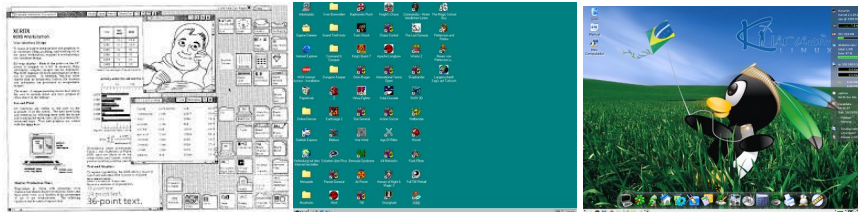




Operating Systems (OS) or:

The thing that makes your computer really work (or not)

David Beserra _/_/_/



Windows

An error has occurred. To continue:

Press Enter to return to Windows, or

Press CTRL+ALT+DEL to restart your computer. If you do this,
you will lose any unsaved information in all open applications.

Error: 0E : 016F : BFF9B3D4

Press any key to continue _

+ Kernel Mode x User Mode

- Most computers have two modes of operation: kernel mode and user mode.
 - In kernel mode, all hardware resources are accessible, whereas in user mode, only certain functionalities are available.
- The operating system is the most fundamental software of a computer.
 - DEF: The operating system runs in kernel mode.
- User applications and the operating system interface run in user mode.

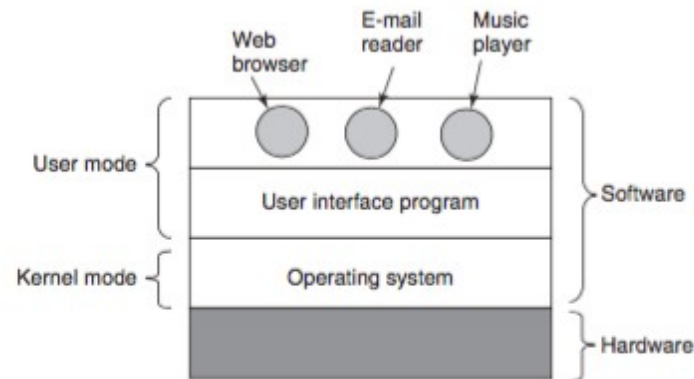


Figure 1-1. Where the operating system fits in.



+ What is the purpose of an operating system?

4

■ Operating systems play two fundamental roles:

- Extending the machine and providing abstractions to user programs.
- Managing the utilization of available resources.

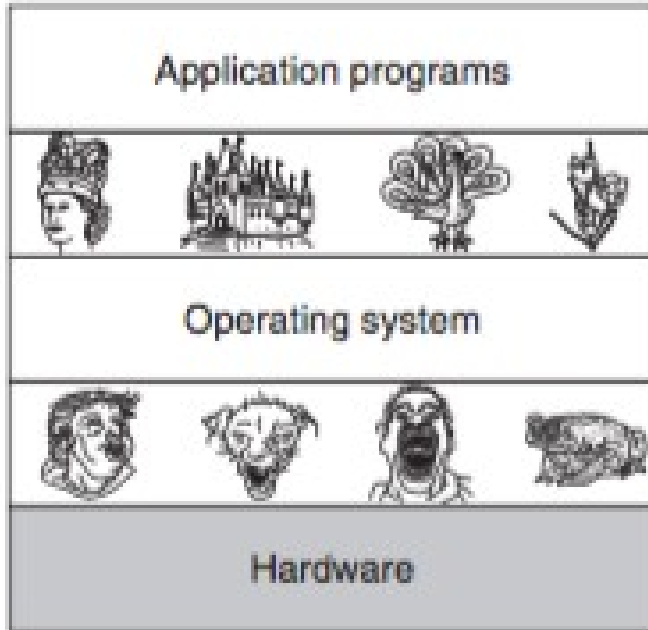


+ What is the purpose of an operating system?

- The operating system as an extended machine:
 - The operating system provides users with an abstraction that hides the complexities of hardware.
 - Example: It is simpler for a programmer to think in terms of files and directories (abstractions that hide complexity) than in terms of sectors, disk rotation speed, etc. (hardware complexity).

+ What is the purpose of an operating system?

- The operating system as an extended machine:
 - Hiding complexities. Hardware interfaces are "ugly," while the abstractions provided by the OS are "pretty. »



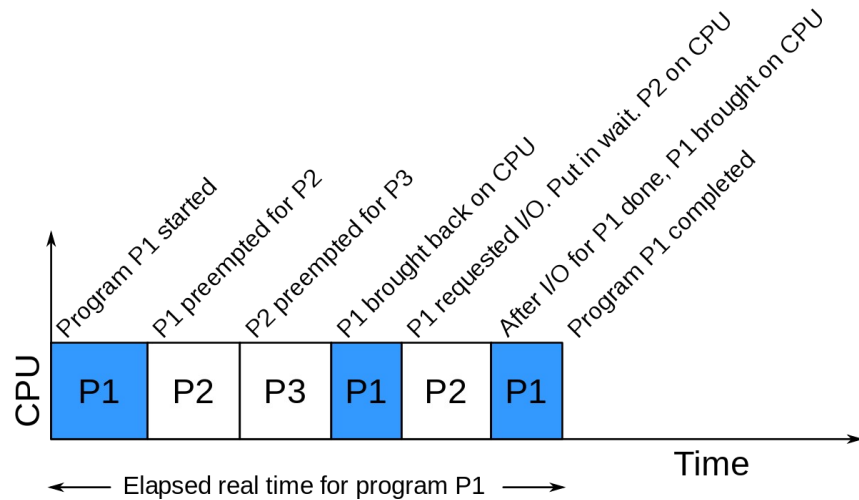
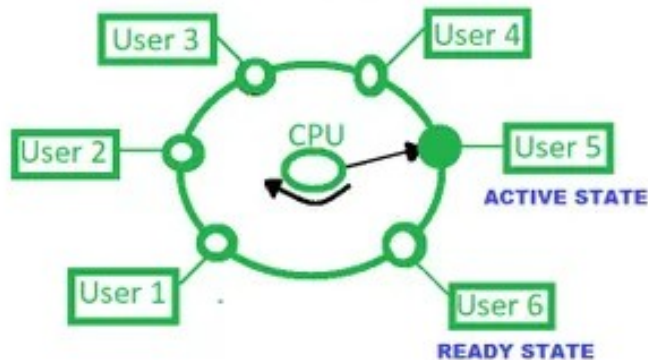
+ What is the purpose of an operating system?

- **The operating system as a resource manager:**
- Modern operating systems allow the simultaneous execution of multiple programs by multiple users. Therefore, hardware resources are **shared** among these multiple programs.
- The operating system is responsible for deciding when a program can use a particular resource.

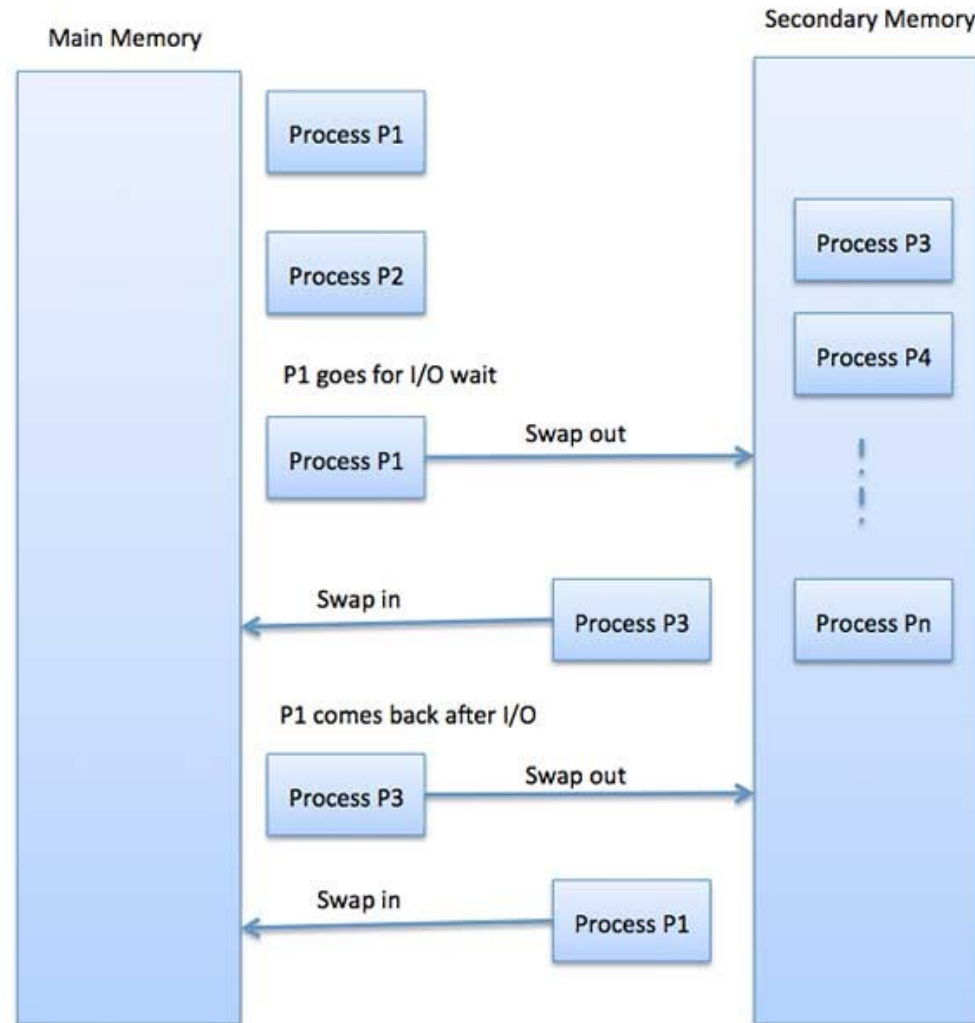
+ What is the purpose of an operating system?

- There are two different ways to share a resource.
 - Time-sharing: Each program uses a resource for a certain period of time.
 - Example: Two programs using the same CPU.
 - Space-sharing: Each program uses a "portion" of a resource.
 - Example: Two programs using a part of RAM or the hard disk..

Time Shared Operating System



+ What is the purpose of an operating system?

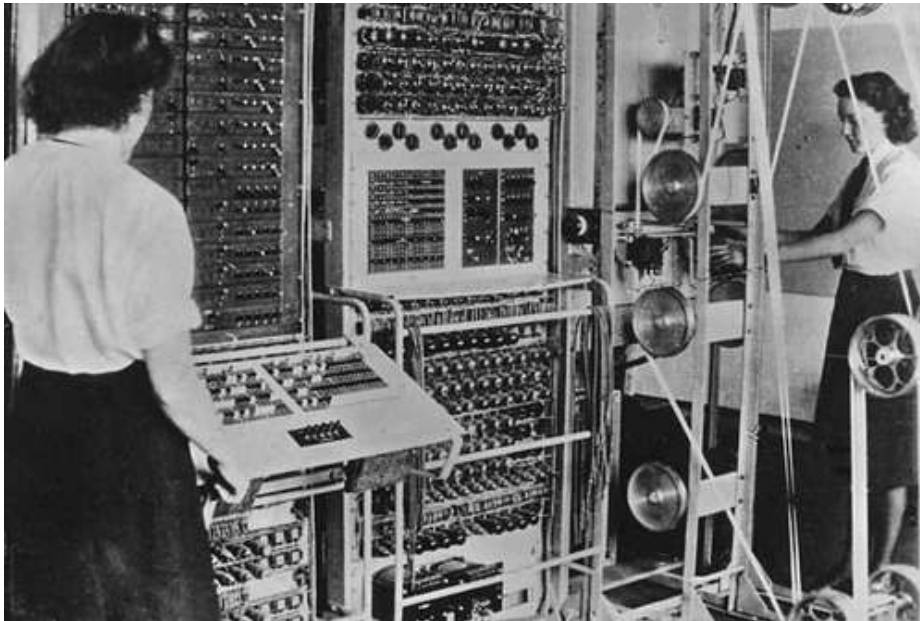


+ The History of the Operating Systems

- The history of operating systems closely mirrors the evolution of computers in general.
- The first generation (1945-1955): Vacuum tubes.
- The second generation (1955-1965): Transistors and batch processing systems.
- The third generation (1965-1980): Integrated circuits and multiprogramming.
- The fourth generation (1980 - Present): Personal computers.
- The fifth generation (from 1990 to the present): Mobile computers.

+ The History of the Operating Systems

- The first generation (1945-1955): Vacuum tubes.
 - Programming directly on the hardware. Therefore, there was no operating system for these computers.



+ The History of the Operating Systems

- The second generation (1955-1965): Transistors and batch processing systems.
 - Operating systems were designed to coordinate the execution of batches of jobs.

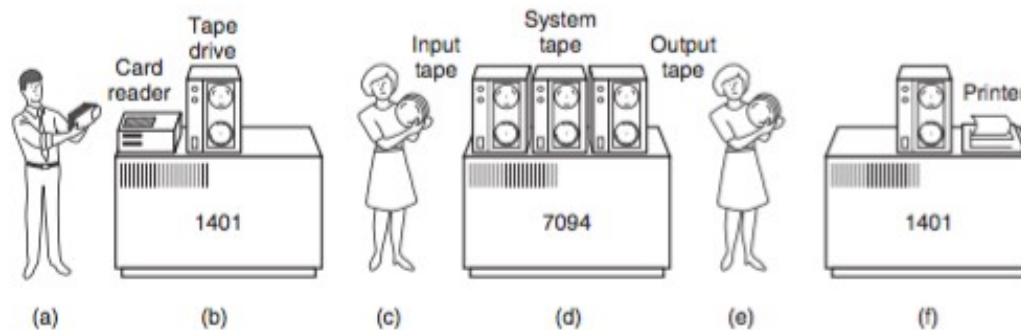
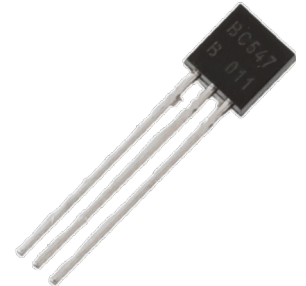


Figure 1-3. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.



+ The History of the Operating Systems

- The second generation (1955-1965): Transistors and batch processing systems.

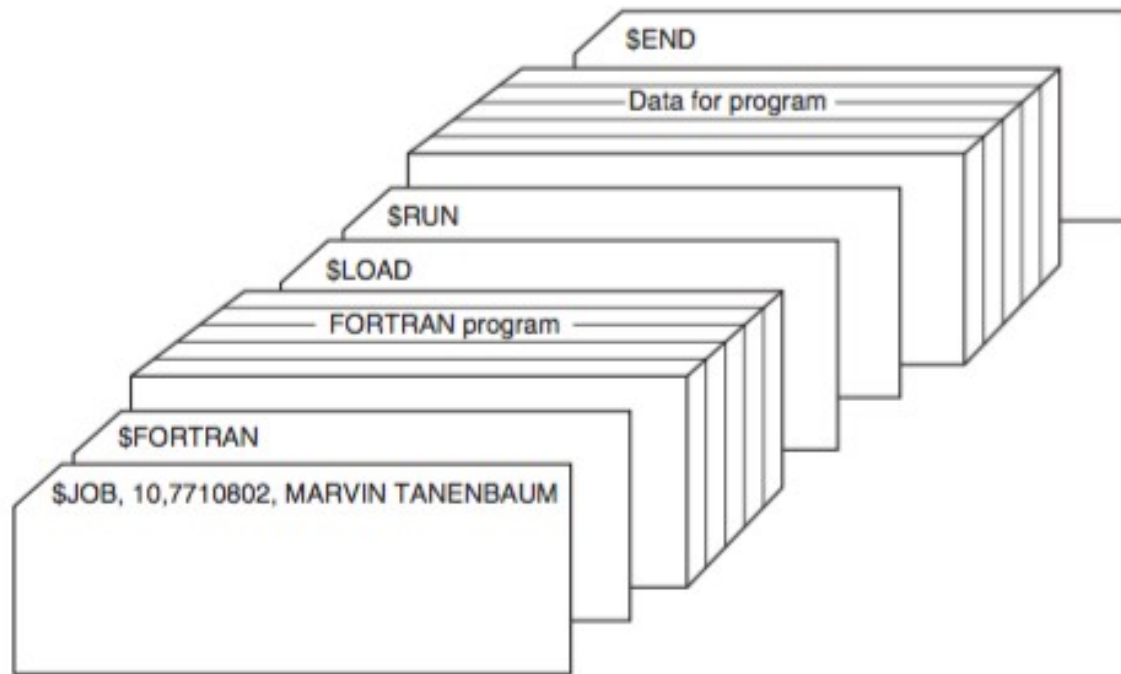


Figure 1-4. Structure of a typical FMS job.

+ The History of the Operating Systems

- The third generation (1965-1980): Integrated circuits and multiprogramming.
 - Common problem in the second generation: time wastage
 - While performing I/O, the CPU remains unused.
 - Solution: Multiprogramming.
 - Memory is divided into multiple slices, one slice for each task in execution.

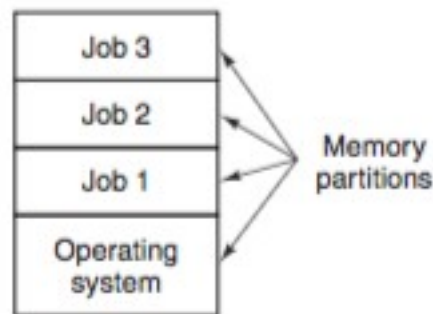


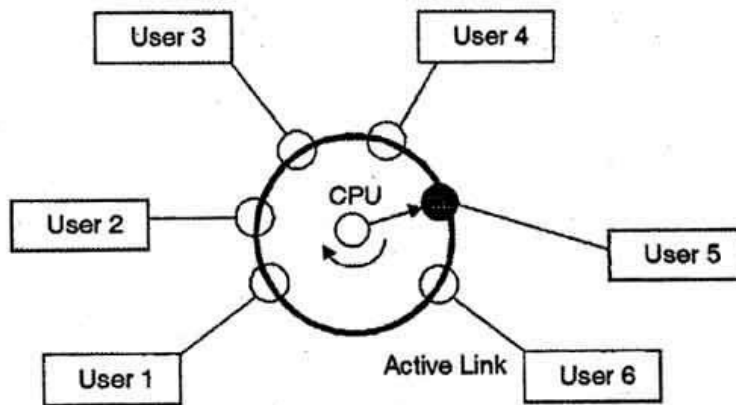
Figure 1-5. A multiprogramming system with three jobs in memory.



+ The History of the Operating Systems

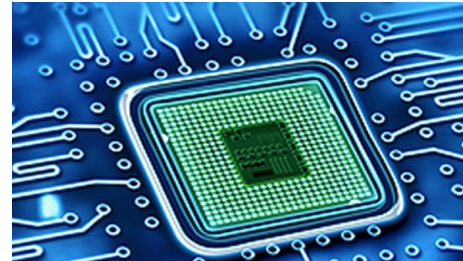
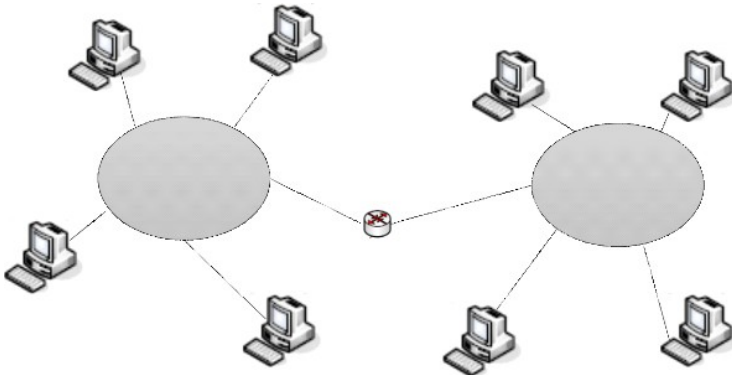
■ Timesharing

- A variant of multiprogramming in which each user has an online terminal.
- In a timesharing system, if 20 users are connected and 17 of them are idle, chatting, or having coffee, the CPU can be allocated in turn to the three tasks that need to be served.



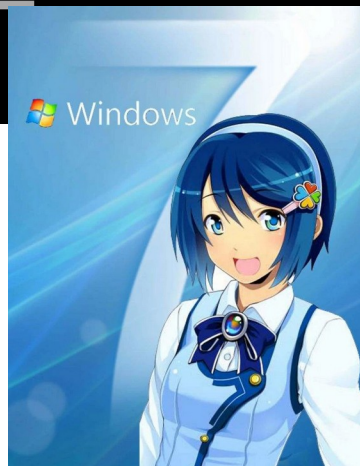
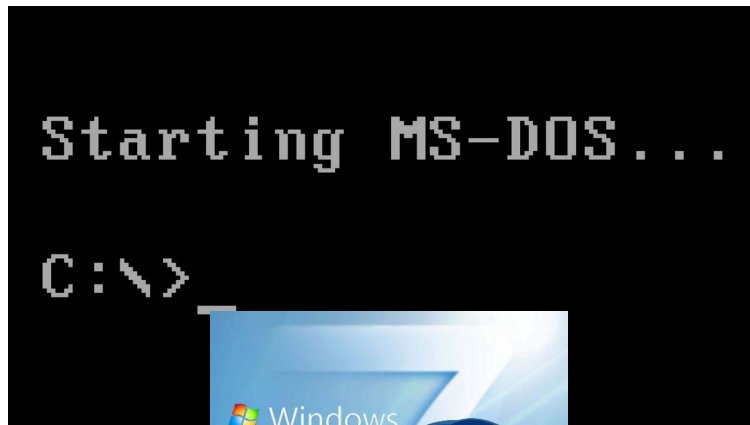
+ The History of the Operating Systems

- The fourth generation (1980 - Present):
 - Personal computers Operating systems oriented towards a connected world Network operating systems



+ The History of the Operating Systems

- La quatrième génération (1980 – Aujourd'hui): ordinateurs personnels



+ The History of the Operating Systems

- The fifth generation (from 1990 to the present): Mobile computers



old.organizers.com



+ Operating Systems's ZOO

- For each type of computer, there is a different operating system.
 - Mainframe operating systems
 - Server operating systems
 - Multiprocessor operating systems
 - Personal computer operating systems
 - Pocket computer operating systems
 - Embedded operating systems
 - Sensor node operating systems
 - Real-time operating systems

+ Operating Systems's ZOO

- Different kinds of application needs different OSes :)





Booting the Computer

- **BIOS (Basic Input/Output System)**

- The BIOS, or Basic Input/Output System, is a small program stored in non-volatile flash memory installed on the computer's motherboard. It is responsible for assisting in the startup and the operating system (OS) loading process.

- **Startup Process:** When the computer is powered on, the BIOS begins executing a series of tasks:

- **Checking Installed Memory:** It checks how much memory (RAM) is installed in the computer.

- **Keyboard Functionality:** The BIOS verifies whether the keyboard is functional.

- **Detecting Connected Devices:** It scans all the buses on the motherboard to detect connected devices such as hard drives, optical drives, USB devices, etc.

- **Device Configuration:** If a new device is found, the BIOS configures it so that the system can recognize and use it.

- **Loading the OS:** The BIOS proceeds to load the operating system into memory from a storage device, typically in the following order of priority: floppy disk (in older PCs), CD-ROM, hard disk drive (HDD), or solid-state drive (SSD).

+ Operating System Concepts

- Basic Concepts in an Operating System
 - **Processes**
 - Address Spaces
 - Files Input / Output
 - Security
 - The Shell (Command Interpreter)

+ Process

- A process is a running program.
- A process contains:
 - A **PID** (process identifier), which is a number assigned to the process by the OS to distinguish it from other processes.
 - The **UID** (user identifier) of the user who created the process.
 - And the **GID** (group identifier) of that user.
 - **An address space:** a list of memory addresses that a process can use, ranging from 0 to a maximum number (MAX).
 - **The executable program.**
 - **Its data.**
 - **An execution stack.**
 - **Resources** associated with the process, e.g., registers in the processor.

+ Process

- A process can generate new processes, called **child processes**. These child processes can, in turn, generate their own child processes..
- The parental relationships between processes can be represented as a tree-like structure, known as a **process tree**.
- Sometimes, two (or more) distinct processes need to communicate with each other to synchronize their activities. This communication is called **interprocess communication (IPC)**.

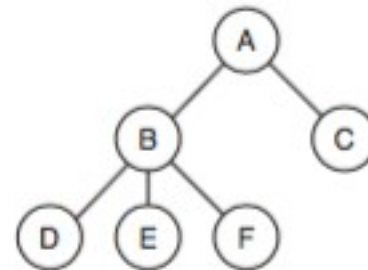


Figure 1-13. A process tree. Process *A* created two child processes, *B* and *C*. Process *B* created three child processes, *D*, *E*, and *F*.

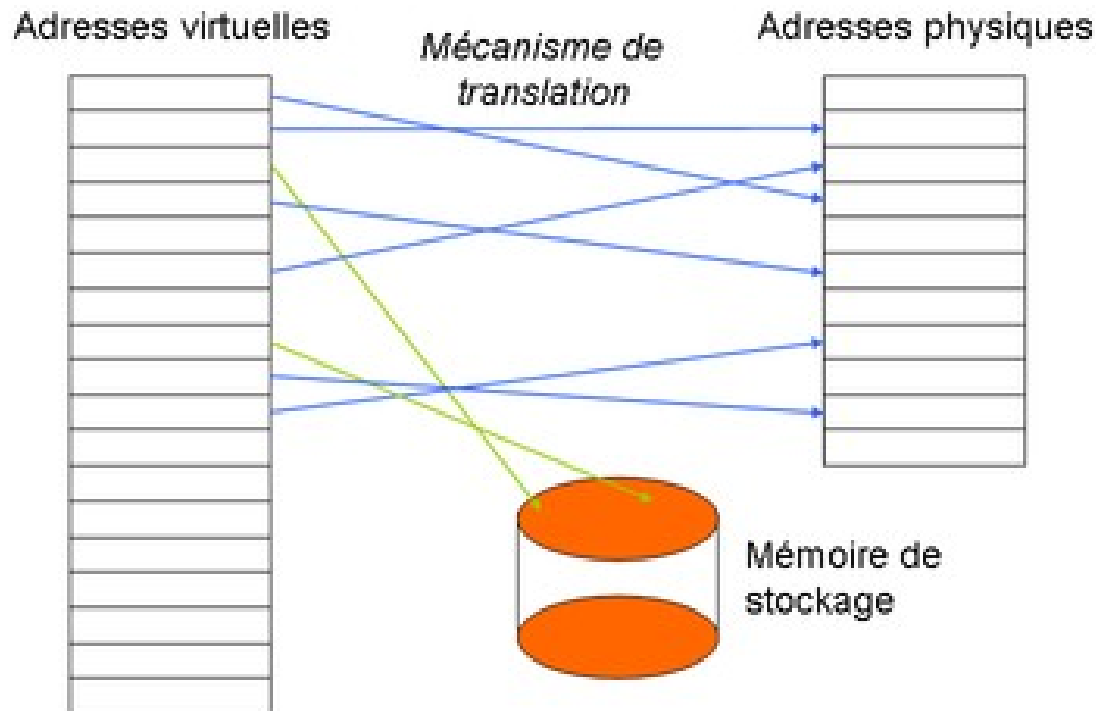
+ Address space management

- In a modern computer, multiple processes can be loaded into memory and executed simultaneously.
- Since each process has its own address space, there are situations where a process's address space exceeds the available memory capacity.
- So, what do we do?
 - We use the concept of **virtual memory**: processes are allowed to maintain a portion of their address space in physical RAM (fast memory) and another portion on disk (much slower storage).
 - When necessary, a process performs swaps: it sends some of its data to addresses on the disk and retrieves data from these locations to place them into physical memory.

+ Address space management

■ Virtual Memory

- The addressing process is decoupled from physical memory.



+File Management

- One of the main functions of operating systems is to hide the complexity of disks and provide developers with a simple, device-independent file model.
- Generally, there are many files on a computer, and it is necessary to organize them in a way that allows access and modification when needed.
- For this purpose, there are file systems that hierarchically organize files stored in the computer's secondary storage devices (HD, CD_ROM, flash drives, etc.).



File Management

- Important concepts related to file systems:
 - **Directory:** A group of files grouped together.
 - **Path name:** The path to access a file in the file system.
 - **Root directory:** This is the directory at the top of the directory tree. All other directories are contained within the root directory.
 - **Working directory:** It is the current working directory of a process.
 - **Mount:** The process of integrating external file systems into the operating system's file system (for example, the file system of a CD).

+File Management

■ Exemple of directory tree

Ex: path to access the file « estoy-aqui » :

/Faculty/Prof. Brown/Commitees/COST-11/estoy-aqui

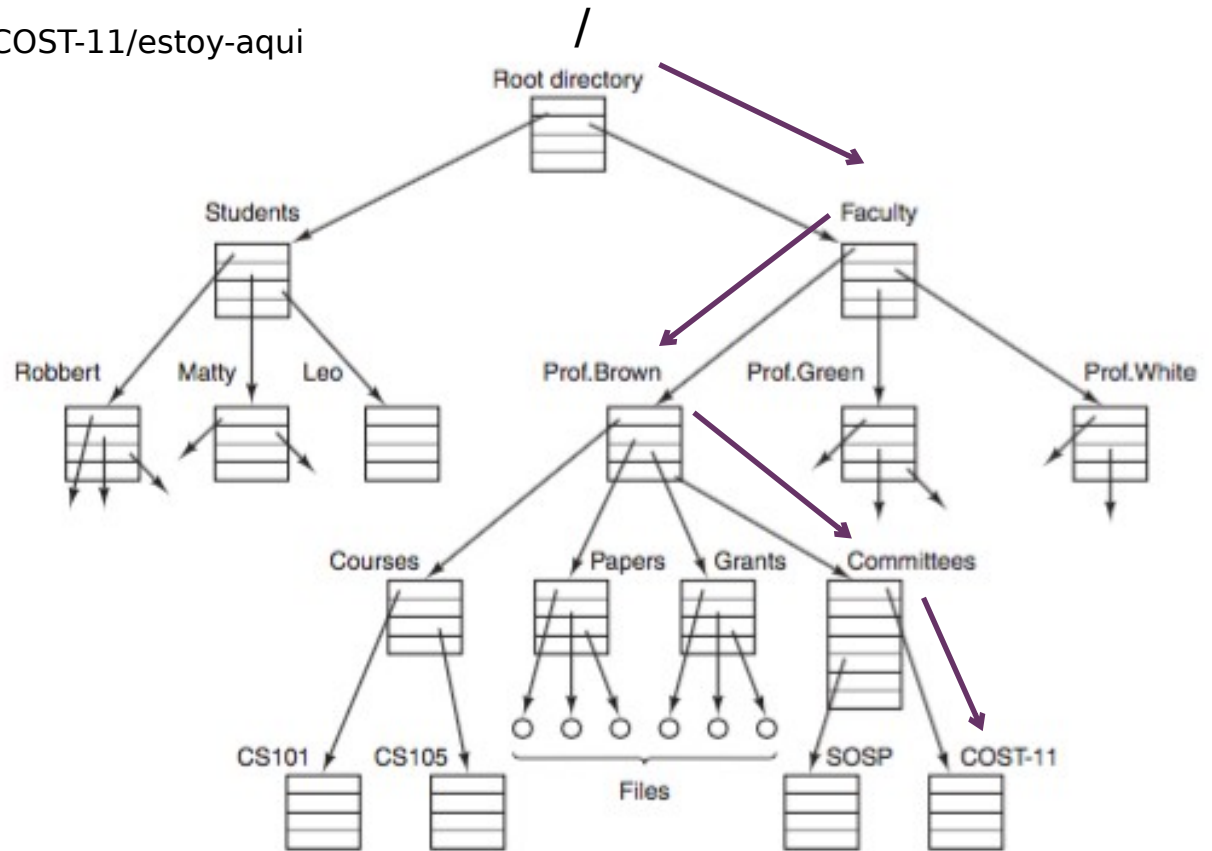


Figure 1-14. A file system for a university department.

estoy-aqui



File Management

- Montage de arbre de répertoires □

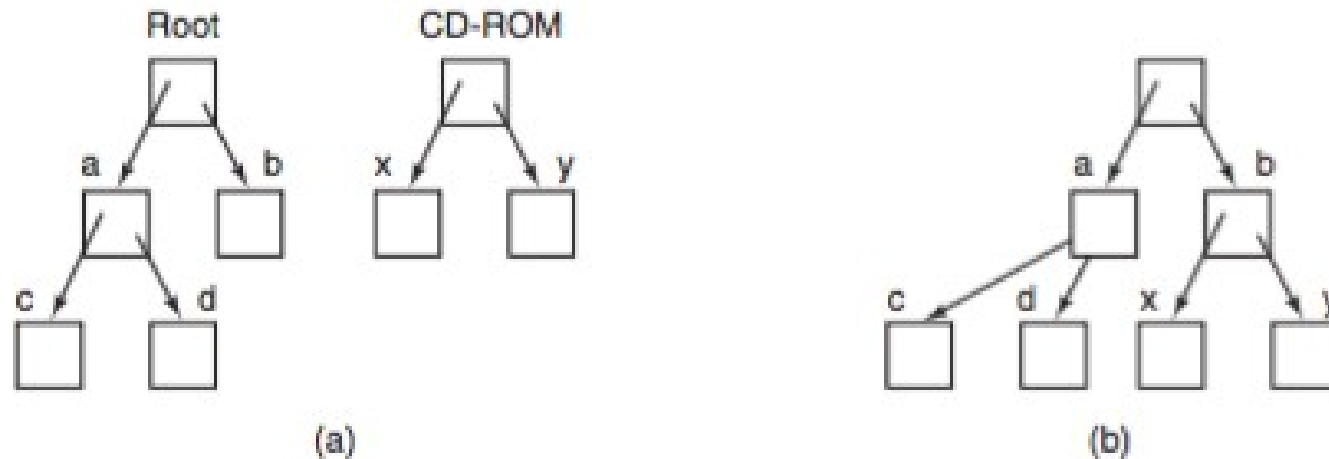


Figure 1-15. (a) Before mounting, the files on the CD-ROM are not accessible. (b) After mounting, they are part of the file hierarchy.

+ File Security

- Each file has 9 bits, grouped in sets of 3, to indicate the access privileges granted to:
 - The file owner
 - Other users
 - Other members of the file owner's user group:

■ rwx bits

user group others

rwxrwxrwx

e r e e r e e r e
a i c a i c a i c
d t u d t u d t u
e e e e e e e e e

user group others

rwxrw-r-- = symbolic

111110100 = 3-bit

7 6 4 = octal

```
ec2-user@ip-10-164-110-7 uc_social_server]$ ls -l
total 120
drwxr-xr-x 8 ec2-user ec2-user 4096 Jul 27 19:28 app
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 27 19:28 bin
drwxr-xr-x 5 ec2-user ec2-user 4096 Jul 27 19:28 config
-rw-r--r-- 1 ec2-user ec2-user 154 Jul 27 20:47 config.ru
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 27 19:28 db
-rw-r--r-- 1 ec2-user ec2-user 1455 Jul 27 20:47 Gemfile
-rw-r--r-- 1 ec2-user ec2-user 3706 Jul 27 20:47 Gemfile.lock
drwxr-xr-x 4 ec2-user ec2-user 4096 Jul 27 19:28 lib
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 27 20:52 log
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 27 20:48 out.txt
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 27 19:29 public
-rw-r--r-- 1 ec2-user ec2-user 257 Jul 27 20:48 Rakefile
-rw-r--r-- 1 ec2-user ec2-user 48780 Jul 27 20:48 README.md
-rw-r--r-- 1 ec2-user ec2-user 478 Jul 27 20:48 README.rdoc
drwxrwxr-x 2 root root 4096 Jul 29 22:42 releases
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 27 19:29 script
drwxr-xr-x 5 root root 4096 Jul 29 22:42 shared
drwxr-xr-x 8 ec2-user ec2-user 4096 Jul 27 19:29 test
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 27 20:48 testfile.yml
drwxr-xr-x 6 ec2-user ec2-user 4096 Jul 27 19:31 tmp
drwxr-xr-x 3 ec2-user ec2-user 4096 Jul 27 19:31 vendor
```

+ Command Interpreter (Shell)

- Interprets OS commands typed by the user.
 - **It is not** part of the OS itself but is very important.
 - Interfaces with the user.

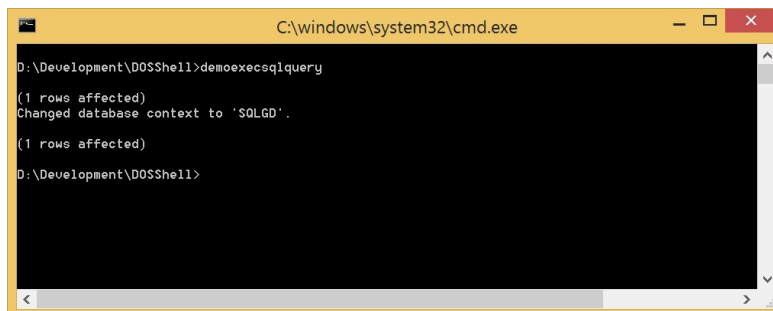


```

(C:\)yaos
[OS/2 Yet Another OS/2 Shell] version 1.9.0 (FREEMWARE).
Original author: LEE, C)in Pheow (C) 1997
Current author: Dave Saville (C) 2011
               davedeezee.org

Usage: yaos [options ...]
Options:
  -b Append invoked program object name to titlebar inside []
  -c <command>: execute a command and then quit
  -o: overwrite mode as default
  -q: quiet mode: no opening message
  -s <filename>: read file containing alias and environment settings
  -t: toggle file names to complete (-w option is ignored)
  -w: popup window for file name completion (-t option is ignored)
  -x <command>: execute command on startup
  -y: auto ycd when normal cd fails

Type: '?' at prompt for help
(C:\)
  
```



```

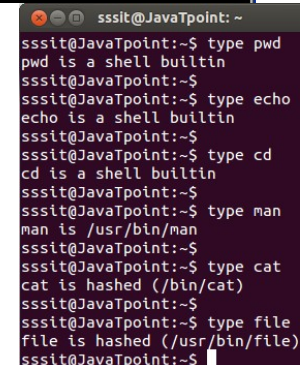
C:\windows\system32\cmd.exe

D:\Development\D0SShell>demoexecsqlquery

(1 rows affected)
Changed database context to 'SQLGD'.

(1 rows affected)

D:\Development\D0SShell>
  
```



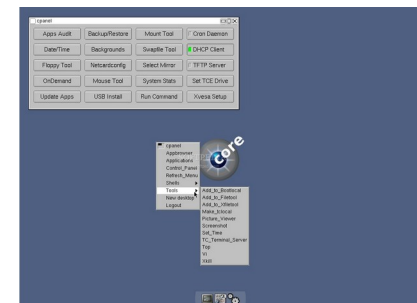
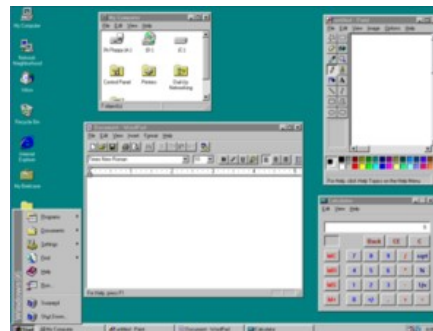
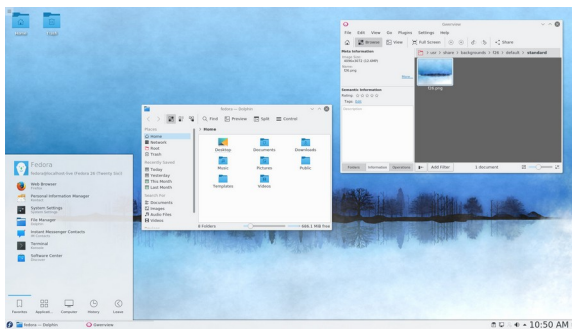
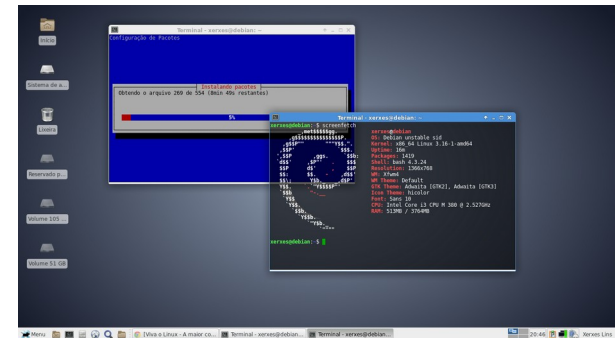
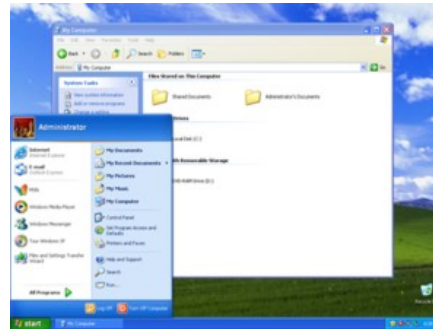
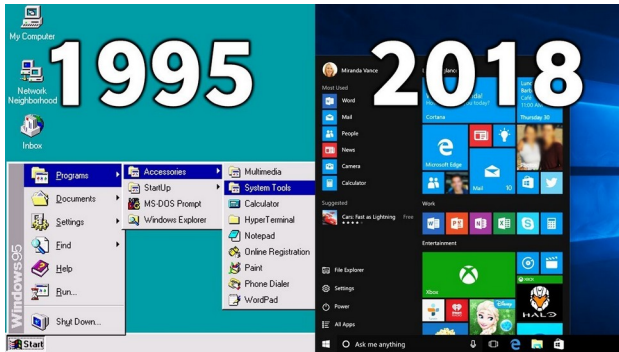
```

sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ type pwd
pwd is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type echo
echo is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type cd
cd is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type man
man is /usr/bin/man
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type cat
cat is hashed (/bin/cat)
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type file
file is hashed (/usr/bin/file)
sssit@JavaTpoint:~$
  
```


+ GUI

33

- User Graphical Interface
 - For those who don't know how to use a shell xD



+ System Calls

- What is this?
 - These are the services that the OS offers to its users.
- Most common types of System Calls:
 - System calls for process management
 - System calls for file management
 - System calls for directory management

+ System calls : process management

System calls to manage processes

Process management	
Call	Description
<code>pid = fork()</code>	Create a child process identical to the parent
<code>pid = waitpid(pid, &statloc, options)</code>	Wait for a child to terminate
<code>s = execve(name, argv, environp)</code>	Replace a process' core image
<code>exit(status)</code>	Terminate process execution and return status

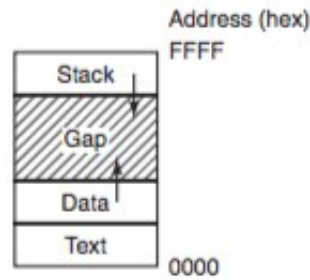


Figure 1-20. Processes have three segments: text, data, and stack.

+ System calls to manage files

- System calls to manage files

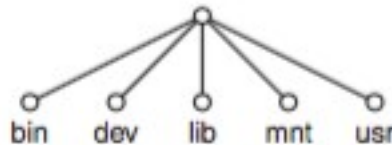
File management

Call	Description
<code>fd = open(file, how, ...)</code>	Open a file for reading, writing, or both
<code>s = close(fd)</code>	Close an open file
<code>n = read(fd, buffer, nbytes)</code>	Read data from a file into a buffer
<code>n = write(fd, buffer, nbytes)</code>	Write data from a buffer into a file
<code>position = lseek(fd, offset, whence)</code>	Move the file pointer
<code>s = stat(name, &buf)</code>	Get a file's status information

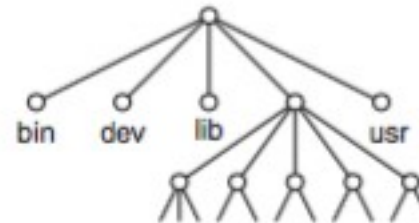
+ System Calls: Directory Management

Directory- and file-system management

Call	Description
<code>s = mkdir(name, mode)</code>	Create a new directory
<code>s = rmdir(name)</code>	Remove an empty directory
<code>s = link(name1, name2)</code>	Create a new entry, name2, pointing to name1
<code>s = unlink(name)</code>	Remove a directory entry
<code>s = mount(special, name, flag)</code>	Mount a file system
<code>s = umount(special)</code>	Unmount a file system



(a)



(b)

Figure 1-22. (a) File system before the mount. (b) File system after the mount.



Operating System Structure

- **Monolithic Systems**
- Layered Systems
- **Microkernels (Microkernel)**
- Client-Server Model
- Virtual Machines
- Exokernels

+ Monolithic Systems

- The entire operating system runs in kernel mode.
- The operating system is written as a set of routines linked to a large executable file.
 - Windows, Linux

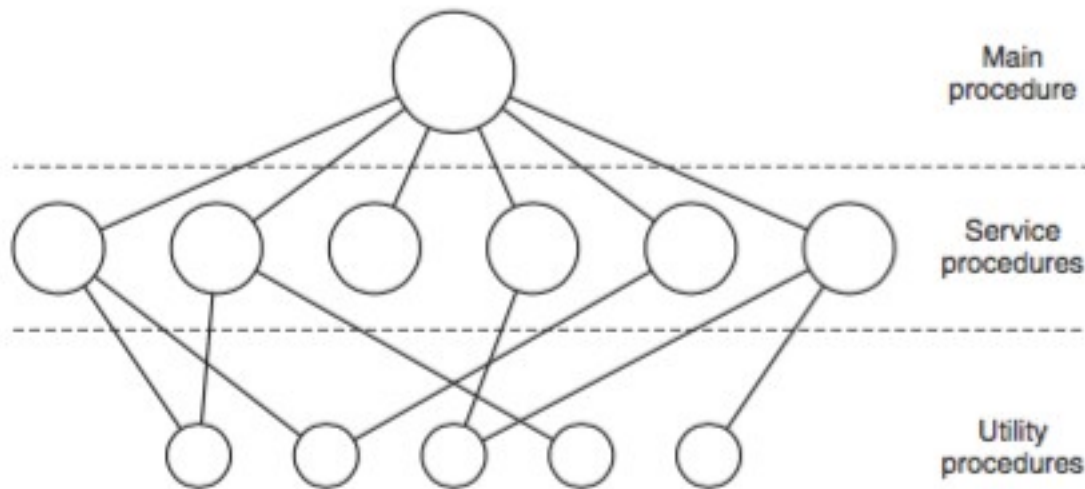


Figure 1-24. A simple structuring model for a monolithic system.



+ Micro-kernel

- The operating system is divided into several small modules, which run in user mode, and only one of these modules, the microkernel, runs in kernel mode.
 - The benefit: the system is more lightweight
- Ex de OS micro-noyaux: Minix 3

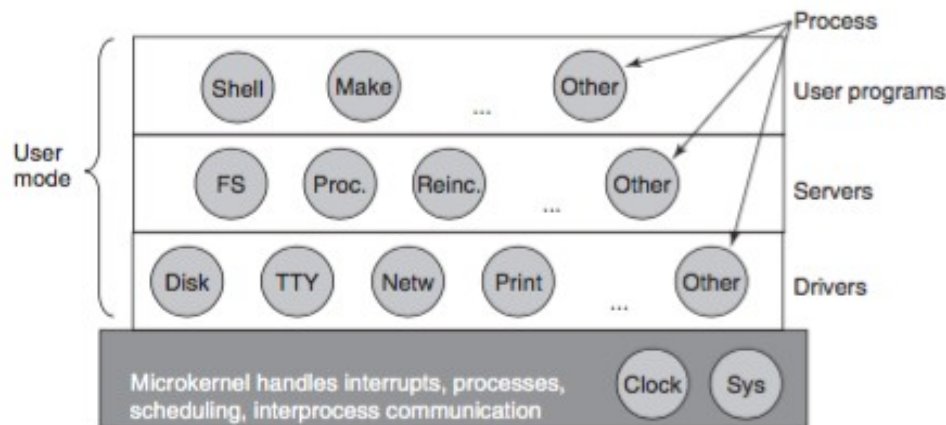
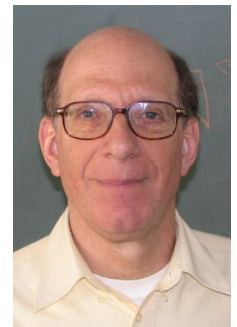
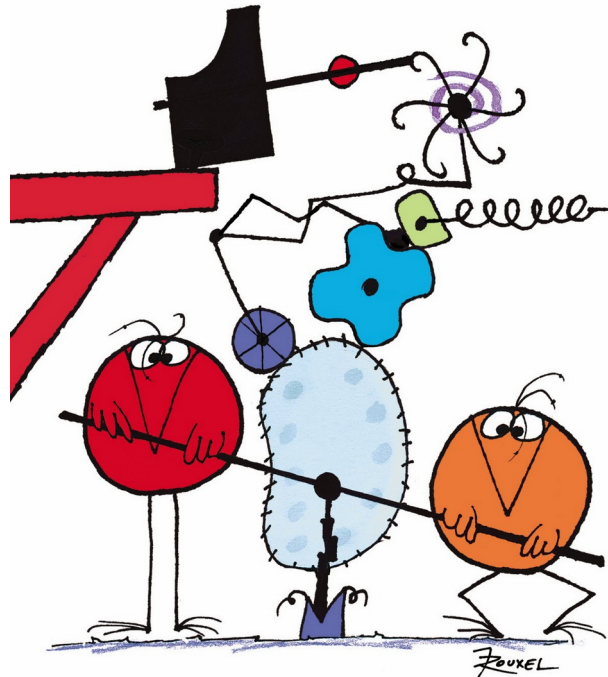


Figure 1-26. Simplified structure of the MINIX system.



+ Références

- TANENBAUM, Andrew S. **Modern Operating Systems**. Pearson Education, Inc, (2009) – Chapter 1 - Introduction



POURQUOI FAIRE SIMPLE QUAND
ON PEUT FAIRE COMPLIQUÉ ?!