OPERATING SYSTEMS

OPERATING SYSTEMS CONCEPTS

1.1 General Definition

2

- An Operating System (OS) is a program which acts as an *interface* between computer system users and the computer hardware.
- It provides a user-friendly environment in which a user may easily develop and execute programs.
- Otherwise, hardware knowledge would be mandatory for computer programming.
- So, it can be said that an OS hides the complexity of hardware from uninterested users.

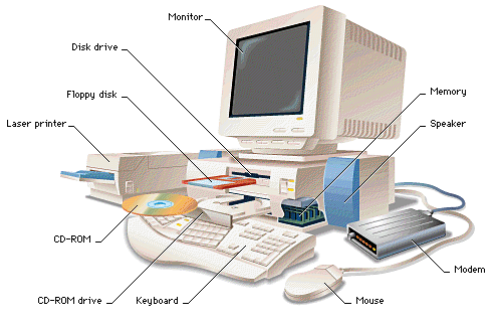
1.1 General Definition

3

- In general, a computer system has some resources which may be utilized to solve a problem. They are
  - Memory
  - Processor(s)
  - I/O
  - File System
  - etc.

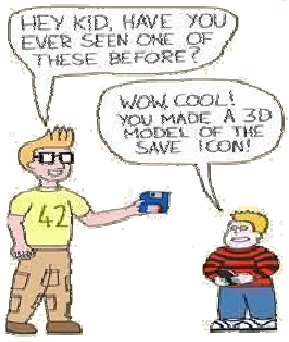
1.1 General Definition

4



Cartoon

5



1.1 General Definition

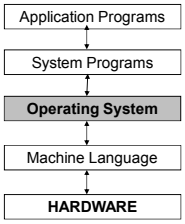
6

- The OS manages these resources and allocates them to specific programs and users.
- With the management of the OS, a programmer is rid of difficult hardware considerations.
- An OS provides services for
  - Processor Management
  - Memory Management
  - File Management
  - Device Management
  - Concurrency Control

1.1 General Definition

7

- Another aspect for the usage of OS is that; it is used as a *predefined library* for hardware-software interaction.
- This is why, system programs apply to the installed OS since they cannot reach hardware directly.



1.1 General Definition

8

- Since we have an already written library, namely the OS, to add two numbers we simply write the following line to our program:

```
c = a + b ;
```

1.1 General Definition

9

- in a system where there is no OS installed, we should consider some hardware work as:
  (Assuming an MC 6800 computer hardware)

```
LDAA $80 → Loading the number at memory location 80
LDAB $81 → Loading the number at memory location 81
ADDB     → Adding these two numbers
STAA $55 → Storing the sum to memory location 55
```

- As seen, we considered memory locations and used our hardware knowledge of the system.

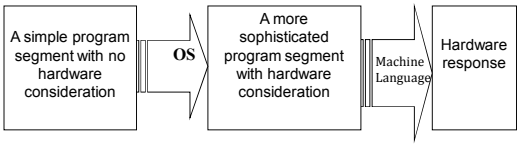
1.1 General Definition

10

- In an OS installed machine, since we have an intermediate layer, our programs obtain *some advantage of mobility* by not dealing with hardware.
- For example, the above program segment would not work for an 8086 machine, where as the
  "c = a + b ;"
  syntax will be suitable for both.

1.1 General Definition

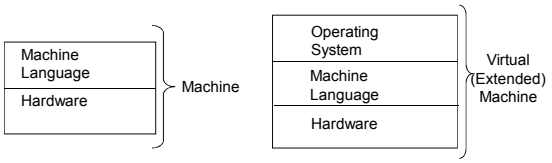
11



1.1 General Definition

12

- With the advantage of easier programming provided by the OS, the hardware, its machine language and the OS constitutes a new combination called as a **virtual (extended) machine.**



1.1 General Definition

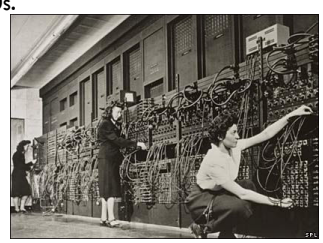
13

- In a more simplistic approach, in fact, OS itself is a program.
- But it has a priority which application programs don't have.
- OS uses the **kernel mode** of the microprocessor, whereas other programs use the **user mode**.
- The difference between two is that; all hardware instructions are valid in kernel mode, where some of them cannot be used in the user mode.

1.2 History of Operating Systems

14

- It all started with computer hardware in about 1940s.



ENIAC 1943

1.2 History of Operating Systems

15

- ENIAC (Electronic Numerical Integrator and Computer), at the U.S. Army's Aberdeen Proving Ground in Maryland.
  - built in the 1940s,
  - weighed 30 tons,
  - was eight feet high, three feet deep, and 100 feet long
  - contained over 18,000 vacuum tubes that were cooled by 80 air blowers.

1.2 History of Operating Systems

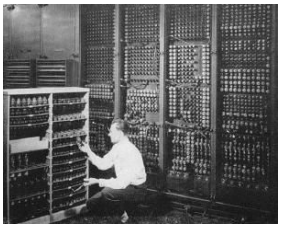
16

- Computers were using vacuum tube technology.

ENIAC's vacuum tubes

1.2 History of Operating Systems

17

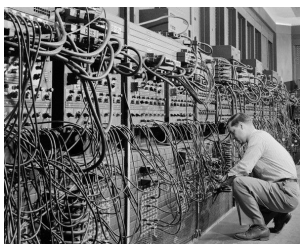


ENIAC's backside

1.2 History of Operating Systems

18

Programs were loaded into memory manually using switches, punched cards, or paper tapes.

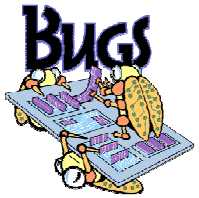


ENIAC : coding by cable connections

1.2 History of Operating Systems

19

Although some people assert origin of the term "debugging" be based on Thomas Edison or aeronautics, in computer science, the most of people believe its coming from an interesting story on first computer studies in the 1940s.



1.2 History of Operating Systems

20

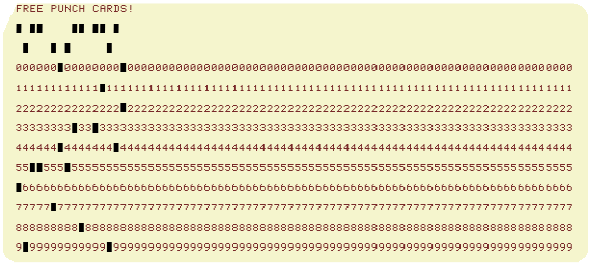
While a group researchers were working on first computer in the history, they discovered a kind of bug stuck in circuits. Because it prevented to run the computer, they cleaned the system from bugs.

Since that day, the term “debugging” is used as cleaning the system, but especially in software.

1.2 History of Operating Systems

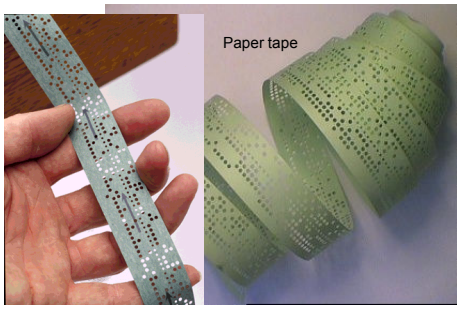
21

punch card



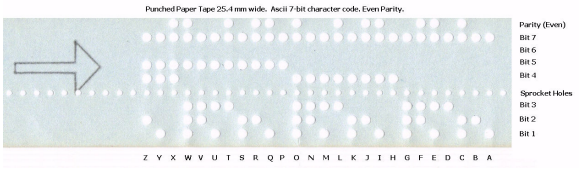
1.2 History of Operating Systems

22



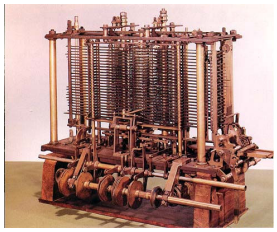
1.2 History of Operating Systems

23



1.2 History of Operating Systems

24

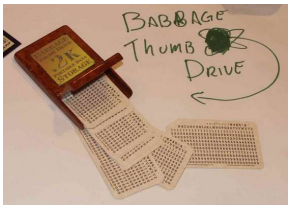


Babbage’s analytical engine
(designed in 1840’s by Charles Babbage, but cold not be constructed by him.
An earlier and simpler version is constructed in 2002, in London)

1.2 History of Operating Systems

25

- Ada Lovalence (at time of Charles Babbage) wrote code for analytical engine to compute Bernulli Numbers



1.2 History of Operating Systems

26

- As time went on, card readers, printers, and magnetic tape units were developed as additional hardware elements.
- Assemblers, loaders and simple utility libraries were developed as software tools.
- Later, off-line spooling and channel program methods were developed sequentially.

1.2 History of Operating Systems

27

- Finally, the idea of **multiprogramming** came.
- Multiprogramming means sharing of resources between more than one processes.
- By multiprogramming the CPU time is not wasted, because, while one process moves on some I/O work, the OS picks another process to execute till the current one passes to I/O operation.

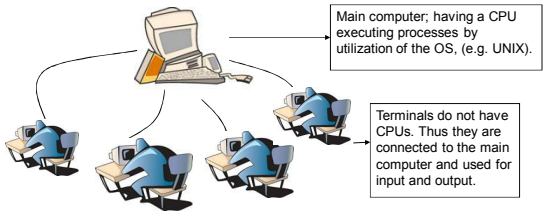
1.2 History of Operating Systems

28

- With the development of interactive computation in 1970s, **time-sharing systems** emerged.
- In these systems, multiple users have *terminals* (not computers) connected to a *main computer* and execute her task in the main computer.

1.2 History of Operating Systems

29



1.2 History of Operating Systems

30

- Another computer system is the **multiprocessor system** having multiple processors sharing memory and peripheral devices.
- With this configuration, they have greater computing power and higher reliability.

1.2 History of Operating Systems

31

- Multiprocessor systems are classified into two as tightly-coupled and loosely-coupled (distributed).
- In the tightly-coupled one, each processor is assigned a specific duty but processors work in close association, possibly sharing the same memory.
- In the loosely coupled one, each processor has its own memory and copy of the OS.

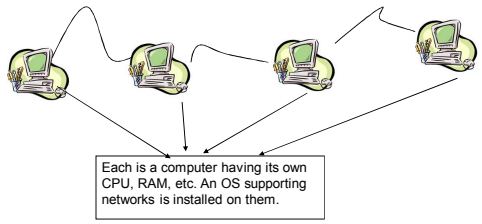
1.2 History of Operating Systems

32

- Use of the networks required OSs appropriate for them.
- In **network systems**, each process runs in its own machine but the OS have access to other machines.
- By this way, file sharing, messaging, etc. became possible.
- In networks, users are aware of the fact that s/he is working in a network and when information is exchanged. The user explicitly handles the transfer of information.

1.2 History of Operating Systems

33



1.2 History of Operating Systems

34

- **Distributed systems** are similar to networks. However in such systems, there is no need to exchange information explicitly, it is handled by the OS itself whenever necessary.
- With continuing innovations, new architectures and compatible OSs are developed. But their details are not in the scope of this text since the objective here is to give only a general view about developments in OS concept.

1.3 Operating Systems Structure

35

According to their structures, operating systems are divided into five type:

- Monolithic Systems
- Layered Systems
- Virtual Machines
- Exokernels
- Client-Server

1.3.1 Monolithic Systems

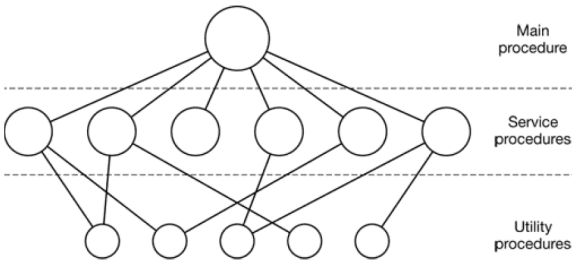
36

In fact, there is no structure. The operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.

When this technique is used, each procedure in the system has a well-defined interface in terms of parameters and results, and each one is free to call any other one, if the latter provides some useful computation that the former needs.

1.3.1 Monolithic Systems

37



1.3.2 Layered Systems

38

Another approach is to organize the operating system as a hierarchy of layers, each one constructed upon the one below it.

The first system constructed in this way was the THE system built at the Technische Hogeschool Eindhoven in the Netherlands by E. W. Dijkstra (1968) and his students.

1.3.2 Layered Systems

39

5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

THE system (Technische Hogeschool Eindhoven) by E. W. Dijkstra (1968)

1.3.3 Virtual Machines

40

Many users wanted to have timesharing, so various groups, both inside and outside IBM decided to write timesharing systems for it.

Produced system (VM/370) was based on an astute observation: a timesharing system provides (1) multiprogramming and (2) an extended machine with a more convenient interface than the bare hardware. The essence of VM/370 is to separate these two functions.

1.3.3 Virtual Machines

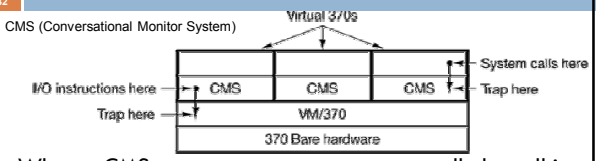
41

The heart of the system, known as the virtual machine monitor, runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up.

However, these virtual machines are not extended machines, with files and other nice features. Instead, they are exact copies of the bare hardware, including kernel/user mode, I/O, interrupts, and everything else the real machine has.

1.3.3 Virtual Machines

42



When a CMS program executes a system call, the call is trapped to the operating system in its own virtual machine. CMS then issues the normal hardware I/O instructions for reading its virtual disk or whatever is needed to carry out the call. These I/O instructions are trapped by VM/370, which then performs them as part of its simulation of the real hardware.

1.3.4 Exokernels

43

The system gives each user a clone of the actual computer, but with a subset of the resources ([0 1023], [1024 2047] and so on). The exokernel allocates resources to virtual machines and then checks any one trying to use another else's resources.

Each virtual machine thinks it has its own disk, so the virtual machine monitor must maintain tables to remap disk addresses (and all other resources). With the exokernel, this remapping is not needed.

1.3.5 Client-Server

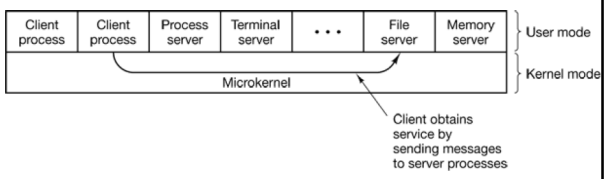
44

The aim is the leaving kernel in minimum (microkernel). The usual approach is to implement most of the operating system in client processes. To request a service such as reading a file,

1. a client process sends the request to a server process,
2. the server process does the work and sends back the answer.

1.3.5 Client-Server

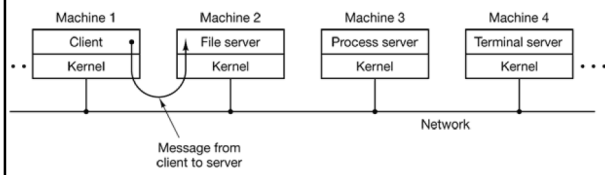
45



Because all the servers run as user-mode processes, and not in kernel mode, they do not have direct access to the hardware. As a consequence, if a bug in the file server is triggered, the file service may crash, but this will not usually bring the whole machine down.

1.3.5 Client-Server

46



The client-server model is its adaptability to use in distributed systems. If a client communicates with a server by sending it messages, the client need not know whether the message comes from its own machine or a remote machine.

1.4 Operating System Variety

47

Basically, there are seven kind of operating systems:

- Mainframe Operating Systems
- Server Operating Systems
- Multiprocessor Operating Systems
- Personal Computer Operating Systems
- Real-Time Operating Systems
- Embedded Operating Systems
- Smart Card Operating Systems

1.4.1 Mainframe OS

48

At the high end are the operating systems for the mainframes, those room-sized computers still found in major corporate data centers. These computers differ from personal computers in terms of their I/O capacity. A mainframe with 1000 disks and millions of gigabytes of data is not unusual; a personal computer with these specifications would be the envy of its friends. Mainframes are also making something of a comeback as high-end Web servers, servers for large-scale electronic commerce sites, and servers for business-to-business transactions.

1.4.2 Server OS

49

One level down are the server operating systems. They run on servers, which are either very large personal computers, workstations, or even mainframes. They serve multiple users at once over a network and allow the users to share hardware and software resources. Servers can provide print service, file service, or Web service. Typical server operating systems are Solaris, FreeBSD, Linux and Windows Server 200x.

1.4.3 Multiprocessor OS

50

A way to get high computing power is to connect multiple CPUs into a single system. According to their connection type and what they share, the systems are called parallel computers, multicomputers, or multiprocessors. They need special operating systems, but often these are variations on the server operating systems, with special features for communication, connectivity, and consistency. Many popular operating systems, including Windows and Linux, run on multiprocessors.

1.4.4 PC OS

51

The next category is the personal computer operating system. Modem ones all support multiprogramming, often with dozens of programs started up at boot time. Their job is to provide good support to a single user. They are widely used for word processing, spreadsheets, and Internet access. Common examples are Linux, FreeBSD, Windows Vista, and the Macintosh operating system. Personal computer operating systems are so widely known that probably little introduction is needed. In fact, many people are not even aware that other kinds exist.

1.4.5 Real Time OS

52

Another type of operating system is the real-time system. These systems are characterized by having time as a key parameter. For example, if a car is moving down an assembly line, certain actions must take place at certain instants of time. If a welding robot welds too early or too late, the car will be ruined. Many of these are found in industrial process control, avionics, military, and similar application areas. These systems must provide absolute guarantees that a certain action will occur by a certain time.

1.4.6 Embedded OS

53

Embedded systems run on the computers that control devices that are not generally thought of as computers and which do not accept user-installed software. Typical examples are microwave ovens, TV sets, cars, DVD recorders, cell phones, MP3 players. The main property which distinguishes embedded systems from handhelds is the certainty that no untrusted software will ever run on it. Systems such as QNX and VxWorks are popular in this domain.

1.4.7 Smart Card OS

54

The smallest operating systems run on smart cards, which are credit card-sized devices containing a CPU chip. They have very severe processing power and memory constraints. Some are powered by contacts in the reader into which they are inserted, but contactless smart cards are inductively powered, which greatly limits what they can do. Some of them can handle only a single function, such as electronic payments, but others can handle multiple functions on the same smart card. Often these are proprietary systems.

1.5 OS Concepts

55

Most operating systems provide certain basic concepts and abstractions such as processes, address spaces, and files that are central to understanding them. In the following sections, we will look at some of these basic concepts ever so briefly, as an introduction. We will come back to each of them in great detail later.

1.5.1 Processes

56

A key concept in all operating systems is the process. A process is basically a program in execution. Associated with each process is its address space, a list of memory locations from 0 to some maximum, which the process can read and write. The address space contains die executable program, the program's data, and its stack. Also associated with each process is a set of resources, commonly including registers, a list of open files, outstanding alarms, lists of related processes, and all the other information needed to run the program.

1.5.2 Address Spaces

57

Every computer has some main memory that it uses to hold executing programs. In a very simple operating system, only one program at a time is in memory. To run a second program, the first one has to be removed and the second one placed in memory. More sophisticated operating systems allow multiple programs to be in memory at the same time. To keep them from interfering with another, some kind of protection mechanism is needed. While this mechanism has to be in the hardware, it is controlled by the operating system.

1.5.3 Files

58

Another key concept supported by virtually all operating systems is the file system. As noted before, a major function of the operating system is to hide the peculiarities of the disks and other I/O devices and present the programmer with a nice, clean abstract model of device-independent files. System calls are obviously needed to create files, remove files, read files, and write files. Before a file can be read, it must be located on the disk and opened, and after it has been read it should be closed, so calls are provided to do these things.

1.5.4 Input / Output

59

All computers have physical devices for acquiring input and producing output. Many kinds of input and output devices exist, including keyboards, monitors, printers, and so on. It is up to the operating system to manage these devices. Consequently, every operating system has an I/O subsystem for managing its I/O devices. Some of the I/O software is device independent, that is, applies to many or all I/O devices equally well. Other parts of it, such as device drivers, are specific to particular I/O devices.

1.5.5 The Shell

60

The operating system is the code that carries out the system calls. Editors, compilers, assemblers, linkers, and command interpreters definitely are not part of the operating system, even though they are important and useful. The Shell is the primary interface between a user sitting at his terminal and the operating system, unless the user is using a graphical user interface. Although it is not part of the operating system, it makes heavy use of many operating system features and thus serves as a good example of how the system calls can be used.