Mohamed Kheider University Biskra
Faculty of Natural and Life Sciences Department of Computer Science

Course material

Module: Introduction to Operating Systems I

Chapter I: Introduction

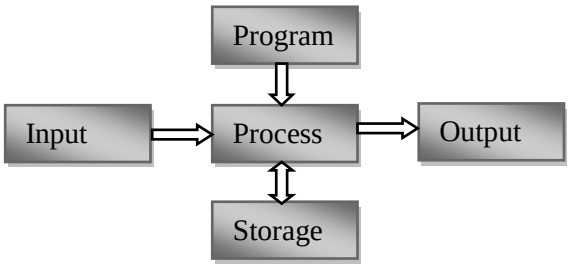
Level: 2LMD

Module leader: Dr. D. Boukhlouf

Academic year 2023/2024

1. Computer system

A computer system is a combination of hardware and software devices that help computers to receive data or information by communicating with each other. It is a set of integrated devices that accept data (input), Then process it and finally give us a result (output).



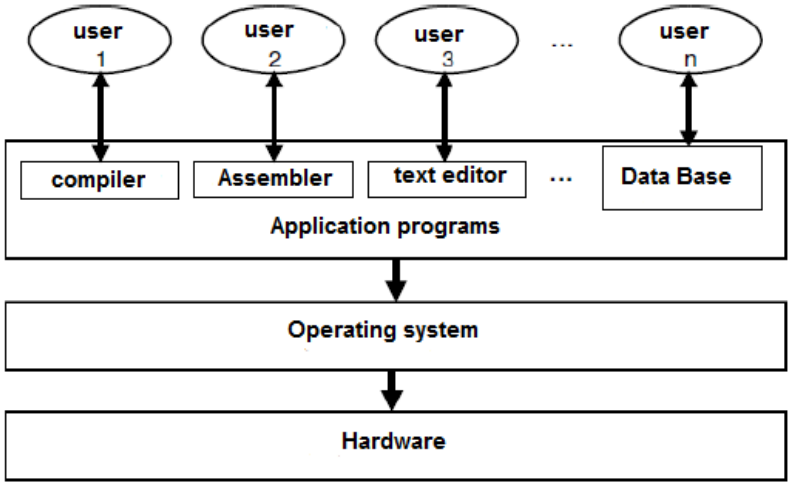
Computer system is divided to the following components:

- Hardware: physical devices as the input and output devices, CPU, RAM, SSD(Solid State Drive) and HDD(Hard Disk Drive), Motherboard, etc.
- Software: is a collection of instructions, data, or programs that are used to run and execute the programs in a computer.
- Firmware & Liveware:

Firmware is basically software that contains some basic instructions that allow hardware to communicate with other software.
Examples: USB drives, Hard drives, Remote Controls, Printers, etc.
Liveware is basically those users or people who operate a computer system using both hardware and software known as "Liveware". It is also known as "Humanware".

2. What is an operating system

An operating system is a program that should enable users to use the functionality of a computer. But it should also help the programmer to develop software in the most efficient way possible. An operating system is started as soon as the computer is turned on. In large machines, it is always running. The system is therefore an interface between the user and the physical machine.



The **operating system** manages and controls the components of the computer. It provides a base called a

virtual machine, on which application programs and utilities will be built by means of **services** or **system calls**. The purpose of an operating system is therefore to develop applications without worrying about the details of hardware operation and management. Thus, for example, a file can be read by a simple system call: read(fd, buf, 255); and it does not matter whether the file in question is on a magnetic disk, a DVD or a RAM disk.

3. Evolution of operating systems

The emergence and evolution of operating systems is linked to the evolution of computers.

3.1 First generation (1945-1955)

In the 1940s, the first "computers" appeared, with programs and data coded directly in binary on **switchboards**, then later (1950) on **punched cards** (the presence or absence of holes corresponds to 0 or 1), loaded into memory, executed and tuned from a control console. The same team designed, built, programmed, administered and maintained the machine.

The advent of **assembler** made it easier to write programs, but the use of machines is still the same. The computer only processes **one program** at a time. For each job, the cards must be inserted into a card reader, and the results printed out, which are very slow operations compared to the work of the CPU. Afterwards, it is an operator who takes care of running the machine but the principle remains the same.

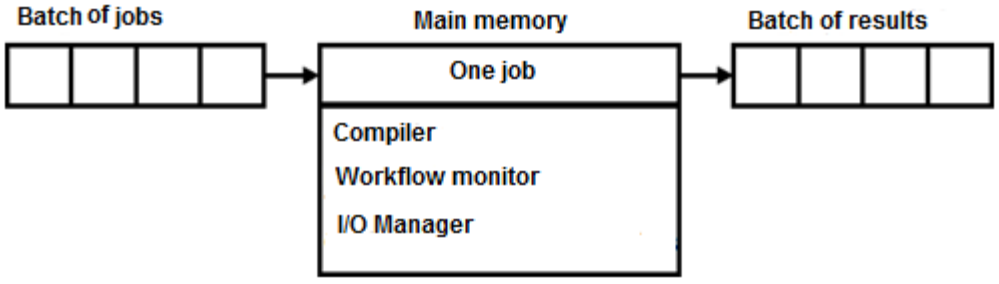
3.2 Second generation (1955-1965)

With the advent of the **transistor** (around 1955), computers are much more reliable, but very expensive (only large private or public companies own them). Different teams now design, build, program, administer and maintain the machine.

For each job to be performed, the programmer passes his or her pack of punched cards (programs in assembler or **FORTRAN**) to the operator who submits them to the compiler and then to the computer. Once the execution is completed, the operator retrieves the result on the printer. As computers are very expensive, the aim is to reduce waiting times by automating manual operations.

Large computers then have three tape drives: one for the operating system, one for the program and its data, one for the output data. In addition, there are machines for transcribing punched cards (brought by the programmers) onto tape and machines for printing the output data (contained on the tape) onto paper. This is called off-line printing (the printer is not directly connected to the computer). But the main computer does not work all the time: a lot of time is still wasted by the operators who are in charge of feeding the machines with cards, paper and tapes.

Batch processing is used: several jobs are transcribed onto the same input tape. The main computer reads the 1st job, then, when it is finished, reads the 2nd, etc. until the end of the tape. A command interpreter allows the loading of the program and data and the execution of the program, the monitor is the program in charge of the sequencing of the users' jobs and the continuity of the operations. The monitor of the 1960s was the precursor of the operating system.



3.3 Third generation (1965-1980)

In the mid-1960s, the emergence of **integrated circuits** allowed a great advance in the performance and cost of computers. Families of machines sharing the same machine language and the same operating system wereset up.

The jobs are processed entirely by the computer without passing through any ancillary machines; the jobs are stored on disk files. As access to a disk is random (in the sense of non-sequential) the monitor can choose the order of the jobs.

In the monitor, the scheduler performs this task. But batch monitors always run one job at a time, at any given time only one program is in memory and can use the CPU. This leads to **multi-programming**. The memory is shared between different programs waiting for an input data for example can be temporarily suspended in favor of a job. The aim is to exploit the CPU as much as possible. This technique is called **spooling** (from SPOOL: Simultaneous Peripheral Operation On Line). In the operating system, the allocator (dispatcher) takes care of the instantaneous management of the CPU, taking into account the planning established by the scheduler. It was necessary to set up systems for controlling memory access and data protection.

Time-sharing is also emerging for multi-user computers (a central computer connected to several terminals).

3.4 Fourth generation (1980-present)

Since the 1980s, we have witnessed the development of personal computers, the constant improvement in development of communications networks, the development of local networks, and the explosion of the Internet. Human/machine interfaces are always being improved.

3.5 Fifth Generations of Computers (Present and Beyond)

The Fifth generation of computers is in a development face that uses AI (Artificial Intelligence).
This types of computer will be very smart. They will make their own decision.
When given a condition, they will also have the ability to reason as human beings do with parallel processing and superconductors working together to develop "Artificial Intelligence".
They will communicate with humans with the help of languages, pictures, sign language, speech, and writing.
They will have a huge amount of knowledge stored in their memory so that they can respond to the inputs given by humans and take appropriate action whenever necessary.
They will be extremely intelligent machines. You can get an idea of (AI) used in the Voice recognition system.

4. The main functions of an operating system

The roles of the operating system are diverse:

· **Processor management**: the operating system is responsible for managing the allocation of the processor to different programs through a **scheduling algorithm**. The type of scheduler is totally dependent on the operating system, depending on the objective.

· **RAM management**: The operating system is responsible for managing the memory space allocated to each application and, if necessary, to each user. If there is insufficient physical memory, the operating system can create a memory area on the hard disk, called "**virtual memory**". Virtual memory allows applications to run that require more memory than is available on the system. On the other hand, this memory is much slower.

· **I/O management**: the operating system unifies and controls program access to hardware resources through drivers (also called device drivers or I/O managers).

· **Application execution management**: the operating system is responsible for the proper execution of applications by allocating the resources necessary for their proper functioning. In this respect, it allows "kill" an application that is no longer responding correctly.

· **Rights management**: The operating system is responsible for the security of running programs by ensuring that resources are only used by programs and users with the appropriate rights.

· **File management**: The operating system manages reading and writing to the file system and the access rights to files by users and applications.

· **Information management**: The operating system provides a number of indicators to diagnose the correct functioning of the machine.

5. Components of an operating system

The operating system is composed of a set of software programs that manage interactions with the hardware. Among this set of software, the following elements are generally distinguished

· The **kernel** represents the fundamental functions of the operating system such as memory management, processes, files, main I/O, and communication features.

· The **command interpreter** (*shell*, as opposed to kernel) that allows communication with the operating system through a command language, so that the user can control the devices without knowing the characteristics of the hardware he is using, the management of physical addresses, etc.

· The *file* **system** (*FS*), which allows files to be stored in a tree structure.

6. The different types of OS

5.1. Multi-tasking systems

An operating system is *multithreaded* when several "**tasks"** (also called *processes*) can be executed simultaneously.

Applications are composed of a sequence of instructions called "**light processes**" (in English *"threads*). These threads will be alternately active, waiting, suspended or destroyed, depending on the priority associated with them or executed sequentially.

A system is said to be **preemptive** when it has a **scheduler** (also called a *planner*), which allocates machine time according to priority criteria to the various processes that request it.

The system is said to be **time-sharing** when a time quota is allocated to each process by the scheduler. This is particularly the case for multi-user systems which allow several users to use different or similar applications simultaneously on the same machine: the system is then said to be "**transactional**". To do this, the system allocates each user a time slot.

5.2. Multi-processor systems

Multiprocessing is a technique of running several processors in parallel to obtain more computing power than a high-end processor or to increase system availability (in case of a processor failure).

SMP (*Symmetric Multiprocessing*) is an architecture in which all processors access a shared memory space.

A multiprocessor system must therefore be able to manage memory sharing between several processors and also to distribute the workload.

5.3. Embedded systems

Embedded systems are operating systems designed to run on small machines, such as PDAs (*personal digital assistants*) or autonomous electronic devices (space probes, robots, vehicle computers, etc.), with limited autonomy. Thus, a key feature of embedded systems is their advanced energy management and their ability to operate with limited resources.

The main "consumer" embedded systems for PDAs are

· PalmOS

· Windows CE / Windows Mobile / Window Smartphone

5.4. Real-time systems

Real time systems, mainly used in industry, are systems whose purpose is to operate in a time constrained environment.

A real-time system must therefore operate reliably within specific time constraints, i.e. it must be able to deliver correct processing of the information received at well-defined time intervals (regular or not).

Some examples of real-time operating systems are

- OS-9 ;

· RTLinux (RealTime Linux) ;

· QNX ;

· VxWorks.

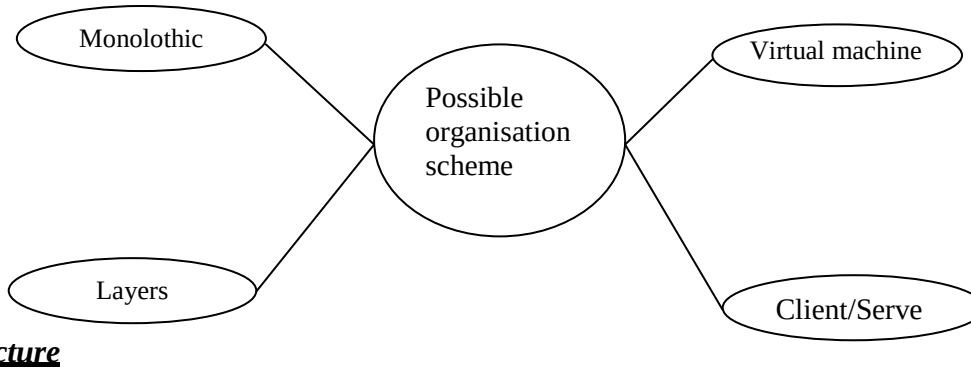
5.5. Examples of operating systems

A distinction is made between different types of operating systems, depending on whether they are capable of handling 16-bit, 32-bit, 64-bit or more information simultaneously.

system	Coding	Single user	Multi user	Single task	Multi task
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			no preemptive
Windows95/98/Me	32 bits	X			cooperative
WindowsNT/2000	32 bits		X		preemptive
WindowsXP	32/64 bits		X		preemptive
Unix / Linux	32/64 bits		X		preemptive
MAC/OS X	32 bits		X		preemptive
VMS	32 bits		X		preemptive

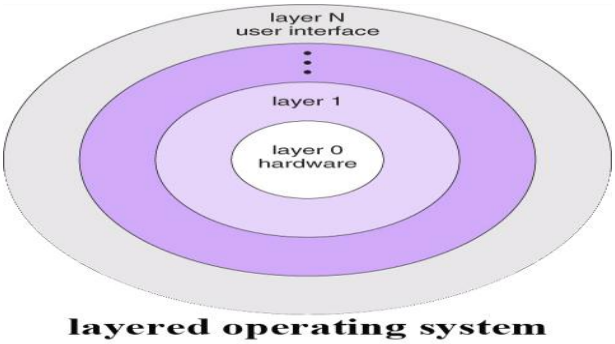
7. Internal structure of an operating system

The operating system, called the kernel, manages the hardware and provides programs with a system call interface. System calls allow programs to create and manage processes and files.



6.1. Layered structure

The operating system is structured in layers. Each layer uses the functions of the lower layers. The main difficulty is the definition of the different layers. For example, it can be organised into six layers, as shown in the following figure:



layered operating system

– At the lowest level is the kernel, the interface between the hardware and the software. It is responsible for managing the CPU, interrupts and processes (communication and synchronisation) using the functions

provided by the hardware. It must reside entirely in memory.

– At the second level is the memory manager, which is responsible for sharing memory between processes waiting to run.

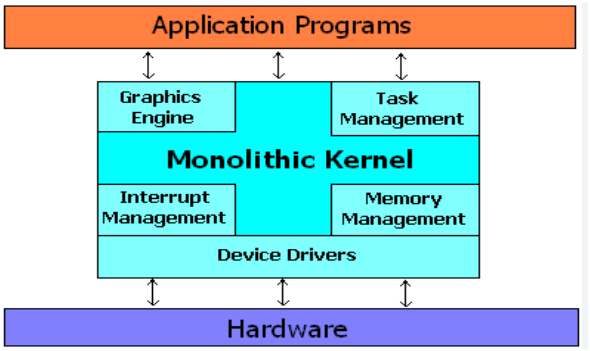
– At the third level, there is the I/O management module, which is responsible for managing all peripherals (keyboard, screen, disks, printers, etc.).

– The fourth level is the file manager, which is responsible for managing disk space, manipulating files while ensuring data integrity, protecting files, etc.

– At the fifth level, there is the resource allocation module, which is responsible for ensuring the proper use of resources; accounting for and providing statistics on the use of the main resources; creating new processes and assigning them a priority level; allowing each process in the system to obtain the necessary resources within a reasonable time limit; mutually excluding processes that require a non-shareable resource and avoiding blocking situations

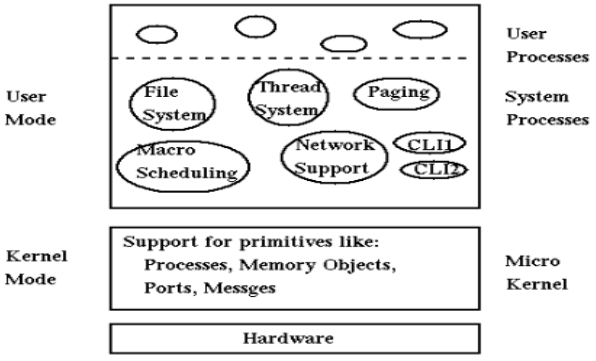
6.2. Monolithic structure

Operating systems under a monolithic structure are a set of procedures of - almost - the same level: a main procedure that calls the required service procedure, service procedures that execute the system calls and a set of utility procedures that assist the service procedures, e.g. retrieving data from user programs. Unix and Linux are examples of monolithic systems.



6.3. Micro-kernel

A more modern architecture than the monolithic one is the micro-kernel architecture used in MACH5/HURD6, Minix and NT. The main attribute that distinguishes micro-kernels from monolithic kernels is the implementation of their respective architectures in supervisor mode (kernel mode) and user mode (user mode). The monolithic architecture implements all the services of the Operating System (kernel controllers, network controllers devices, virtual memory, file system, networks, etc.) in the supervisor mode domain of the CPU.



On the other hand, the micro-kernel architecture makes a division between the services of the Operating System, dividing them into "high-level" implemented in the user domain and "low-level" implemented in the supervisor mode space.

6.4. The client/server model

In the client/server model, the operating system consists of a kernel and a set of servers. The kernel manages the communication between the clients and the servers. The clients are the service requesters. For example, to request a service, such as reading a block from a file, a user process (also called a client process) sends a request to a server process, which does the work and returns a response. Servers run in user mode. Therefore, they cannot access the hardware directly. Therefore, an error or bug in the file server, for example, will not usually affect the whole machine. The damage is limited to the server.

6.5. Virtual machines

The core of the system takes care of multi-programming by providing the layer above with several virtual machines. Different operating systems can run on top of a virtual machine.
Examples of virtual machines:
– IBM VM: provides each user with their own single-task virtual machine. The virtual machines were scheduled with time sharing.
– Java: Programs compiled in Java run on a virtual machine (JVM).
– VMWare7: on PC, runs Windows, Linux, OS/2, etc. sessions at the same time.
– Nachos: OS running in a MIPS virtual machine, running on Unix.