Course Title

Operating Systems & Systems Programming

Course No.

CSE - 3201 / CSE - 2205

Operating Systems & Systems Programming

Book References

Operating Systems Concepts

- Abraham Silberschatz and Peter Baer Galvin

Operating System: Design and Implementation

- Andrew S. Tanenbaum

Operating Systems

- William Stallings.

Systems Programming

Operating Systems & Systems Programming

Operating Systems

Introduction and Overview
Memory Management
Process and IPC
Deadlock

Systems Programming

Machine Structure,
Machine Language and Assembly Language
Assemblers
Loaders

Introduction : Definition

An operating system acts as an intermediary between the user of a computer and the computer hardware.

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.

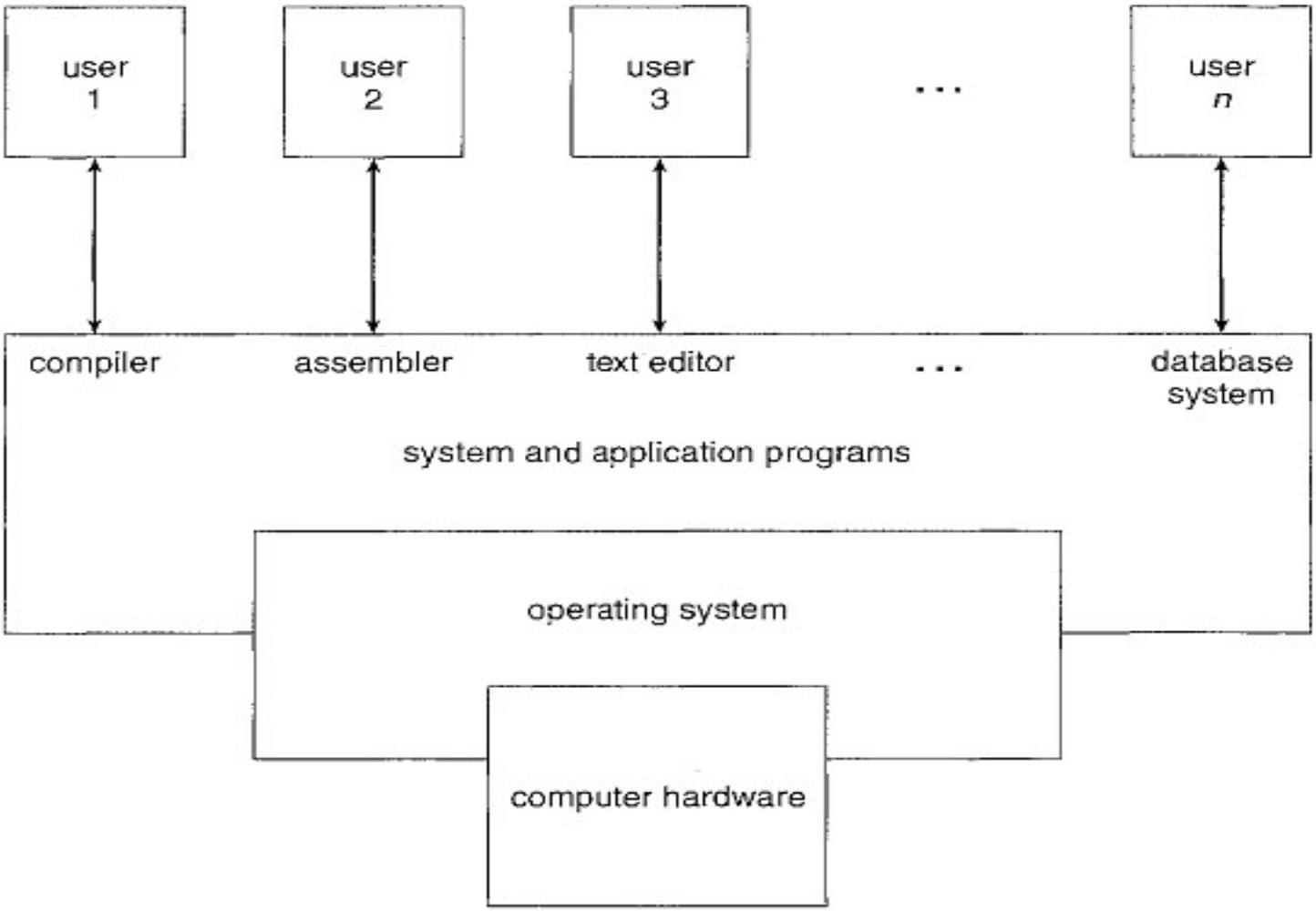
The goal is to maximize
- the ease of use
- resource utilization

Introduction : Definition

A computer system can be divided roughly into four components:

- The hardware
- The operating system
- System and application programs and
- The users

Introduction : Definition



Introduction : Definition

The hardware – the central processing unit (CPU), the memory, and input/output (I/O) devices – provides the basic computing resources for the system. The word processors/ spreadsheets/ compilers, and Web browsers- define the ways in which these resources are used to solve users' computing problems.

The operating system controls the hardware and coordinates its use among the various application programs for the various users.

Introduction : Definition

We can also view a computer system as consisting of hardware/ software/ and data. The operating system provides the means for proper use of these resources in the operation of the computer system. An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an environment within which other programs can do useful work.

Introduction : Definition

Computers have two modes of operation:

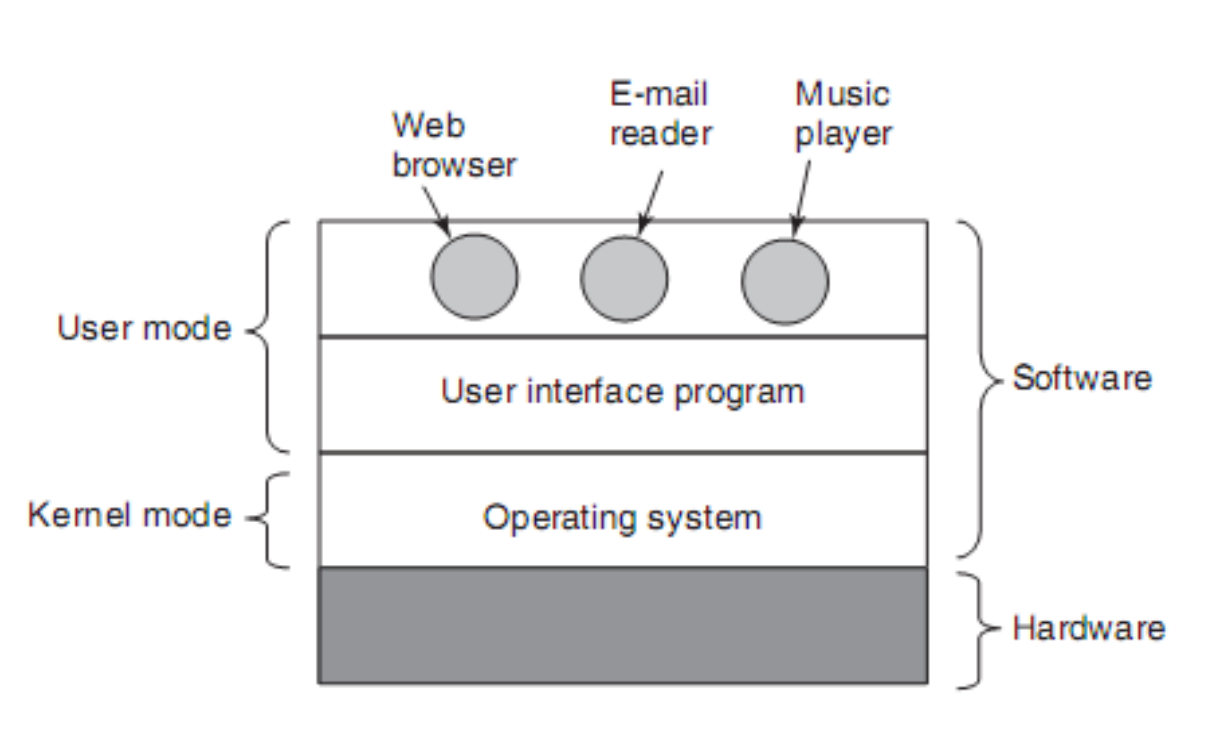
Kernel mode and

User mode.

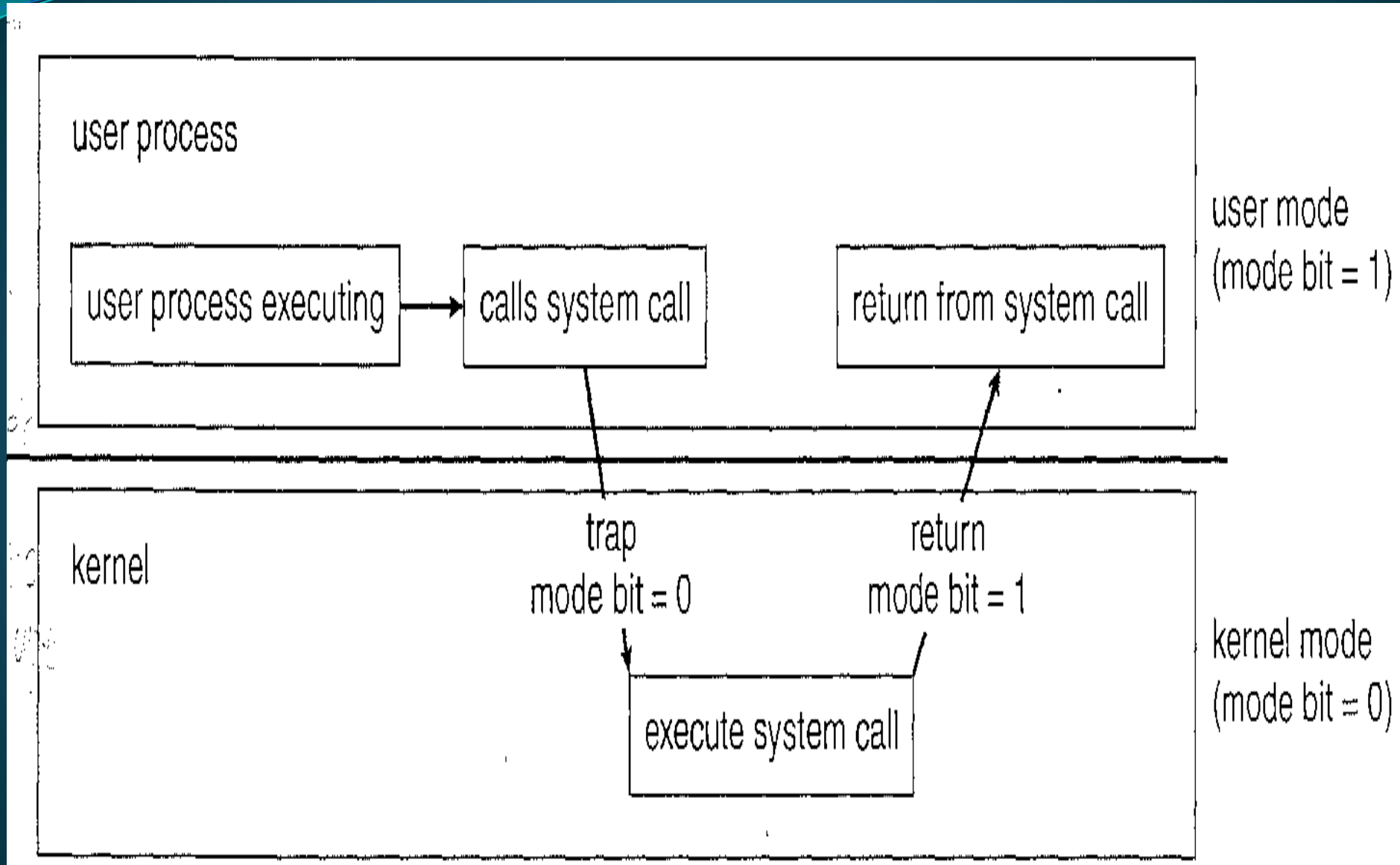
Operating system, the most fundamental piece of software, runs in kernel mode (also called supervisor mode). In this mode it has complete access to all the hardware and can execute any instruction the machine is capable of executing.

Introduction : Definition

The rest of the software runs in user mode, in which only a subset of the machine instructions is available. In particular, those instructions that affect control of the machine or do I/O (Input / Output) are forbidden to user-mode programs.



Introduction : Definition



History of OS

Early computers were

- physically enormous (large)
- run from consoles
- the programmer was also the operator of the computer system, would write a program and then would operate the program directly from the operator's console.
- First, the program would be loaded manually into memory from the front panel switches (one instruction at a time), from **paper tape**, or from **punched cards**.

History of OS

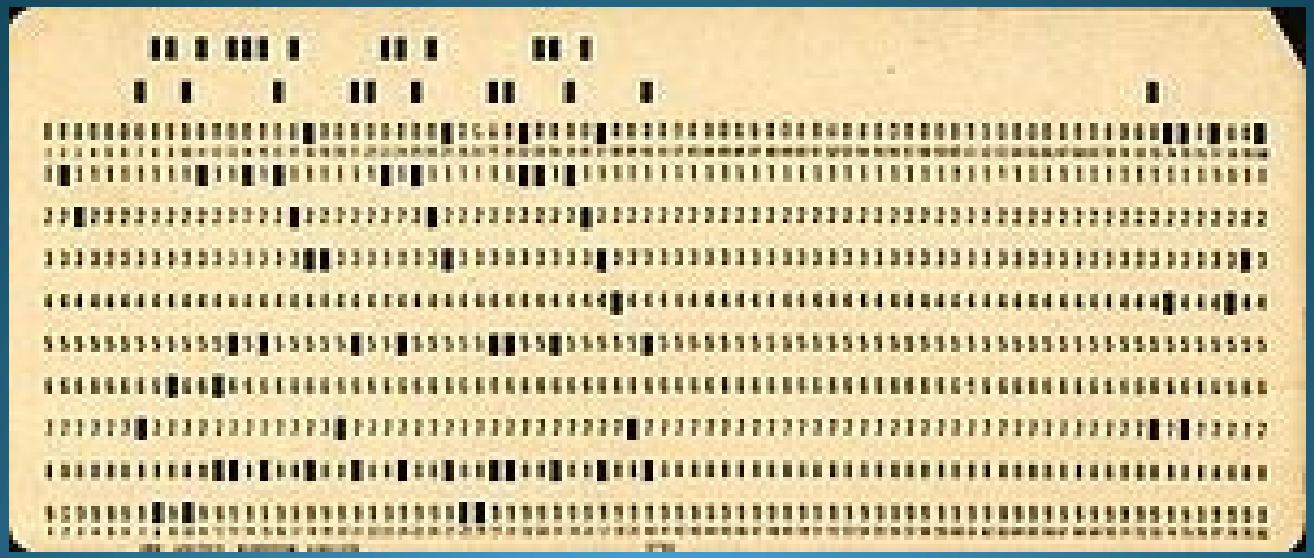
Paper Tape

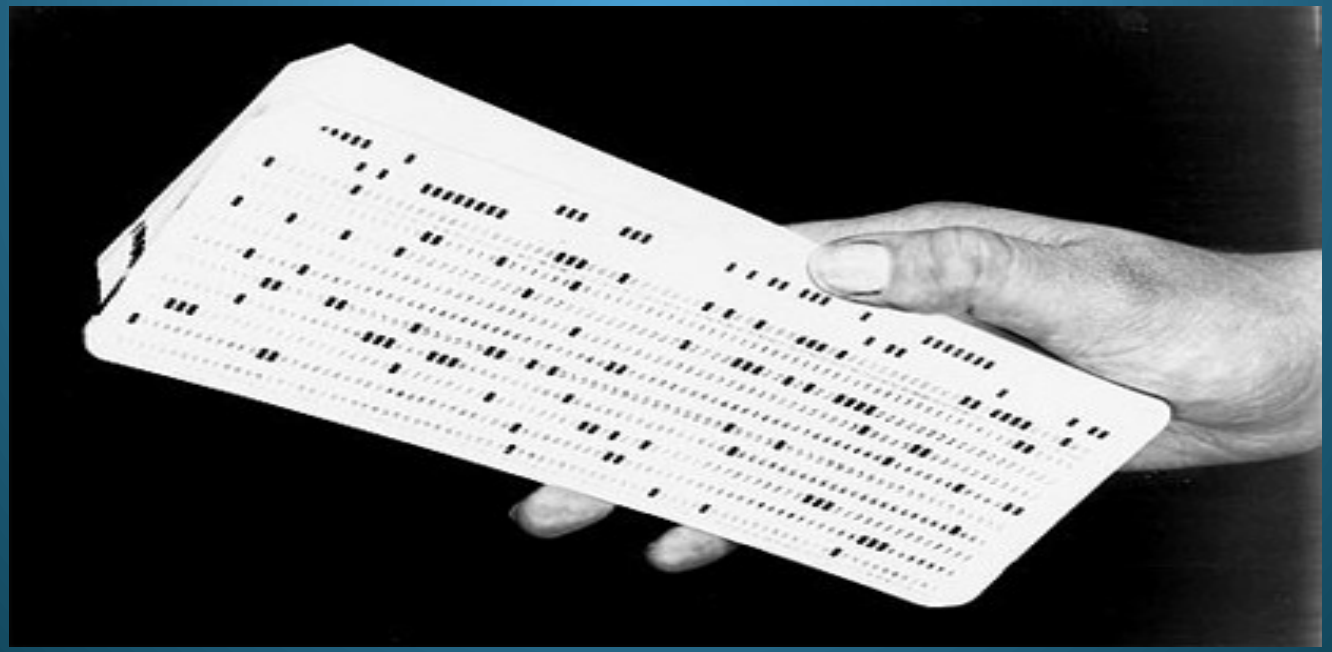
Five Hole Punched Paper Tape

Eight Hole Punched Paper Tape

History of OS

Punch Cards





History of OS

- Then the appropriate buttons would be pushed to set the starting address and to start the execution of the program. As the program ran, the programmer/operator could monitor its execution by the display lights on the console.

- If errors were discovered, the programmer could halt the program, examine the contents of memory and registers, and debug the program directly from the console.

- Output was printed or was punched onto **paper tape** or **cards** for later printing.

History of OS

- As time went on, additional software and hardware were developed. Card readers, line printers, and magnetic tape became commonplace.
- **Assemblers,** loaders, and linkers were designed to ease the programming task.
- Libraries of common functions were created. Common functions could then be copied into a new program without having to be written again, providing **software reusability**.

History of OS

The routines that performed I/O were especially important. Each new I/O device had its own characteristics, requiring careful programming. A special subroutine called a device driver -was written for each I/0 device.

History of OS

Later, compilers for FORTRAN, COBOL, and other languages appeared, making the programming task much easier but the operation of the computer more complex. To prepare a FORTRAN program for execution, for example,

- The programmer would first need to load the **FORTRAN compiler** into the computer. The compiler was normally kept on **magnetic tape**, so the proper tape would need to be mounted on a tape drive.

- The program would be read through the **card reader** and written onto another **tape**.

History of OS

- The FORTRAN **compiler** produced assembly-language output, which then had to be assembled.
- This procedure required mounting another **tape** with **the assembler**.
- The output of the assembler would need to be **linked** to supporting library routines.
- Finally, the binary object form of the program would be ready to execute. It could be loaded into memory and debugged from the console, as before.

History of OS

A significant amount of set-up time could be involved in the running of job. Each job consisted of many separate steps:

- Loading the FORTRAN compiler tape
- Running the compiler
- Unloading the compiler tape
- Loading the assembler tape
- Running the assembler
- Unloading the assembler tape
- Loading the object program
- Running the object program

History of OS

Time consuming (for setting up)

If an error occurred during any step, the programmer/operator might have to start over at the beginning. Each job step might involve the loading and unloading of magnetic tapes, paper tapes, and punch cards.

Expensive (for setting up)

The job set-up time was a real problem. While tapes were being mounted or the programmer was operating the console, the CPU sat idle. That time, computer time was extremely valuable, and owners wanted their computers to be used as much as possible. They needed high utilization to get as much as they could from their investments.

History of OS

The solution was two fold.

First, a professional computer operator was hired. Since the operator had more experience with mounting tapes than a programmer, set-up time was reduced.

The programmer provided whatever cards or tapes were needed, as well as a short description of how the job was to be run. Since the operator would not understand the program, therefore, in the case of program error, a dump of memory and registers was taken, and the programmer had to debug from the dump. Dumping the memory and registers allowed the operator to continue immediately with the next job but left the programmer with the more difficult debugging problem.

History of OS

Second, jobs with similar needs were batched together and run through the computer as a group to reduce set-up time.

For instance, suppose the operator received one FORTRAN job, one COBOL job, and another FORTRAN job.

If he/she ran them in that order, she would have to set up for FORTRAN (load the compiler tapes and so on), then set up for COBOL, and then set up for FORTRAN again.

If he/she ran the two FORTRAN programs as a batch, however, he/she could set up only once for FORTRAN, saving operator time.

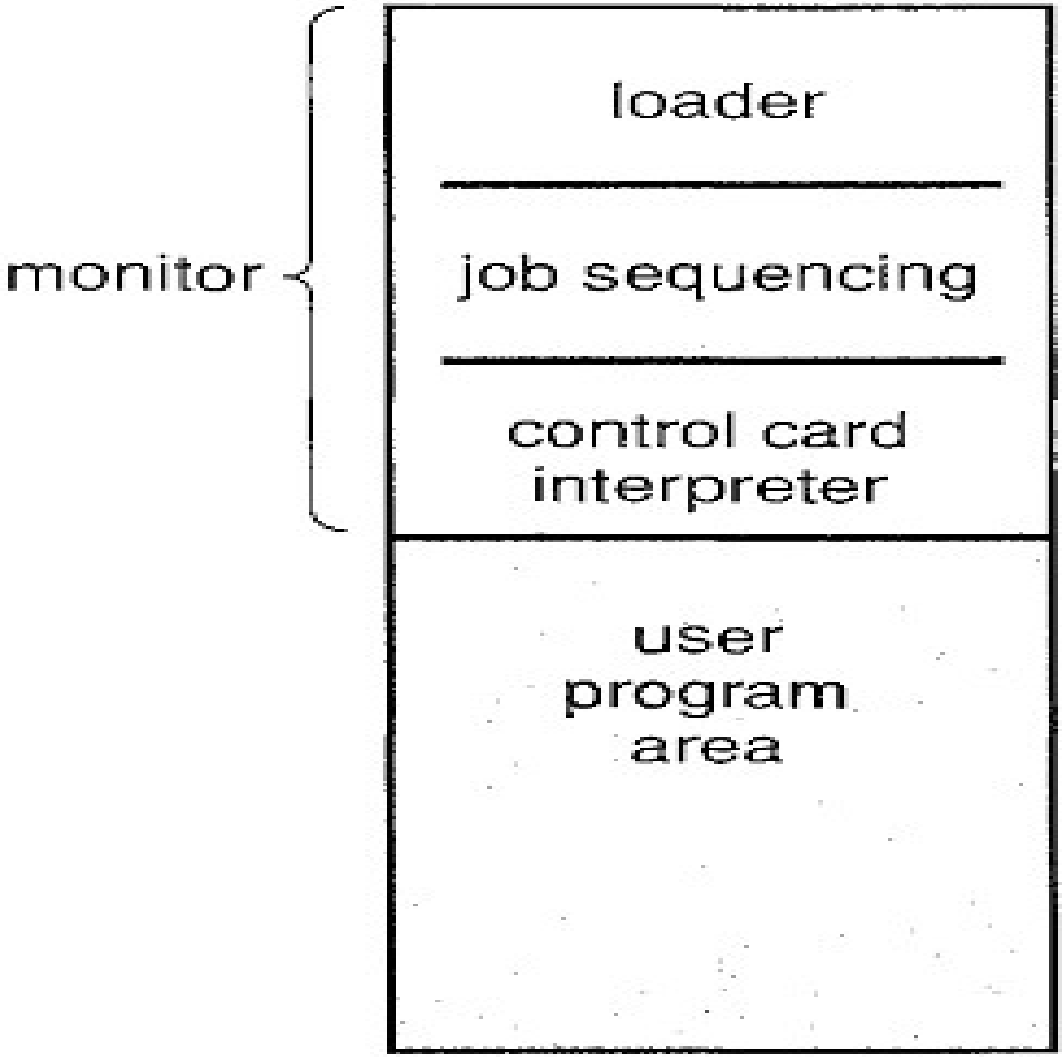
History of OS

But there were still problems. For example, when a job stopped, the operator would have to notice that it had stopped (by observing the console), determine why it stopped (normal or abnormal termination), dump memory and register (if necessary), load the appropriate device with the next job, and restart the computer. During this transition from one job to the next, the CPU sat idle.

History of OS

To overcome this idle time, people developed automatic job sequencing ; with this technique, the first rudimentary operating systems were created. A small program, called a resident monitor, was created to transfer control automatically from one job to the next (Figure in the nest slide). The resident monitor is always in memory (or resident).

History of OS



History of OS

When the computer was turned on, the resident monitor was invoked, and

- It would transfer control to a program
- When the program terminated, it would return control to the resident monitor, which would then go on to the next program.

Thus, the resident monitor would automatically sequence from one program to another and from one job to another.

History of OS

But how would the resident monitor know which program to execute?

Previously, the operator had been given a short description of what programs were to be run on what data. Control Cards were introduced to provide this information directly to the monitor.

History of OS

For example, a normal user program might require one of three programs to run:
the FORTRAN compiler (FTN),
the assembler (ASM), or
the user's program (RUN).

We could use a separate control card for each of these:
$FTN- Execute the FORTRAN compiler.
$ASM- Execute the assembler.
$RUN- Execute the user program.
These cards tell the resident monitor which program to run.

History of OS

We can use two additional control cards to define the boundaries of each job:

$JOB-First card of a job

$END-Final card of a job

History of OS

A resident monitor thus has several identifiable parts:

- The control-card interpreter is responsible for reading and carrying out the instructions on the cards at the point of execution.
- The loader is invoked by the control-card interpreter to load system programs and application programs into memory at intervals.
- The device drivers are used by both the control-card interpreter and the loader for the system's I/0 devices.

History of OS

The switch to batch systems with automatic job sequencing was made to improve performance.

The problem, quite simply, is that humans are considerably slower than the computer. Consequently, it is desirable to replace human operation with operating-system software. Automatic job sequencing eliminates the need for human set-up time and job sequencing.

History of OS

Even with this arrangement, however, the CPU is often idle. The problem is the speed of the mechanical I/0 devices, which are intrinsically slower than electronic devices. Even a slow CPU works in the microsecond range, with thousands of instructions executed per second.

A fast card reader, in contrast, might read 1,200 cards per minute (or 20 cards per second).

History of OS

Thus, the difference in speed between the CPU and its I/0 devices may be three orders of magnitude or more.

Over time, of course, improvements in technology resulted in faster I/0 devices. Unfortunately, CPU speeds increased even faster, so that the problem was not resolved.

History of OS

One common solution to the I/0 problem was to replace slow card readers (input devices) and line printers (output devices) with magnetic-tape units.

Most computer systems in the late 1950s and early 1960s were batch systems reading from card readers and writing to line printers or card punches.

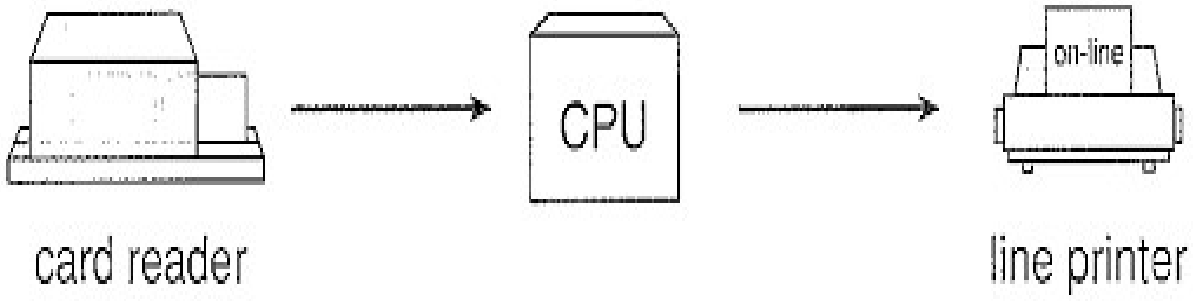
The CPU did not read directly from cards, however; instead, the cards were first copied onto a magnetic tape via a separate device. When the tape was sufficiently full, it was taken down and carried over to the computer.

History of OS

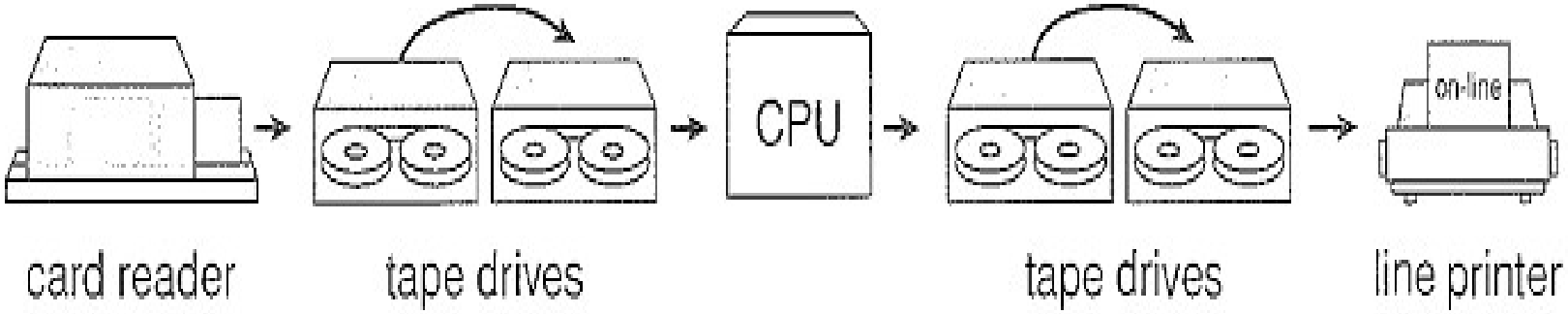
When a card was needed for input to a program, the equivalent record was read from the tape. Similarly, output was written to the tape, and the contents of the tape were printed later.

The card readers and line printers were operated off-line, rather than by the main computer (Figure in the next slide).

History of OS



(a) Online System



(b) Off-line System

History of OS

An obvious advantage of off-line operation was that the main computer was no longer constrained by the speed of the card readers and line printers but was limited only by the speed of the much faster magnetic tape units.

The real gain in off-line operation comes from the possibility of using multiple reader-to-tape and tape-to-printer systems for one CPU. If the CPU can process input twice as fast as the reader can read cards, then two readers working simultaneously can produce enough tape to keep the CPU busy.

History of OS

There is a disadvantage, too, however- a longer delay in getting a particular job run. The job must first be read onto tape. Then it must wait until enough additional jobs are read onto the tape to fill it. The tape must then be rewound, unloaded, hand-carried to the CPU, and mounted on a free tape drive. This process is not unreasonable for batch systems, of course. Many similar jobs can be batched onto a tape before it is taken to the computer.

History of OS

The problem with tape systems was that the card reader could not write onto one end of the tape while the CPU read from the other. The entire tape had to be written before it was rewound and read, because tapes are by nature sequential-access devices.

Disk systems eliminated this problem by being random-access devices. Because the head is moved from one area of the disk to another, it can switch rapidly from the area on the disk being used by the card reader to store new cards to the position needed by the CPU to read the next card.

History of OS

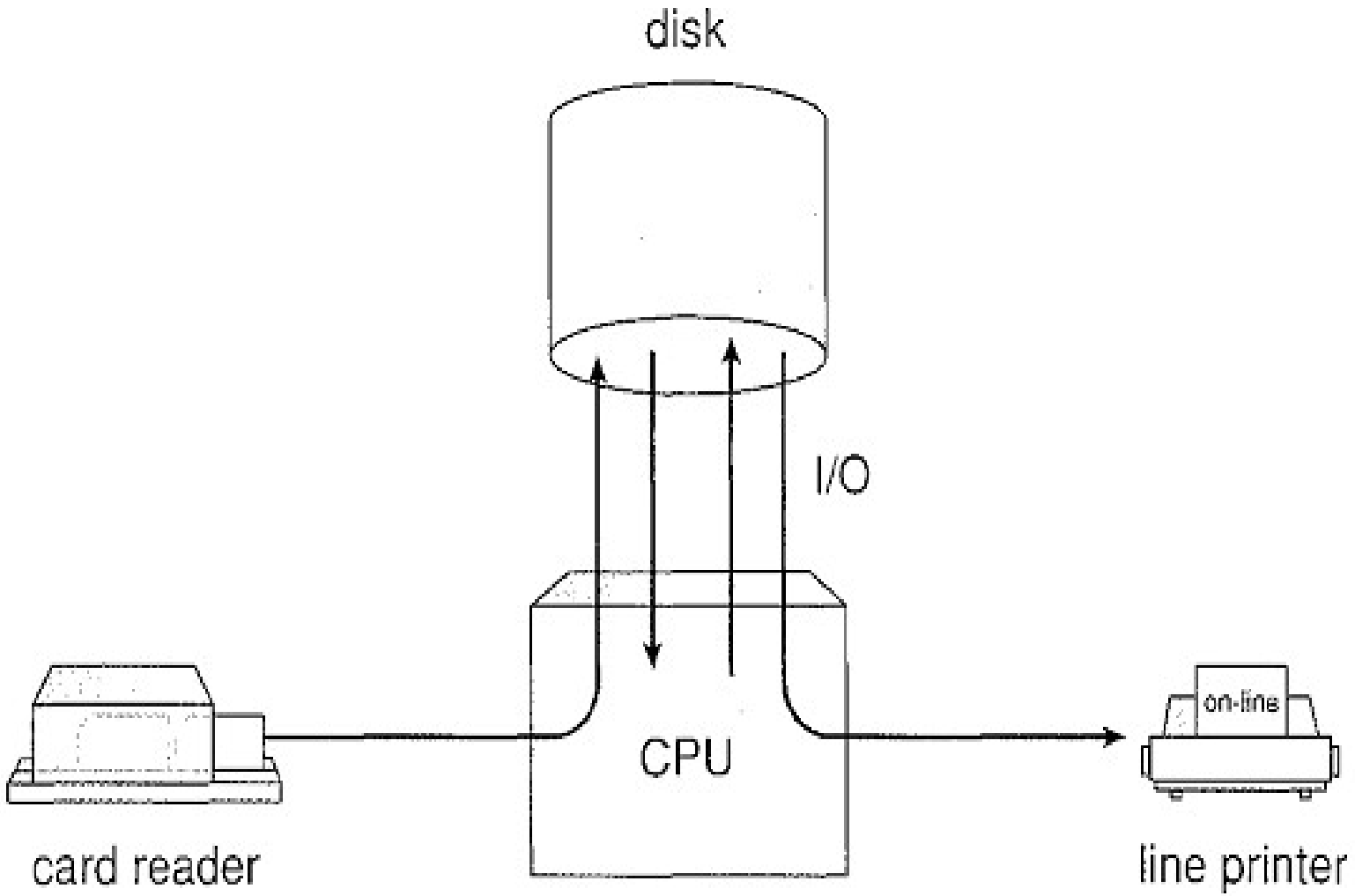
In a disk system, cards are read directly from the card reader onto the disk. The location of card images is recorded in a table kept by the operating system.

When a job is executed, the operating system satisfies its requests for card-reader input by reading from the disk.

Similarly, when the job requests the printer to output a line, that line is copied into a system buffer and is written to the disk. When the job is completed, the output is actually printed.

This form of processing is called spooling (Figure in the next slide); the name is an acronym for simultaneous peripheral operation on-line.

History of OS



History of OS

Multi-programming Operating System
Time Sharing Operating System
    CTSS (Compatible Time Sharing System) at MIT
Real time Operating System
Network Operating System
Distributed Operating System

History of OS

After the success of the CTSS system, M.I.T., Bell Labs, and General Electric (at that time a major computer manufacturer) decided to embark on the development of a ‘‘computer utility,’’ that is, a machine that would support some hundreds of simultaneous timesharing users.

Their model was the electricity system—when you need electric power, you just stick a plug in the wall, and within reason, as much power as you need will be there. The designers of this system, known as MULTICS (MULTiplexed Information and Computing Service), envisioned one huge machine providing computing power for everyone in the Boston area.

History of OS

MULTICS was a mixed success. It was designed to support hundreds of users on a machine (only slightly more powerful than an Intel 386-based PC, although it had much more I/O capacity). This is not quite as crazy as it sounds.

There were many reasons that MULTICS did not take over the world, not the least of which is that it was written in the PL/I programming language (Programming Language One), and the PL/I compiler was years late and barely worked at all when it finally arrived. In addition, MULTICS was enormously ambitious for its time, much like Charles Babbage's analytical engine in the nineteenth century.

History of OS

Despite its lack of commercial success, MULTICS had a huge influence on subsequent operating systems (especially UNIX and its derivatives, FreeBSD, Linux, iOS, and Android).

One of the computer scientists at Bell Labs who had worked on the MULTICS project, Ken Thompson, subsequently found a small PDP-7 (Programmed Data Processor or Programmable Data Processor) minicomputer that no one was using and set out to write a stripped-down, one-user version of MULTICS. This work later developed into the UNIX operating system, which became popular in the academic world, with government agencies, and with many companies.

History of OS

Because the source code was widely available, various organizations developed their own (incompatible) versions, which led to chaos. Two major versions developed, System V, from AT&T, and BSD (Berkeley Software Distribution) from the University of California at Berkeley. These had minor variants as well.

To make it possible to write programs that could run on any UNIX system, IEEE developed a standard for UNIX, called POSIX (Portable Operating System Interface), that most versions of UNIX now support. POSIX defines a minimal system-call interface that conformant UNIX systems must support. In fact, some other operating systems now also support the POSIX interface.

History of OS

In 1987, Andrew S Tanenbaum released a small clone of UNIX, called MINIX, for educational purposes. Functionally, MINIX is very similar to UNIX, including POSIX support. Since that time, the original version has evolved into MINIX 3, which is highly modular and focused on very high reliability. It has the ability to detect and replace faulty or even crashed modules (such as I/O device drivers) on the fly without a reboot and without disturbing running programs. Its focus is on providing very high dependability and availability.

A book describing its internal operation and listing the source code in an appendix is also available (Tanenbaum and Woodhull, 2006). The MINIX 3 system is available for free (including all the source code) over the Internet at www.minix3.org.

History of OS

The desire for a free production version of MINIX led a Finnish student, Linus Torvalds, to write Linux (released on September 17, 1991)

This system was directly inspired by and developed on MINIX and originally supported various MINIX features (e.g., the MINIX file system).

It has since been extended in many ways by many people but still retains some underlying structure common to MINIX and to UNIX.

History of OS

In 1974, Intel came out with the 8080, the first general-purpose 8-bit CPU, it wanted an operating system for the 8080, in part to be able to test it.

Intel asked one of its consultants, Gary Kildall, to write one. Kildall and a friend first built a controller for the newly released Shugart Associates 8-inch floppy disk and hooked the floppy disk up to the 8080, thus producing the first microcomputer with a disk.

Kildall then wrote a disk-based operating system called CP/M (Control Program for Microcomputers) for it. Since Intel did not think that disk-based microcomputers had much of a future, when Kildall asked for the rights to CP/M, Intel granted his request. Kildall then formed a company, Digital Research, to further develop and sell CP/M.

History of OS

In 1977, Digital Research rewrote CP/M to make it suitable for running on the many microcomputers using the 8080, Zilog Z80, and other CPU chips. Many application programs were written to run on CP/M, allowing it to completely dominate the world of micro-computing for about 5 years.

History of OS

In the early 1980s, IBM designed the IBM PC and looked around for software to run on it. People from IBM contacted Bill Gates to license his BASIC interpreter. They also asked him if he knew of an operating system to run on the PC. Gates suggested that IBM contact Digital Research, then the world's dominant operating systems company.

Making what was surely the worst business decision in recorded history, Kildall refused to meet with IBM, sending a subordinate instead. To make matters even worse, his lawyer even refused to sign IBM's nondisclosure agreement covering the not-yet-announced PC.

Consequently, IBM went back to Gates asking if he could provide them with an operating system.

History of OS

When IBM came back, Gates realized that a local computer manufacturer, Seattle Computer Products, had a suitable operating system, DOS (Disk Operating System).

He approached them and asked to buy it (allegedly for $75,000), which they readily accepted.

Gates then offered IBM a DOS/BASIC package, which IBM accepted. IBM wanted certain modifications, so Gates hired the person who wrote DOS, Tim Paterson, as an employee of Gates' fledgling company, Microsoft, to make them.

History of OS

The revised system was renamed MS-DOS (MicroSoft Disk Operating System) and quickly came to dominate the IBM PC market.

A key factor here was Gates' (in retrospect, extremely wise) decision to sell MS-DOS to computer companies for bundling with their hardware, compared to Kildall's attempt to sell CP/M to end users one at a time (at least initially).

History of OS

By the time the successor to the IBM PC, the IBM PC/AT, came out in 1983 with the Intel 80286 CPU, MS-DOS was firmly entrenched and CP/M was on its last legs.

MS-DOS was later widely used on the 80386 and 80486. Although the initial version of MS-DOS was fairly primitive, subsequent versions included more advanced features, including many taken from UNIX. (Microsoft was well aware of UNIX, even selling a microcomputer version of it called XENIX during the company's early years.)

History of OS

CP/M, MS-DOS, and other operating systems for early microcomputers were all based on users typing in commands from the keyboard. That eventually changed due to research done by Doug Engelbart (Douglas Carl Engelbart) at Stanford Research Institute (SRI) in the 1960s. Engelbart invented the Graphical User Interface, complete with windows, icons, menus, and mouse. These ideas were adopted by researchers at Xerox PARC and incorporated into machines they built.

History of OS

One day, Steve Jobs, who co-invented the Apple computer in his garage, visited PARC, saw a GUI, and instantly realized its potential value, something Xerox management famously did not.

Jobs then embarked on building an Apple with a GUI. This project led to the Lisa, which was too expensive and failed commercially.

Jobs' second attempt, the Apple Macintosh, was a huge success, not only because it was much cheaper than the Lisa, but also because it was user friendly, meaning that it was intended for users who not only knew nothing about computers but furthermore had absolutely no intention whatsoever of learning.

History of OS

In the creative world of graphic design, professional digital photography, and professional digital video production, Macintoshes are very widely used and their users are very enthusiastic about them.

In 1999, Apple adopted a kernel derived from Carnegie Mellon University's Mach microkernel which was originally developed to replace the kernel of BSD UNIX. Thus, Mac OS X is a UNIX-based operating system, albeit with a very distinctive interface.

History of OS

When Microsoft decided to build a successor to MS-DOS, it was strongly influenced by the success of the Macintosh. It produced a GUI-based system called Windows, which originally ran on top of MS-DOS (i.e., it was more like a shell than a true operating system). For about 10 years, from 1985 to 1995, Windows was just a graphical environment on top of MS-DOS.

However, starting in 1995, a freestanding version, Windows 95, was released that incorporated many operating system features into it, using the underlying MS-DOS system only for booting and running old MS-DOS programs.

History of OS

In 1998, a slightly modified version of this system, called Windows 98 was released. Nevertheless, both Windows 95 and Windows 98 still contained a large amount of 16-bit Intel assembly language.

Another Microsoft operating system, Windows NT (where the NT stands for New Technology), which was compatible with Windows 95 at a certain level, but a complete rewrite from scratch internally. It was a full 32-bit system. The lead designer for Windows NT was David Cutler, who was also one of the designers of the VAX VMS operating system, so some ideas from VMS are present in NT.

History of OS

In fact, so many ideas from VMS were present in it that the owner of VMS, DEC, sued Microsoft. The case was settled out of court for an amount of money.

Microsoft expected that the first version of NT would kill off MS-DOS and all other versions of Windows since it was a vastly superior system, but it fizzled. Only with Windows NT 4.0 did it finally catch on in a big way, especially on corporate networks.

Version 5 of Windows NT was renamed Windows 2000 in early 1999. It was intended to be the successor to both Windows 98
and Windows NT 4.0. That did not quite work out either, so Microsoft came out with yet another version of Windows 98 called Windows Me (Millennium Edition).

History of OS

In 2001, a slightly upgraded version of Windows 2000, called Windows XP was released. That version had a much longer run (6 years), basically replacing all previous versions of Windows.

After Windows 2000, Microsoft broke up the Windows family into a client and a server line. The client line was based on XP and its successors, while the server line included Windows Server 2003 and Windows 2008. A third line, for the embedded world, appeared a little later. All of these versions of Windows forked off their variations in the form
of service packs. It was enough to drive some administrators (and writers of operating systems textbooks) balmy.

History of OS

Then in January 2007, Microsoft finally released the successor to Windows XP, called Vista. It came with a new graphical interface, improved security, and many new or upgraded user programs.

Microsoft hoped it would replace Windows XP completely, but it never did. Instead, it received much criticism and a bad press,
mostly due to the high system requirements, restrictive licensing terms, and support for Digital Rights Management, techniques that made it harder for users to copy protected material.

Early System

With the arrival of Windows 7, a new and much less resource hungry version of the operating system, many people decided to skip Vista altogether. Windows 7 did not introduce too many new features, but it was relatively small and quite stable.

In less than three weeks, Windows 7 had obtained more market share than Vista in seven months. In 2012, Microsoft launched its successor, Windows 8, an operating system with a completely new look and feel, geared for touch screens.

The company hopes that the new design will become the dominant operating system on a much wider variety of devices: desktops, laptops, notebooks, tablets, phones, and home theater PCs. So far, however, the market penetration is slow compared to Windows 7.

History of OS

Symbian OS
Blackberry OS
Apple’s iOS
Android

History of OS

We can make calls not just with our portable phones and wrist watches, but soon with eyeglasses and other wearable items.

Moreover, the phone part is no longer that interesting. We receive email, surf the Web, text our friends, play games, navigate around heavy traffic

History of OS

The first real mobile phone appeared in 1946

Weighed some 40 kilos.

You could take it wherever you went as long as you had a car in which to carry it.

The first true handheld phone appeared in the 1970s and, at roughly one kilogram

History of OS

While the idea of combining telephony and computing in a phone-like device has been around since the 1970s also, the first real smartphone did not appear until the mid-1990s when Nokia released the N9000, which literally combined two, mostly separate devices: a phone and a PDA (Personal Digital Assistant).

In 1997, Ericsson coined the term smartphone for its GS88 “Penelope.”

History of OS

Most smartphones in the first decade after their inception were running Symbian OS

It was the operating system of choice for popular brands like Samsung, Sony Ericsson, Motorola, and especially Nokia.

Other operating systems like RIM's Blackberry OS (introduced for smartphones in 2002), Apple's iOS (released for the first iPhone in 2007) started eating into Symbian's market share

Symbian's market share plummeted

In 2011, Nokia ditched Symbian and announced it would focus on Windows Phone as its primary platform

History of OS

It did not take very long for Android, a Linux-based operating system released by Google in 2008, to overtake all its rivals.

For phone manufacturers, Android had the advantage that it was open source and available under a permissive license. As a result, they could tinker with it and adapt it to their own hardware with ease.