system-77bc7421c7e5?sk=b80b38575284317290c86e56001e43b1). Medium. Retrieved December 21, 2019.
20. Linux Online (2008). "Linux Logos and Mascots" (https://web.archive.org/web/20100815085106/http://www.linux.org/info/logos.html). Archived from the original (http://www.linux.org/info/logos.html) on 15 August 2010. Retrieved 11 August 2009.
21. "Top 5 Operating Systems from January to April 2011" (http://gs.statcounter.com/#os-ww-monthly-201101-201104-bar). StatCounter. October 2009. Archived (https://archive.today/20120526/http://gs.statcounter.com/%23mobile_browser-ww-monthly-201012-201111-bar#os-ww-monthly-201101-201104-bar) from the original on 26 May 2012. Retrieved 5 November 2009.
22. "IDC report into Server market share" (https://web.archive.org/web/20120927155332/http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22360110&sectionId=null&elementId=null&pageType=SYNOPSIS). Idc.com. Archived from the original (http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22360110&sectionId=null&elementId=null&pageType=SYNOPSIS) on 27 September 2012. Retrieved 7 August 2012.
23. LinuxDevices Staff (23 April 2008). "Linux still top embedded OS" (https://web.archive.org/web/20160419010154/http://archive.linuxgizmos.com/linux-still-top-embedded-os/). *LinuxGizmos.com*. Archived from the original (http://archive.linuxgizmos.com/linux-still-top-embedded-os/) on 19 April 2016. Retrieved 5 April 2016.
24. "Sublist Generator" (http://www.top500.org/statistics/sublist/). Top500.org. Retrieved 6 February 2017.
25. "Global Web Stats" (http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8). Net Market Share, Net Applications. May 2011. Archived (https://web.archive.org/web/20100125022803/http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8) from the original on 25 January 2010. Retrieved 7 May 2011.

26. "Global Web Stats" (http://www.w3counter.com/globalstats.php). W3Counter, Awio Web Services. September 2009. Archived (https://archive.today/20120628/http://www.w3counter.com/globalstats.php) from the original on 28 June 2012. Retrieved 24 October 2009.
27. "Operating System Market Share" (http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8). Net Applications. October 2009. Archived (https://web.archive.org/web/20100125022803/http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8) from the original on 25 January 2010. Retrieved 5 November 2009.
28. "w3schools.com OS Platform Statistics" (http://www.w3schools.com/browsers/browsers_os.asp). Archived (https://web.archive.org/web/20110805133049/http://w3schools.com/browsers/browsers_os.asp) from the original on 5 August 2011. Retrieved 30 October 2011.
29. "Stats Count Global Stats Top Five Operating Systems" (http://gs.statcounter.com/#os-ww-monthly-201010-201110). Archived (https://archive.today/20120526/http://gs.statcounter.com/%23mobile_browser-ww-monthly-201012-201111-bar#os-ww-monthly-201010-201110) from the original on 26 May 2012. Retrieved 30 October 2011.
30. "Global statistics at w3counter.com" (http://www.w3counter.com/globalstats.php). Archived (https://archive.today/20120628/http://www.w3counter.com/globalstats.php) from the original on 28 June 2012. Retrieved 23 January 2012.
31. "Troubleshooting MS-DOS Compatibility Mode on Hard Disks" (http://support.microsoft.com/kb/130179/EN-US). Support.microsoft.com. Archived (https://web.archive.org/web/20120810073641/http://support.microsoft.com/kb/130179/EN-US) from the original on 10 August 2012. Retrieved 7 August 2012.
32. "Using NDIS 2 PCMCIA Network Card Drivers in Windows 95" (http://support.microsoft.com/kb/134748/en). Support.microsoft.com. Archived (https://web.archive.org/web/20130217043405/http://support.microsoft.com/kb/134748/en) from the original on 17 February 2013. Retrieved 7 August 2012.
33. "INFO: Windows 95 Multimedia Wave Device Drivers Must be 16 bit" (http://support.microsoft.com/kb/163354/en). Support.microsoft.com. Archived (https://web.archive.org/web/20130217043412/http://support.microsoft.com/kb/163354/en) from the original on 17 February 2013. Retrieved 7 August 2012.
34. Arthur, Charles. "Windows 8 will run on ARM chips - but third-party apps will need rewrite" (https://www.theguardian.com/technology/2011/jan/05/microsoft-windows-8-arm-processors). *The Guardian*. Archived (https://web.archive.org/web/20161012022725/https://www.theguardian.com/technology/2011/jan/05/microsoft-windows-8-arm-processors) from the original on 12 October 2016.
35. "Operating System Share by Groups for Sites in All Locations January 2009" (https://web.archive.org/web/20090706135203/http://news.netcraft.com/SSL-Survey/CMatch/osdv_all). Archived from the original (http://news.netcraft.com/SSL-Survey/CMatch/osdv_all) on 6 July 2009. Retrieved 3 May 2010.
36. "Behind the IDC data: Windows still No. 1 in server operating systems" (http://blogs.zdnet.com/microsoft/?p=5408). *ZDNet*. 26 February 2010. Archived (https://web.archive.org/web/20100301032456/http://blogs.zdnet.com/microsoft/?p=5408) from the original on 1 March 2010.
37. Stallings, William (2008). *Computer Organization & Architecture*. New Delhi: Prentice-Hall of India Private Limited. p. 267. ISBN 978-81-203-2962-1.
38. "Operating Systems: Security" (https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/15_Security.html). *www.cs.uic.edu*. Retrieved 27 November 2020.
39. Poisson, Ken. "Chronology of Personal Computer Software" (http://www.islandnet.com/~kpolsson/compsoft/soft1998.htm) Archived (https://web.archive.org/web/20080514022217/http://www.islandnet.com/~kpolsson/compsoft/soft1998.htm) 14 May 2008 at the Wayback Machine. Retrieved on 2008-05-07. Last checked on 2009-03-30.
40. "Reading: Operating System" (https://courses.lumenlearning.com/sanjacinto-computerapps-v2/chapter/reading-operating-system/). *Lumen*. Retrieved 5 January 2019.

41. "The History of Unix" (https://archive.org/stream/byte-magazine-1983-08/1983_08_BYTE_08-08_The_C_Language#page/n189/mode/2up). *BYTE*. August 1983. p. 188. Retrieved 31 January 2015.

Further reading

- Auslander, Marc A.; Larkin, David C.; Scherr, Allan L. (1981). "The evolution of the MVS Operating System" (http://www.research.ibm.com/journal/rd/255/auslander.pdf) (PDF). IBM J. Research & Development.
- Deitel, Harvey M.; Deitel, Paul; Choffnes, David (25 December 2015). *Operating Systems* (https://archive.org/details/modernoperatings00tane). Pearson/Prentice Hall. ISBN 978-0-13-092641-8.
- Bic, Lubomur F.; Shaw, Alan C. (2003). *Operating Systems*. Pearson: Prentice Hall.
- Silberschatz, Avi; Galvin, Peter; Gagne, Greg (2008). *Operating Systems Concepts*. John Wiley & Sons. ISBN 978-0-470-12872-5.
- O'Brien, J. A., & Marakas, G. M.(2011). *Management Information Systems*. 10e. McGraw-Hill Irwin.
- Leva, Alberto; Maggio, Martina; Papadopoulos, Alessandro Vittorio; Terraneo, Federico (2013). *Control-based Operating System Design*. IET. ISBN 978-1-84919-609-3.
- Arpaci-Dusseau, Remzi; Arpaci-Dusseau, Andrea (2015). *Operating Systems: Three Easy Pieces* (http://pages.cs.wisc.edu/~remzi/OSTEP/).

External links

- Operating Systems (https://curlie.org/Computers/Software/Operating_Systems) at Curlie
- Multics History (http://www.cbi.umn.edu/iterations/haigh.html) and the history of operating systems