Computer Systems

UD 05. FUNDAMENTALS OF OPERATING SYSTEMS

Computer Systems
CFGS DAW

Sergio García / Alfredo Oltra
sergio.garcia@ceedcv.es
alfredo.oltra@ceedcv.es
2022/2023

Versión:220712.1009

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

INDEX

UD05. FUNDAMENTALS OF OPERATING SYSTEMS

1. INTRODUCTION

An operating system is the basic software to manage the computer. It manages all hardware resources, hiding complexity to final user, who see a graphical interface. Also, it is responsible to run applications. We can say that operating system is between hardware and applications.

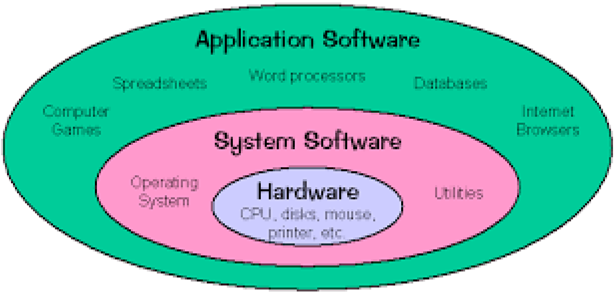


Figure. Hardware, operating system and applications

2. HISTORY

First computers didn’t have an operating system and programmes interacted with them modifying the hardware.

The operating system concept appear in 50s, doing basic task like running a program and when it finishes, run another program.

In 60s main concepts of operating systems are developed, like multi-user, multi-task, multi-processor and real time operating systems. At late 60s Unix appear.

In 70s computers arrive to a lot of new users (instead of only governmental agencies and big corporations) and tools like C programming language were born (Unix was re-writen in C).

In 80s computers try to be more human-friendly, and it starts the graphical interface development.

In 90s appear operating systems like Linux or Windows. They are the most used nowadays.

In 2000s appear other popular operating systems like Android or IOS.

3. OPERATING SYSTEM CLASSIFICATION

We can classify operating systems by:

3.1 Number of users

- **Mono-user**: only one user can work with one computer at the same time.
- **Multi-user:** several users can work with one computer at the same time.

3.2 Number of processors

- **Mono-processor**: operating system only supports one processor.
- **Multi-processor:** operating system supports more than one processor.
  - **Symmetric**: process are distributed through all processor.
  - **Asymmetric**: one or several processors are in charge of system process and one or several processors are in charge of user process.

3.3 Number of task

- **Mono-task**: operating system only can manage one task.
- **Multi-task:** operating system can manage several tasks at the same time.

> A mono-processor operating system could be multi-task? Yes, it could. One processor only can do an operation at the same time, but there are techniques to change quickly between task, and it does the effect that we are running several tasks sat the same time. That technique is used for most popular operating systems.

3.4 Operating systems classification

Operating systems	Number of users	Number of task	Number of processors
MS-DOS	Mono	Mono	Mono
Windows 9x, Me	Mono	Multi	Mono
Windows XP/Vista/7/8/10	Mono/Multi	Multi	Multi
UNIX, Linux, Windows NT Server, Windows Server	Multi	Multi	Multi

4. MOST POPULAR OPERATING SYSTEMS

Figure. Logos of most popular operating systems.

In this point we are going to talk about most popular operating systems. Several of them are free software (you can obtain source code and modify it, and usually you don't have to pay for them), but others are privative software (source code is secret, and you have to pay for them).

Nowadays, the most popular operating systems are:

- Linux: very popular at server and desktop computers. It is free software.
- Windows systems: they are popular at server and desktop computers. They are privative software.
- Mac OsX: for use in Mac computers, based on BSD. It is privative software.
- Android: for use in mobile devices, usually mobile phones and tablets. It is free software.
- Apple IOS: for use in Apple mobile devices, often iPhone and iPad. It is privative software.

5. OPERATING SYSTEMS FUNCTION

We are going to describe the main functions of an operating system.

A) Process management: the operating system manage processes in order to decide which of them uses the CPU.

B) Memory management: the operating system manages memory to decide organization and limits for each process.

C) Input / Output devices management: the operating system manages I/O operations.

D) File system management: the operating system manages how data is organized by a file system.

On next points those functions will be detailed.

6. PROCESS MANAGEMENT

Process management is a problem of multi-task systems. Mono-task system doesn't have that problem because a process always uses all the CPU.

6.1 What is a process?

A process is a program running that needs to be allocated in memory, and it needs to use CPU. Also, it can use other resources like I/O devices. A good example is the difference between a recipe (the program) and cook that recipe (the process).

> A process is not a program. A program is a set of an executable, data, resources, ... but a process is a running program. Usually when you run a program, it only creates one process, but some programs create more than one process.

6.2 Process states

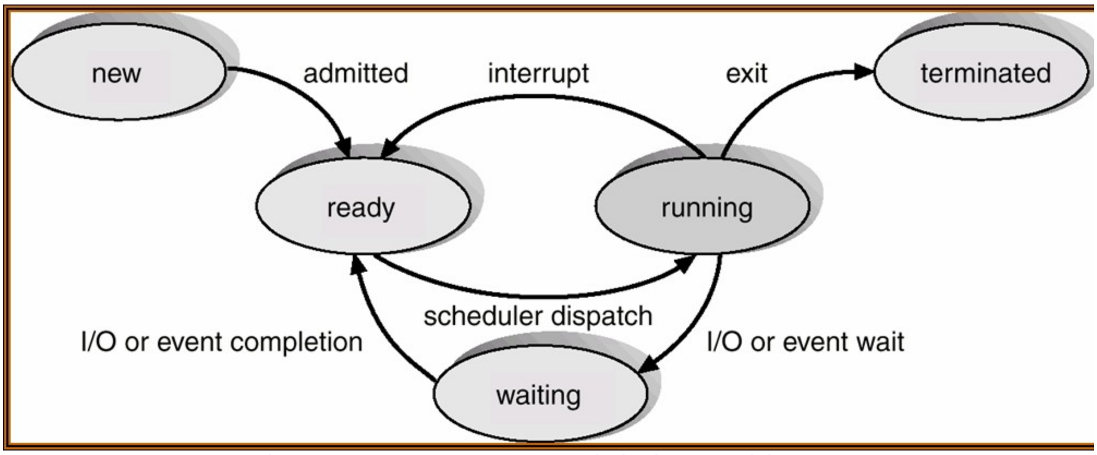


Figure. Process states and their transitions.

The state of a process defines what is its situation currently. We are going to define process states and their transitions.

- New: when a program is run, there is a system program call "long term scheduler" that decides if a process is admitted or not (It depends usually on memory available, number of processes not to high, ...).
  - New → Ready: when "long term scheduler" accepts the process, it goes to ready state.
- Ready: the process has all the resources that need to use the CPU. But it isn't using the CPU. It has to wait to be selected by a system program called "short term scheduler".

- Ready → Running: when "short term scheduler" select our process, it goes to running state.

- Running: the process is using CPU and doing its operations.
    - Running → Waiting: process needs a I/O operation to continue and voluntary leaves CPU.
    - Running → Ready: process is ok to run, but "short term scheduler" decides it has to leave CPU.
    - Running → Terminated: process did its last operation and leaves CPU.
- Terminated: the process has ended its execution.
- Waiting: the process is waiting for a I/O operation (for example, waiting for a data read from a hard disk).
    - Waiting → Ready : I/O operation has finished, and the process is ready to use CPU.

6.3 Short term scheduler algorithms

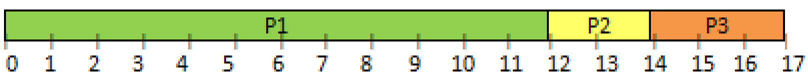
Short term scheduler has to decide which process is using CPU. If computer has only one processor, only one process could be using it in a time instant.

There are several algorithms to decide which process uses CPU and try to do it as equitable as possible. Those algorithms could be modified setting priority (making several process more important than others).

> We are going to study several algorithms, but modern computers use Round robin and modifications of it.

- **FIFO (First in First Out):**
    - It is like a supermarket queue. The first arrives, the first served.
    - Example of use:

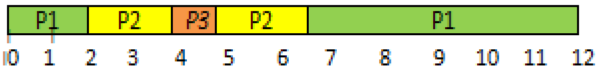
Process	Arriving time	CPU
P1	0	12
P2	2	2
P3	5	3



- **Shortest Remaining time First**
    - The process who needs less CPU time is served. In this algorithm we suppose an incoming process can expel a running process.
    - This algorithm is not viable for two reasons:

    - We can’t predict the duration of process.
    - It can produce starving (if constantly are arriving short processes to CPU, a long process will never use CPU)
  - Example of use:

Process	Arriving time	CPU
P1	0	7
P2	2	4
P3	4	1



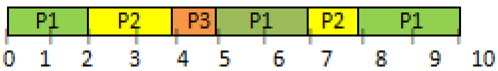
- **Round Robin:**
  - It uses a concept called quantum. Quantum is the number of CPU instant that can be done if a process is waiting for use CPU. It is equitable and it (and modifications of it) are the most used in real operating systems.
  - Example to understand it: It is like if you have to rent a football pitch. If quantum is 2, you can rent it for two hours. If other users are waiting, when you finish they can't rent the pitch for two hours, but if nobody if waiting, you can rent it for 2 hours again.
    - If a team leaves before using 2 hours (they only play 1 hour) the pitch could be rented again.

> 🗲 If there is 1 process in CPU, and it loses its quantum and at the same time a new process arrive, the process that was in CPU has preference.

  - Example of use:

(Quantum = 2):

Process	Arriving time	CPU
P1	0	6
P2	0	3
P3	0	1



> 🗲 Actually, process can’t arrive at the same time. But if in a theoretical exercise several processes arrive at the same time (like the example), you can decide the order they go to processor.

7. MEMORY MANAGEMENT

Memory management is a problem of multi-task systems. Mono-task system doesn't have that issue because a process always uses all the memory available.

Memory is a highly demanded element: there is little quantity and all the processes want a lot. The management of how to distribute it has been modified since the appearance of the first operating systems, according to the amount of memory available and the number of concurrent processes, which is increasing.

Usually a portion of RAM is called page. The size of the page depends on the operating system implementation.

Memory management is the part of operating system that solves those problems:

- **Protection problem**: a process should not invade other process memory space. To solve this problem, operating system detects if a process is using more than the provided space and if it does, it raises an error.
- **Reubication problem**: when a program is compiled it doesn't know in which memory positions it is going to run. It depends on memory state, how many processes are running, ...
  - To solve this problem, a program always thinks it has a "virtual memory space", starting in address 0 with all memory available.
  - When a program is running, the operating system transforms the addresses of its virtual space to virtual addresses that operating system uses internally. Virtual addresses are explained in next point.
- **Assignation problem:** when the processes finish, they go out and free the memory space they occupied, creating gaps in memory. Therefore, it is possible that there was plenty of space in memory for a new process, but maybe that space is not continuous. The operating systems solve this problem by this way: it uses virtual addresses, which instead of being one-dimensional (3, 56, 2340 ...) are bidimensional ((23,12567) (656,12300), (45,4))) where the first field indicates the direction of Initial memory (of the real memory) and the second the offset with respect to the first. With this trick, the program calls, for example, the virtual direction (45,452), and the Operating System convert it in the real direction 497.

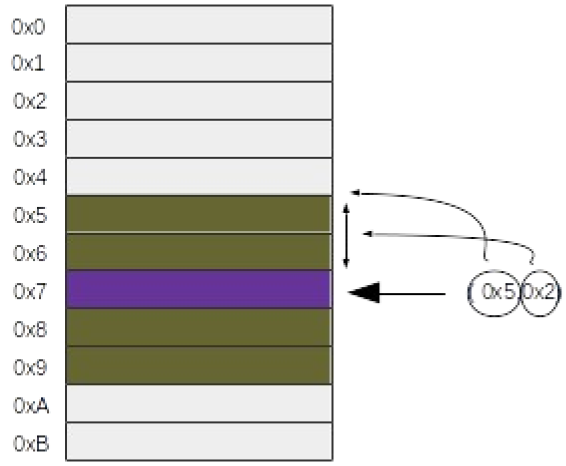


Figure. Address 0x5 and 0x2 offset.

- **Low memory available problem (Swapping)**: when we need to use more memory than physically available, we can use a technique called swapping. This technique uses other devices (like Hard disk) that are slower than RAM memory to allocate less used pages. When a page allocated in the hard disk is required, a swap is done between a RAM page (frequently, less used of them) and the hard disk page.

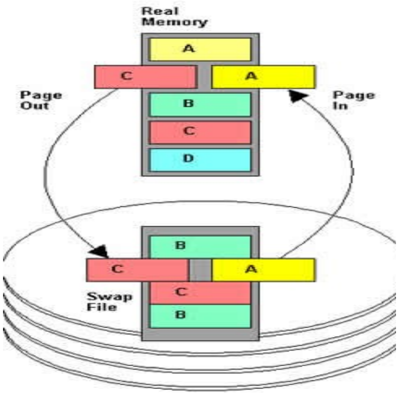


Figure. Swapping 2 pages.

〃 A very important thing to remember is that a process only can be executed if it is located in memory RAM. Never in hard disk.

〃 For performance reasons when we request a information that is in hard disk, it is swapped with less used page of memory and then the program can read it. The information is never accessed directly from hard disk as if hard disk was an extension of RAM memory.

8. INPUT / OUTPUT MANAGEMENT

8.1 Ways to manage I/O

Mainly, there are 3 ways to manage I/O:

- **Programmed I/O:** each user process has to check (using CPU time) if a I/O operation is performed. When an I/O operation is performed, it reads results. This way only works for mono-task systems and is very inefficient.
- **Interruptions:** operating system detects when a I/O operation is done and notifies it to the running process and only uses CPU time reading the result. When a process is waiting for I/O isn't using CPU time (Is in state “Waiting”).
- **DMA (Direct Memory Access):** It is a technique usually combined with interruptions. For several I/O operations (like hard disk to RAM operations) CPU doesn't do the operation, it is done by other chips, reducing the CPU workload.

8.2 Techniques to increment I/O performance

Usually, I/O devices are slower than CPU. To reduce this problem, we usually use caches. For example, if a operating system wants to write a data in a hard disk, it doesn't wait to hard disk to finish the operation. The operating system writes the information in the cache and when it is done, computer thinks “data has been written”, but it isn't really written since the hardware component has done the process.

This performance technique is one of the reasons to avoid extract USB device or to shutdown your computer badly. If you see in your screen that a I/O operation is done, maybe operating system “thinks” it's done, buy not really yet.

9. FILE SYSTEM MANAGEMENT

9.1 File system structure

A file system is the Operating System component whose job is decide how to order information in a device (usually a hard disk). In a external way, the most used file systems usually use inverted tree structure.

Mainly the file system have two kinds of objects: files and directories.

- **File**: is an object that stores information.
- **Directory**: is a special file that only can include references to other files or directories. The main directory is usually called root directory, and it is the root of the "inverted tree".

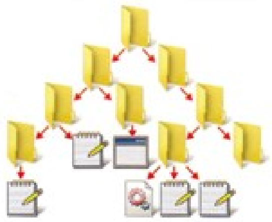


Figure. Inverted tree structure sample with files and directories.

9.2 File system attributes

A file system has those attributes:

- Maximum partition size: maximum length of a partition.
- Maximum file size: maximum length of a file.
- Cluster size: usually configurable. Is the minimum space unit that a hard disk can use. If you store a file with less size than cluster or the last cluster used by a file is not full, that space is wasted[1].
  - If you have a big cluster size, you obtain the best performance, but you waste disk space. Nowadays, hard disk are massive, and it is a good idea to have a big cluster size.
  - If you have a small cluster size, you obtain worse performance, but you gain disk space.

1 In this webpages you can see cluster sizes for Microsoft file systems https://support.microsoft.com/es-es/kb/140365

> When you save a file in the hard disk, the file will be divided into clusters that do not have to be consecutive. For example, a 10kB file in a hard disk with a 4kB sector size and with two sectors per cluster, will need 3 clusters (the last one almost empty). Maybe, each one of those clusters are in a different cylinder or, even, in a different platter. So to load or save it you need three (slow) access to hard disk (you lose performance)

In general, internally, to order the files, all the file systems have a table (or something similar) to know where are each one of the parts of the file. This table is as a directory to find the files.

There are a lot of file systems (ext3, ext4, NTFS, FAT32, FAT16, HPFS, ...). Even many Operating System can use several of them. But in a hard disk partition you only can use one of them. Each one has different characteristics: maximum file size, security against failures, maximum number of nested directories, maximum characters in file names, etc.

> A partition is a logical division of the hard disk. It is very useful to separate information and to prevent future errors. In fact, you can format a partition without affecting the rest of the disk.

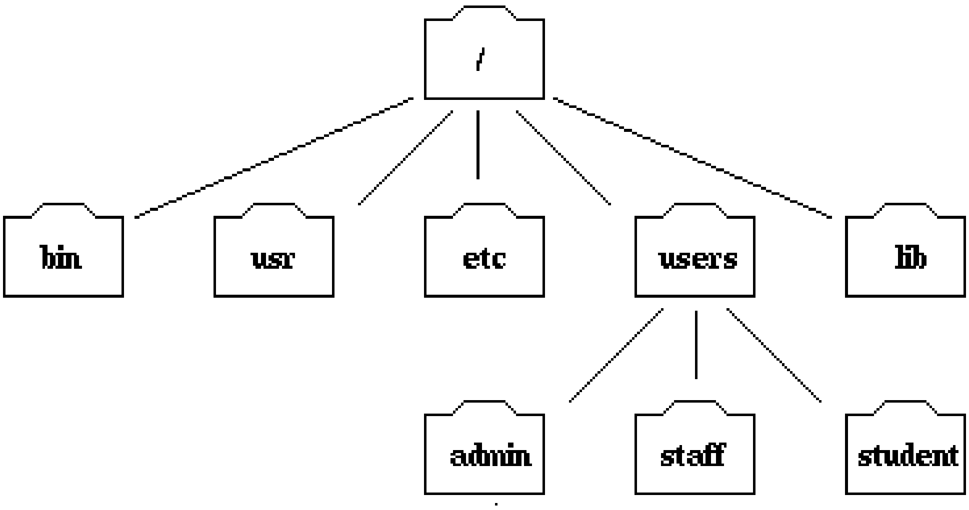
> Format a disk is not to erase it. A format, among other things, organizes from scratch the file system table. Erasing it is a consequence of this organization.

> If your operating system does not recognize a hard disk file system, it can not save or read information on it. For instance, there are a lot of TVs that only support FAT32. FAT32 only supports files smaller than 2 GB, so if you want to watch a film in HD, maybe you need tu use a most sophisticated file system like NTFS. But, although you can save the film in the hard disk (because the computer recognizes NFTS, you do not watch it in you TV.

9.3 Absolute and relative paths

When we go through a file system, and we want to refer to a directory or a file, we can use two kinds of paths: relative and absolute.

- Absolute path: it includes all the path starting from the root.
- Relative path: it depends on what position are we.
    - Special symbols in relative paths:
        - . → 1 single point reference to current directory.
        - .. → 2 points reference to parent directory
    - Sample:



- An absolute path to access to admin is "/users/admin".
- If you are in */bin*, a relative path to access to admin is *../users/admin*
- If you are in /, a relative path to access to admin is *users/admin* or *./users/admin*.

9.4 Most popular file systems features[2]

	FAT16	FAT32	NTFS	ext4
Operating system	MS-DOS 6.22, W9X	Windows 9X, Windows Server, Windows XP/7/10	Windows Server, Windows XP/7/10	Linux
Max file size	2 GiB	4 GiB	Limited by volume size	16 GiB to 16 TiB
Max partition size	2 GiB	2 TiB	Limited by volume size	1EiB

10. HOW TO RESOLVE A PROCESS SCHEDULING PROBLEMS

The problems of scheduling algorithms are solved just as a computer does: creating the algorithm and following it.

In general, create these algorithms is very easy, but the Round Robin algorithm is a little more complicated.

1º) The first queued process is executed

2º) Any process want to enter? If it enters, it is queued (if the queue is empty does not run, it is only queued)

3º) The executed process finished the quantum, but it has not finished all its cycles? Yes, then put it at the end of the queue.

11. ADDITIONAL MATERIAL

[1] Operating systems tutorial (advance)

http://www.tutorialspoint.com/operating_system/

[2] Solucionador de problemas de planificación de procesoss

http://uhurulabs.com/PlanificadoresCPU/

2 More information at: https://en.wikipedia.org/wiki/Comparison_of_file_systems

12. BIBLIOGRAPHY

[1] Operating systems
https://en.wikipedia.org/wiki/Operating_system

[2] Process management
https://en.wikipedia.org/wiki/Process_management

[3] Memory management
https://en.wikipedia.org/wiki/Memory_management

[4] Device management
https://es.wikipedia.org/wiki/Device_Management

[5] File system
https://en.wikipedia.org/wiki/File_system

[6] Comparison of file systems
https://en.wikipedia.org/wiki/Comparison_of_file_systems