University of Tlemcen

Computer Science Department

1st Year Engineer

"Introduction to the operating systems 2"

Session 2: Introduction to operating systems

Academic year: 2023-2024

Objectives

- Principles and functioning of an operating system.
- Understand the role of an operating system and its importance for the performance of a machine.

Outline

- Why an operating system?
- Presentation of an operating system.
- Principles and functioning of an operating system.
- The role of an operating system

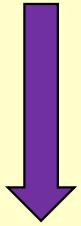
Why an operating system?

- Example of an oven with some digital functions.
  - *An LCD screen*
  - *Buttons*
  - *Timer*
  - *Alarm*
  - *Heating system*
- The furnace's computer system is built around a microcontroller, it :
  - *reads the status of the buttons,*
  - *configures the timer,*
  - *displays the remaining time on the display, and controls the heating device.*
- This program is run on the microcontroller to perform these functions.

- Example: a complex computer system linked to a smartphone.
  - *On his journey, the owner uses the following applications:*
    - *GPS navigation*
    - *Music streaming*
    - *SMS communication*
    - *Receiving a call*
  - *These applications are developed by different and sometimes competing companies and run on smartphones from different manufacturers.*

- *These applications use the smartphone's hardware resources:*
  - *The processor*
  - *The memory*
  - *The screen*
  - *The touch screen*
  - *The speaker (for streaming and notifications from other applications)*
  - *GPS reception (Navigation)*
  - *4G or 5G communication*
- *Phone calls take priority over other applications*
  - *The arrival of a call will interrupt music streaming and audio instructions from the messaging system.*

- Interactions between applications and smartphone components.
  - *These interactions raise two main issues:*
    - *How do software publishers develop applications that are compatible with the electronics of all smartphones?*
    - *How applications share resources equitably (for example, not all applications can be activated on screen at the same time).*



  - *The solution is to use a software layer between the hardware (smartphone components) and the applications (software).*
  - *This layer is called the "operating system".*

- By acting as an intermediary between applications, the CPU, memory and hardware peripherals, the operating system offers standardized functionality.
  - *This layer of abstraction allows software publishers to develop for an OS and not for hardware.*
  - *The OS allows these applications to run in parallel without any knowledge of each other.*

Benefits of using an OS

- The benefits of using an OS or a general-purpose computer system (such as a smartphone)
  - *An analogy with a computer*
- Any computer system with a few kilobytes of memory can host an operating system.
- Hardware resources can be divided into 5 categories:
  - *Processor*
  - *Memory*
  - *Secondary memory (storage)*
  - *Input/output (keyboard, screen, etc.)*
  - *Communication interfaces (Ethernet, WiFi, etc.)*

- *The OS is close to the hardware,*
- *It is designed for specific architectures,*
  - *The Windows operating system does not run on a small processor*
- *Recent processors have two modes:*
  - *Supervisor mode*
  - *User mode*
- *The OS operates in supervisor mode, so it can control access to the machine's resources.*
- *On the software side, applications run in user mode interact with the hardware only through the OS.*
- *The OS :*
  - *Schedules executions to share the processor,*
  - *Allocates memory space,*
  - *Protects resources from simultaneous access by applications,*
  - *Manages the permissions of different users,*
  - *Does not allow access to restricted areas or misconfigure peripherals.*

What is an operating system?

- An operating system (OS) is a set of programs that run permanently on a computer and control it from the moment it boots up and for as long as it is switched on.
  - *The main one is called the kernel (heart of the operating system),*
  - *It allocates resources,*
  - *Most access to memory and peripherals is via the processor.*
  - *The kernel and processor pair is the center of any system.*
  - *For memory management, the OS uses the Memory Management Unit (MMU).*
  - *Applications pass through the kernel via system calls*

Examples of operating systems

- *Unix: Created in 1969, rapidly multi-user, written in the C language.*
- *MS-DOS (Microsoft disk operating system): OS of the first PCs, single user, single task, command line interface.*
- *MacOS: derived from Unix*
- *Windows*
- *Android, IOS*
- *QNX, RTOS: for embedded systems.*

- Exemples :
  - *Unix : Créé en 1969, rapidement multi-utilisateur, écrit en langage C.*
  - *MS-DOS (Microsoft disque operating system) : SE des premiers PC, mono-utilisateur, mono-tâche, interface ligne de commande.*
  - *MacOS: dérivé de Unix*
  - *Windows*
  - *Android, IOS*
  - *QNX, RTOS: pour les systèmes embarqués.*

The kernel

- The resident part (always in RAM) of the OS is called the Kernel.
- The other parts are brought into RAM as required
- Contains the critical functions of the OS: they must always be ready for use
  - *Interruption handling*
  - *UCT management*
  - *Memory management*
  - *Inter-process communication*
  - *etc.*

- Computer: Without the software, an useless machine
- Two types of software:
  - *System programs: manage the operation of the computer*
    - *Examples:*
      - *Managing hardware interruptions*
      - *Reading data from a USB flash drive*
      - *Sharing memory between processes*
      - *Display a diagnosis of errors encountered during*
  - *Application programs: carry out work requested by users*
    - *Examples:*
      - *Configuring a printer*
      - *Interpret user commands*
      - *Archiving and restoring files*
      - *Compressing data in a file*

Terminology related to the OS

- *Hardware : Basic computer resources: CPU (central processing unit), memory, I/O devices.*
- *Operating system: Intermediary program between the user and the hardware.*
- *Application programs: How to use the hardware to solve users' computing problems.*
- *Users: People, machines, other computers.*
- *Distributed system: Allows the user to share resources on a network.*
- *Data processing system: A data processing system is a set of computer resources whose purpose is to produce, process, store, route, present or destroy data.*
  - *There are three classes:*
    - *Interactive system (operating system),*
    - *Reactive system (real time),*
    - *Transformational system (compiler).*

Roles of OS

- The operating system plays two roles:
  - *a virtual machine (abstract)*
    - *The OS provides the programmer with an interface for accessing the computer's resources (in the form of system calls). In this way, the programmer can abstract the details of how the resources work.*
    - *This interface is based on abstract objects, the most important of which are files and processes. For example, the programmer sees a disk as a collection of files that can be read, written and closed.*
  - *a resource administrator*
    - *The OS manages the use of resources by different users and any conflicts.*

Aims of an OS

- Provide an environment where the user can run programs.
- To make the computer system practical for the user.
- Use the hardware efficiently.

Evolution of OS

- OS: developed to make the hardware easier to use.
- Project and use of OS led to changes in hardware.
- Historical view of OS: problems with OS led to hardware innovations.
- OS has a long history, from the time it began to replace computer operators to today's multi-programming systems.

Tasks of an operating system

- *Process management*
- *Memory management*
- *File management*
- *I/O management*

- User programs can access these different functions using system calls.
- To create a system as large and complex as an OS, it needs to be broken down into smaller parts.

Operating modes

- The hardware allows two modes of operation:
  - *Monitor mode, supervisor mode, system mode: execution by the OS.*
    - *Privileged instructions: machine instructions that may cause harm.*
    - *Privileged instructions are only executed in supervisor mode.*
  - *User Mode: execution by the user.*
    - *If an attempt is made to execute a privileged instruction, the hardware does not execute it but treats the instruction as illegal and blocks the OS.*

- Certain instructions are privileged, such as I/O instructions.
- So how can a user program execute I/O?

System calls (1)

- A system call is a function provided by the kernel of an OS and used by programs running in user space.
- The role of the kernel is to manage hardware resources and provide programs with a uniform interface for accessing these resources.
- Some classic system calls :
  - *open, read, write and close for manipulating file systems,*
  - *alloc, free to allocate and deallocate memory.*

System calls (2)

- System calls are functions
  - *called from a program in user space*
  - *whose execution (processing) takes place in kernel space ;*
  - *returned by the calling program in user space.*
- In addition to a change of execution mode, a system call involves at least two context switches:
  - *Context of the calling program ;*
  - *Kernel context ;*

- On most operating systems, system calls can be used as simple functions written in C.
- On most kernels (particularly monolithic kernels such as the Linux kernel), system calls are implemented by a machine instruction (interupt, supervisor call, etc.) which switches the processor in the kernel to supervisor mode (having passed the system call parameters, for example in the registers).

Examples of system calls

- A system call is an atomic operation (it may or may not have been executed in error).
- Categories of system calls :
  - *Process control:*
    - *load, execute, create, terminate processes, report events, free memory, etc.*
  - *File manipulation:*
    - *create, delete, open, close, read, write, reposition, etc.*
  - *Device management:*
    - *request, release, get, attach, etc*

Purpose of the SO

- An OS provides the necessary environment for programs to run normally.
- To do this, the OS:
  - *Allocates hardware and software resources to meet the needs of the programs.*
  - *Presents programs with an interface better suited to their needs than that provided directly by the hardware.*

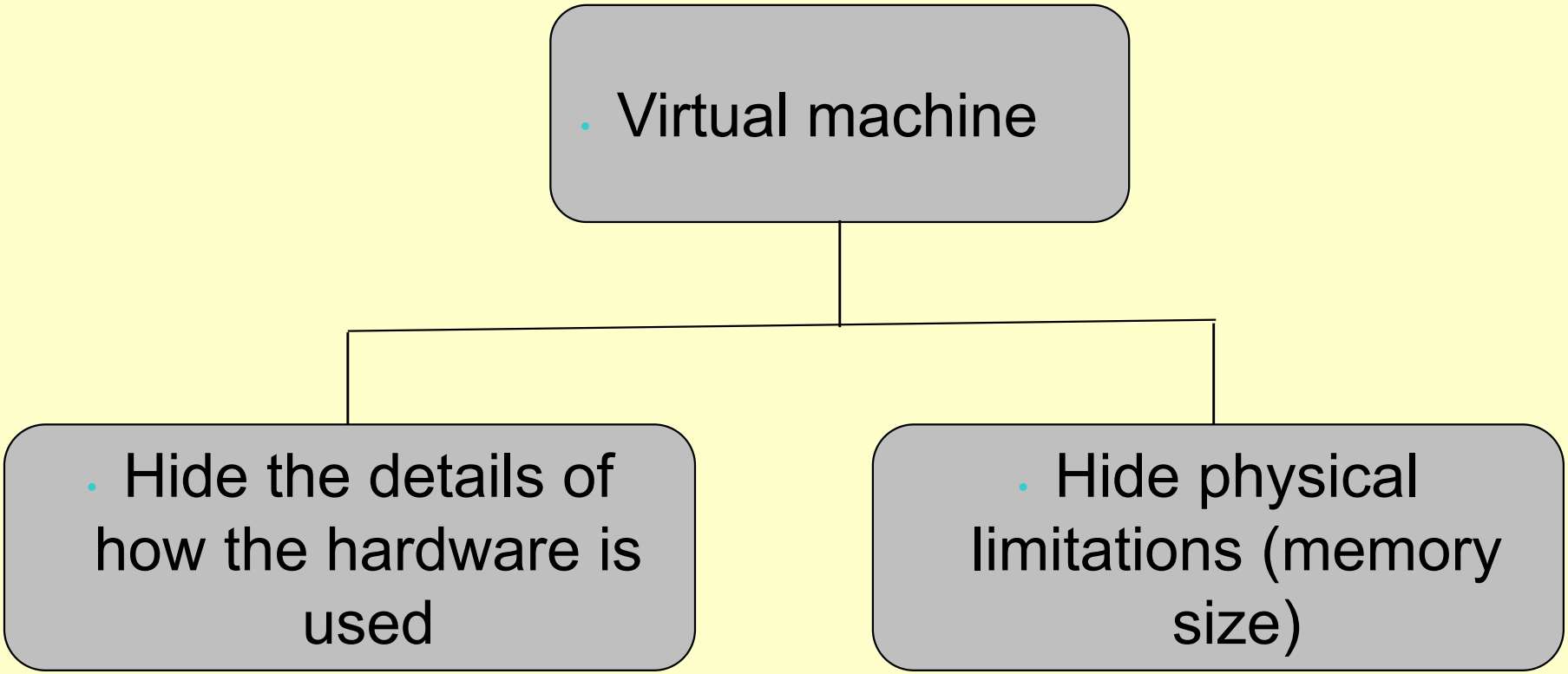
The OS as a virtual machine

Few programs would be developed if every programmer had to know how the hardware worked.

Finding a way to free programmers from the complexity of hardware

ABSRACTION

- The virtual machine part of operating systems:
  - *Subtracts the hardware from the programmer's view,*
  - *Provides a nice, simple view of named files that can be read and written to.*


Virtual machine
Hide the details of how the hardware is used
Hide physical limitations (memory size)

The OS as a resource manager

- Modern computers are made up of processors, memories, clocks, disks, network interfaces, printers and other peripherals that can be used by several users at the same time.
- The job of the OS is to control and order the allocation of processors, memory and peripherals to the various programs that use them.

- Imagine what would happen if three programs running on one computer tried simultaneously to print their results on the same printer.
  - *The first lines printed might come from program 1 the next from program 2, then from program 3 and so on.*

  → *The result would be total disorder.*
- The operating system can avoid this potential chaos by transferring the results to be printed to a buffer file on disk.

  → *When a print completes, the operating system can then print one of the files in the buffer.*

Multi-tasking systems

- Most modern operating systems allow several tasks to be carried out at the same time.
  - *For example, while running a user's program, a computer can read data from a disk or display results on a terminal or printer.*
  - *This is known as a multi-tasking or multi-programmed operating system.*

- The fundamental concept of multi-tasking operating systems is the process.
- A process is an instance of a program that is currently running
  - *A process is represented by a program (the code),*
  - *but also by its data and by parameters indicating where it is at, allowing it to continue if it is interrupted (execution stack, ordinal counter...).*

    → *This is referred to as the program environment.*

Multi-tasking systems: time-sharing

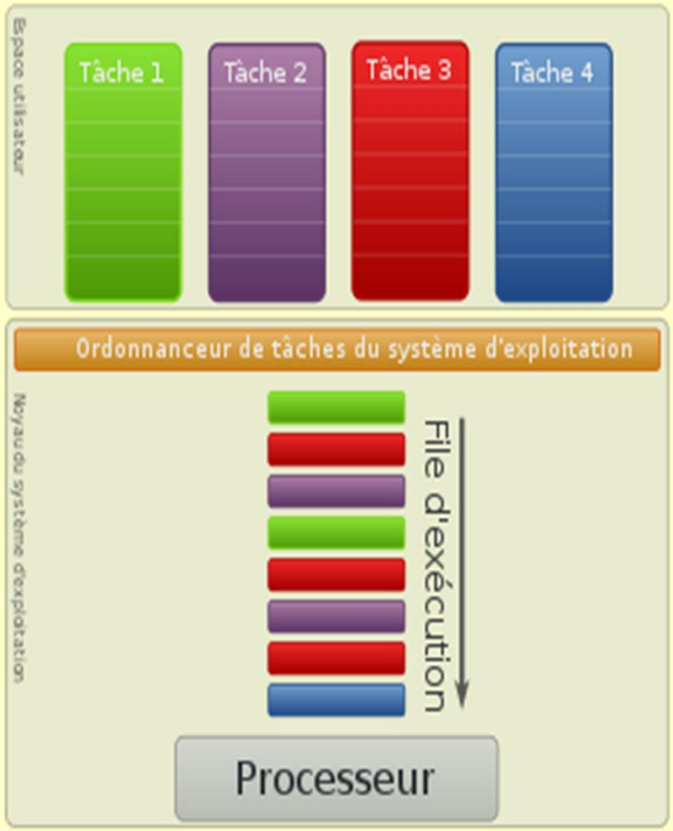
- Most multi-tasking operating systems are implemented on a computer with a single microprocessor.
- At any given time, this microprocessor is really only running one process, but the system can switch it from one program to another.
- This gives users the impression that all programs are running at the same time.

Multi-user systems

- As with multi-tasking systems, multi-use is emulated by allocating time slots to each user.
- Naturally, switching from one application to another slows down each of them and affects the response time perceived by users.

Task Manager/Scheduler

- An operating system scheduler only makes sense in a multitasking system.
- It manages the order in which instructions for different tasks are executed.
- It is responsible for saving and restoring the task context (this context is made up of the processor registers), also known as context switching.



Scheduler: Context switching

- Context switching involves saving the state of one process and restoring the state of another as part of the scheduling of a multitasking operating system.
- Context switching involves at least three steps. For example, assuming you want to switch processor usage from process P1 to process P2:
  - *Save the context of process P1 somewhere in memory (usually on P1's stack).*
  - *Retrieve P2's context from memory (usually from P2's stack).*
  - *Restore P2's context in the processor, with the last stage of restoration consisting of resuming P2's execution at its last execution point.*

Scheduler

- Most recent schedulers allow you to specify which processor the tasks are executed on.
- Some also allow tasks to be migrated to other machines on a computing grid.
- The scheduling algorithm determines which task should be executed first and on which processor.
  - *This algorithm must enable the machine's resources to be used efficiently.*

- Scheduling can be cooperative
  - *Tasks must be written in such a way that they co-operate with each other and accept suspension for the execution of another task.*
- Scheduling can also be preemptive
  - *The scheduler is responsible for interrupting tasks and choosing the next one to run.*
- Some kernels are themselves preemptive
  - *The scheduler can interrupt the kernel itself to make way for an activity (typically, still in the kernel) with a higher priority.*

OS as a memory manager

- The memory manager is the subset of the operating system that manages the computer's memory.
- Its most basic task is to allocate memory to processes when they need it.

- The memory manager
  - *Hides the physical location of the memory (in RAM or on the hard disk, in the paged memory space)*
  - *Presents the program with a uniform global memory known as virtual memory.*
- In this way, any process believes it is manipulating "logical" memory, which has the following properties:
  - *The memory can be expanded to the theoretical capacity of the machine;*
  - *Memory is private (protected): a process cannot access the memory of another process (except for specific allocations and authorizations).*

File manager

- One of the fundamental tasks of the operating system is to hide the specifics of disks and other I/O devices and provide the programmer with a nice, easy-to-use model.

  → This is done through the notion of file

Peripheral manager

- Controlling the computer's input/output (I/O) peripherals is one of the primary functions of an operating system.
  - *It must send commands to peripheral devices, intercept interrupts and handle errors.*
  - *It must also provide a simple, easy-to-use interface between the peripherals and the rest of the system, which should, as far as possible, be the same for all peripherals, i.e. independent of the peripheral being used.*

- Many operating systems offer a level of abstraction that allows users to perform I/O without going into the details of the hardware.
  - *This level of abstraction makes each device appear as a special file, allowing I/O devices to be treated as files.*

The command interpreter

- The operating system itself is the code used to define system calls.
  - *System programs such as text editors, compilers, assemblers, link editors and command interpreters are not part of the operating system.*
  - *The command interpreter executes an infinite loop that displays "command prompt" (indicating that something is expected), reads the name of the program entered by the user at that time and executes it.*