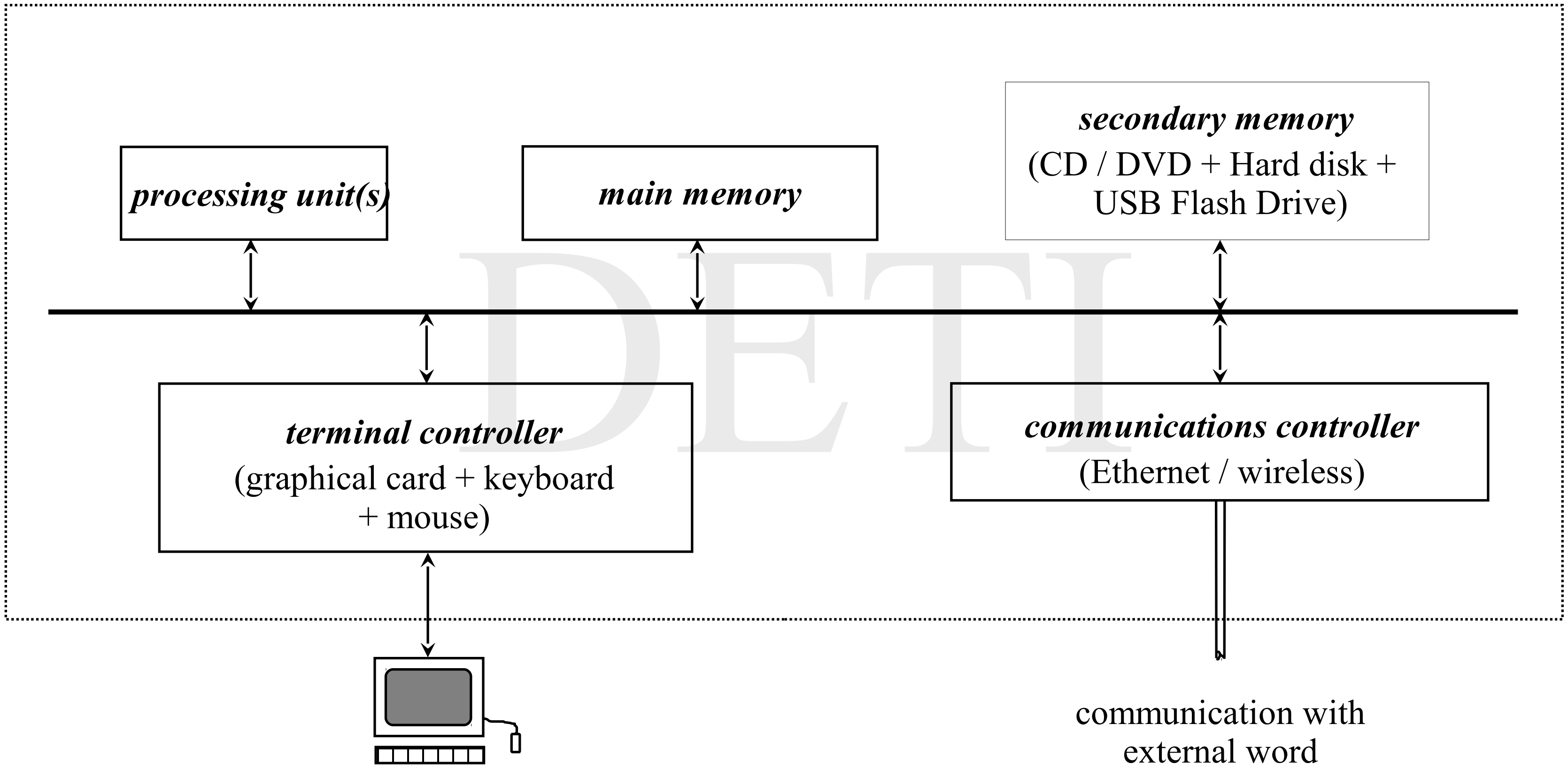
Sistemas de Operação / Operating Systems

Introduction

António Rui Borges / Artur Pereira

Contents

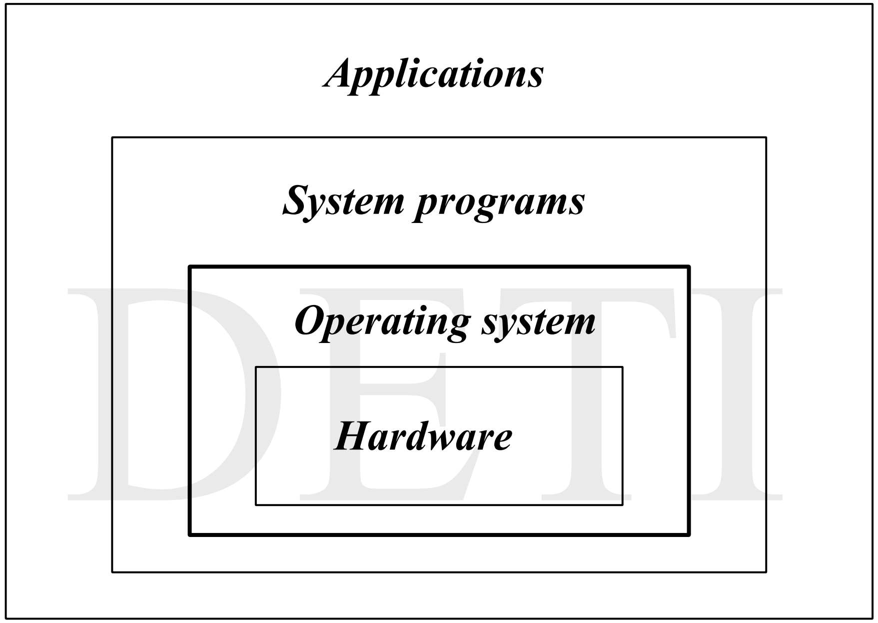
Typical computational system



Operating system: simplified view

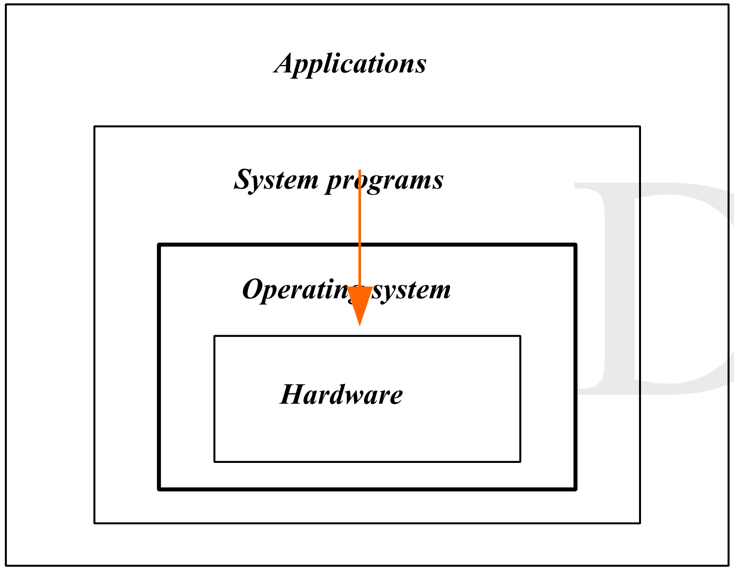
- Support (base) program executed by the computational system
  - gives 'life' to the computer (*hardware*), providing an abstract interaction environment
- Two types are possible:
  - ***graphical*** - based on windows, mainly manipulated via the mouse
    - icons and menus are main interaction elements
    - mouse is the main user input tool
  - ***textual (shell)*** - based on commands introduced via the keyboard
    - a command programming language allows the contruction of complex commands
  - They are not mutually exclusive

Operating system: global view



- Two different perspectives:
  - operating system as an extended machine
  - operating system as a resource manager

Operating system as an extended machine

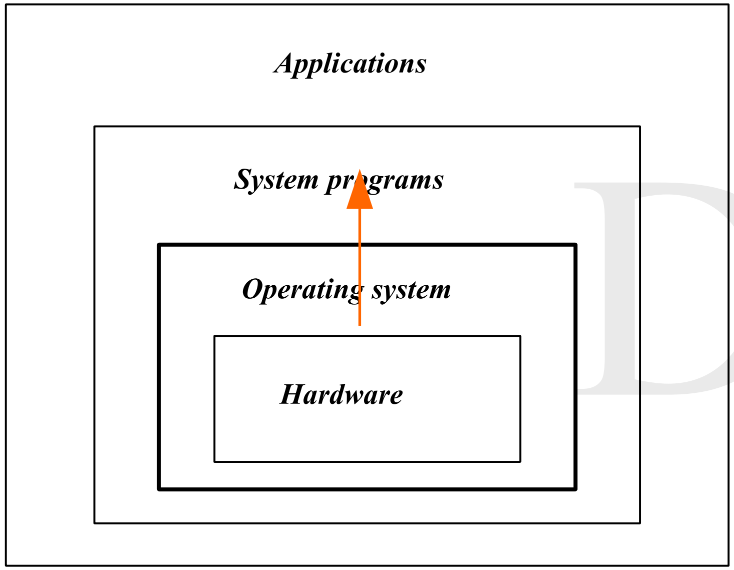


- The operating system provides an abstract view of the underlying computational system that frees programmers of the hardware details
  - A functional model of the computational system, a virtual machine, that is simple to understand and program
  - The interface with the hardware is an uniform programming environment, defined as a set of *system calls*, that allows the portability of applications among structurally different computational systems

Operating system as an extended machine

- Some typical functionalities of the operating system
  - Establishment of a user interaction environment
  - Provision of facilities for the development, testing and validation of programs
  - Provision of mechanisms for the controlled execution of programs, including their intercommunication and synchronization
  - Dissociation of the program's address space from the constraints imposed by the size of the main memory
  - Organization of secondary memory in file systems
  - Definition of a general model for access to input/output devices, regardless of their specific characteristics
  - Detection of error situations and establishment of appropriate response

Operating system as an resource manager



- A computational system is a system composed of a set of resources (processor(s), main memory, secondary memory, I/O device controllers) targeting the processing and storage of information
- The operating system:
  - is the program that manages the computer system, making the controlled and orderly allocation of its different resources to the programs that compete for them
    - resource usage is multiplexed in space and time
  - aims at maximizing the performance of the computational system, ensuring the most efficient use of existing resources

Evolution of operating systems

prehistory

(Mid 19th century)

Charles Babbage

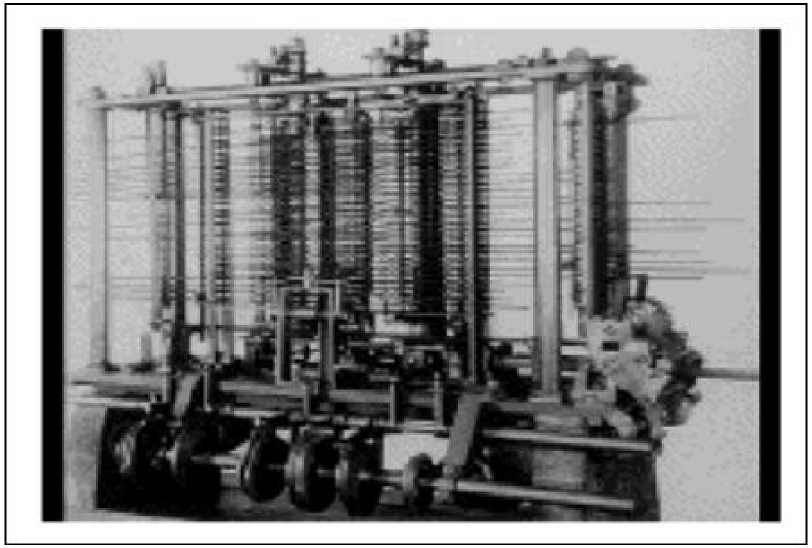
Ada Lovedace

tecnology

- mechanical device

notes

- never work properly
- no operating system



Analytical engine

Charles Babbage

Evolution of operating systems

1st generation

(1945-1955)

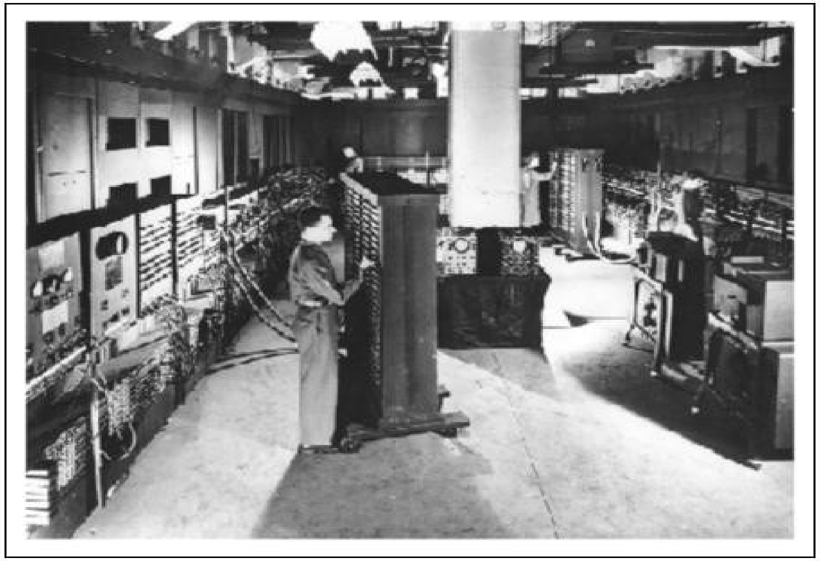
Atanasoft / Berry
Konrad Zuse
Howard Aiken
Eckert / Mauchley

tecnology

- vacuum tubes
- electromechanical relays

notes

- no operating system
- programmed in machine language
- programmer has full control of the machine
- punched cards



ENIAC
Eckert & Mauchley
Univ. of Pennsylvania

Evolution of operating systems

2nd generation

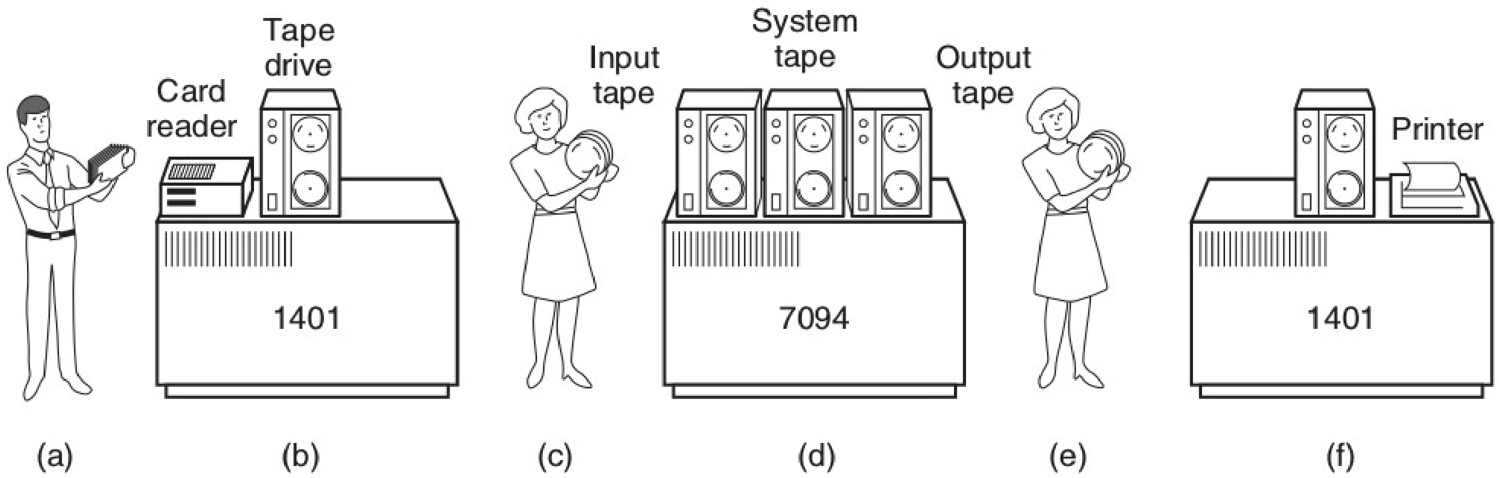
(1955-1965)

tecnology

- transistor devices

notes

- FORTRAN and assembly
- simple batch operating system
- rudimentary command language (job control language)



Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Evolution of operating systems

3rd generation	tecnology	notes
(1965-1980)	• integrated circuits (SSI/MSI) • family of computers (IBM 360) • minicomputers (DEC PDP-n) • microprocessor	• multiprogrammed batch operating system - spooling • interative systems (time-sharing): CTSS (MIT); MULTICS; Unix • (proprietary) general usage operating systems • real-time systems

Evolution of operating systems

4th generation	tecnology	notes
(1980-present)	• LSI/VLSI • personal computers (microcomputers) • network	• standard operating systems (MS-DOS, Macintoch, Windows, Unix) • network operating systems

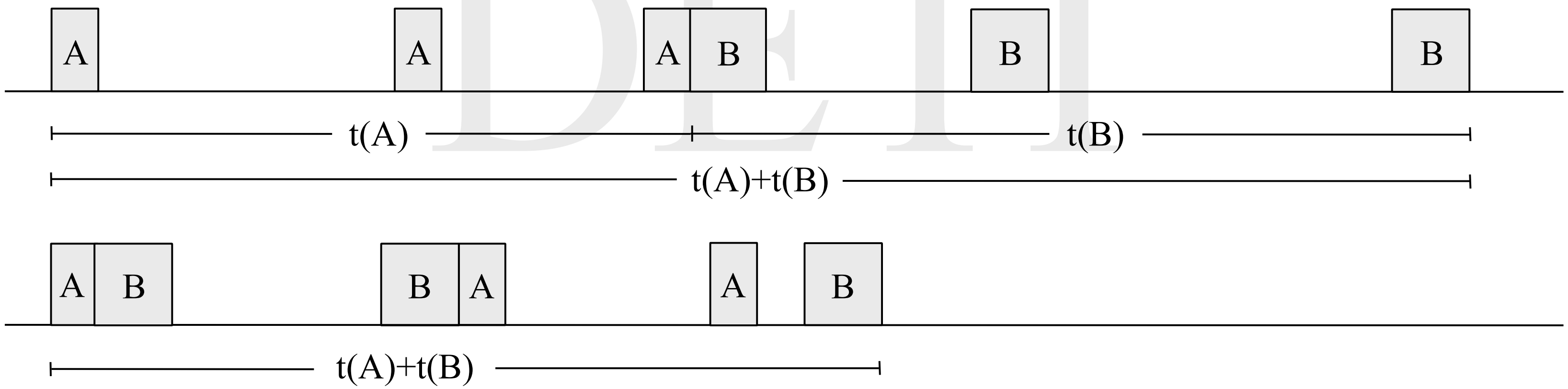
5th generation	tecnology	notes
(1990-present)	• broadband, wireless • system on chip • smartphone	• mobile operating systems (Symbian, iOS, Android) • cloud computing • ubiquitous computing

Taxonomy of operating systems

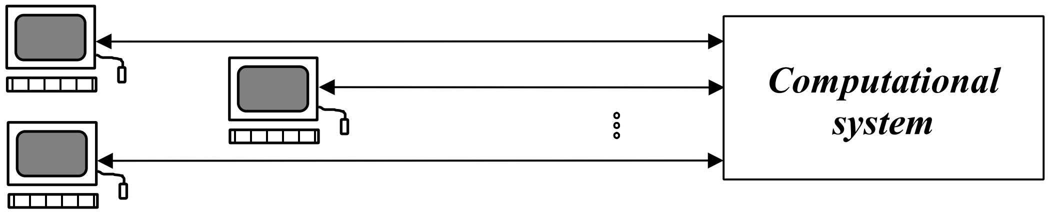
- Operating systems can be classified based on how processing is done:
  - Serial processing
  - (Single) batch processing
  - multiprogrammed batch system
  - time-sharing system
  - real-time system
  - network system
  - distributed system
- Can also be classified based on purpose:
  - Mainframe operating systems
  - Server operating systems
  - Multiprocessor operating systems
  - Personal computer operating systems
  - Real time operating systems
  - Handheld computer operating systems
  - Embedded operating systems
  - Sensor node operating systems
  - Smart card operating systems

Multiprogrammed batch

- Purpose: optimize processor usage
- Method: while a program is waiting for the conclusion of an input/output operation, another program can use the processor

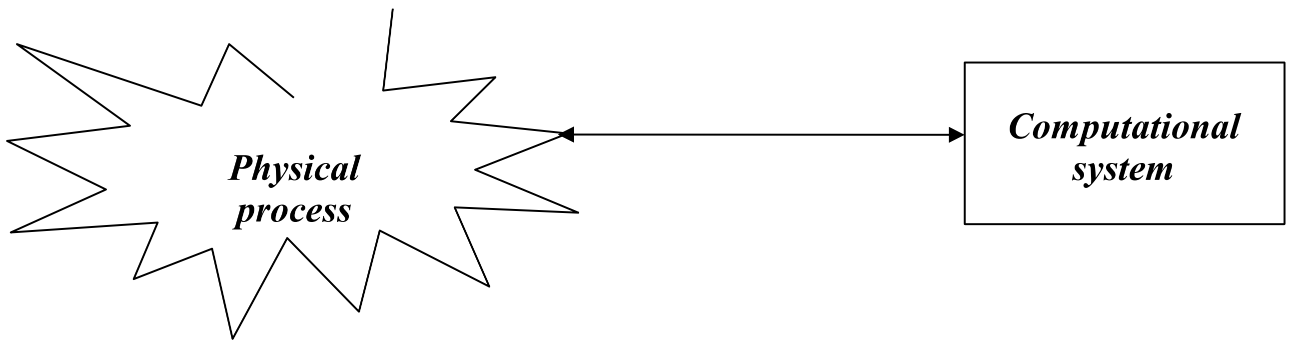


Interactice system (time-sharing)



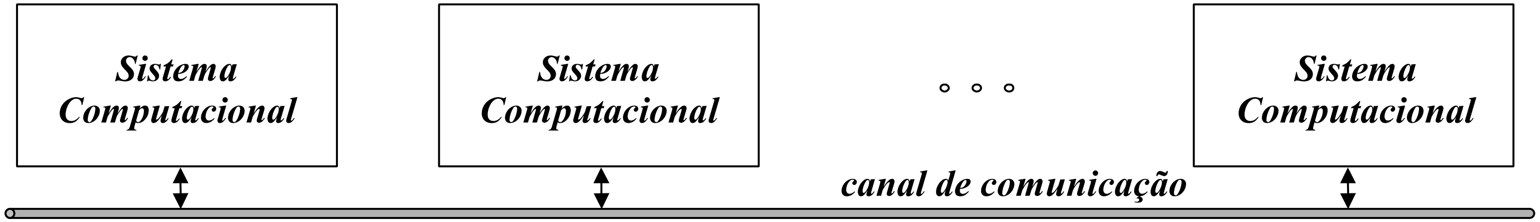
- Purpose: provide a user friendly interface, minimizing the response time to external requests
- Method: keeping different users, each one in a different terminal, in direct and simultaneous connection with the system:
  - using multiprogramming, the processor is successively assigned to the execution of the various programs present during very short time intervals (time quantum)
  - since a human response is slow, when compared to a computer, the illusion that the system is entirely dedicated to every user is created.

Real time system



- Purpose: use the computer in the online monitoring and control of physical processes
- Method: variant of an interactive system that allows to impose maximum limits to the response times of the computational system to different classes of external requests

Network operating system



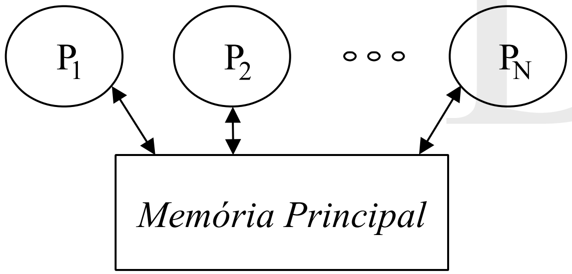
- Purpose: take advantage of the interconnection facilities of computer systems (at the hardware level) to establish a set of services common to an entire community
- Services: file transfer (*ftp*); access to remote file system (*NFS*); resource sharing (printers, etc.); access to remote computational systems (*telnet*, *remote login*, *ssh*); e-mail; access to the world wide web (*browsers*)

Distributed operating system

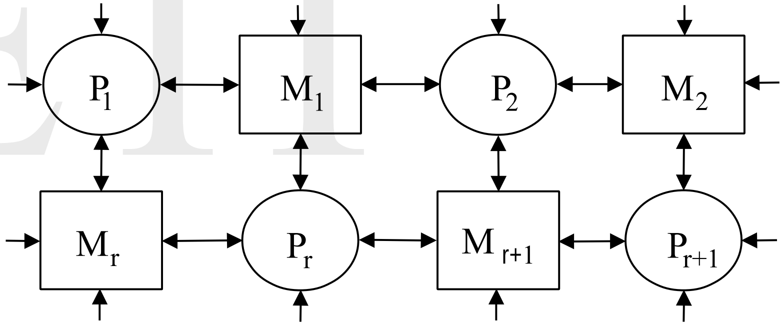
- Purpose: take advantage of the facilities for constructing multiprocessor computing systems, or for interconnecting different computational systems, to establish an integrated interaction environment where the user views the parallel computing system as a single entity
- Methodology: ensure a complete transparency in the access to processors or other resources of the distributed system, in order to allow
  - static/dynamic load balancing
  - increasing the speed of processing by incorporation of new processors or new computational systems
  - parallelization of applications
  - implementation of fault tolerance mechanisms

Multiprocessing vs. Multiprogramming

- Parallelism - ability of a computational system to simultaneously run two or more programs
  - more than one processor is required (one for each simultaneous execution)
- The operating system of such computational systems supports **multiprocessing**



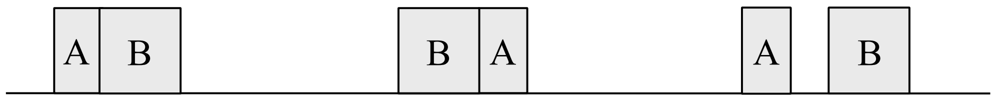
symetric processing (SMP)



planar mesh

Multiprocessing vs. Multiprogramming

- Concurrency - illusion created by a computational system of apparently being able to simultaneously run more programs than the number of existing processors
  - The existing processor(s) must be assigned to the different programs in a time multiplexed way
  - The operating system of such computational systems supports **multiprogramming**



- Programs A and B are executing concurrently in a single processor computational system

Internal structure

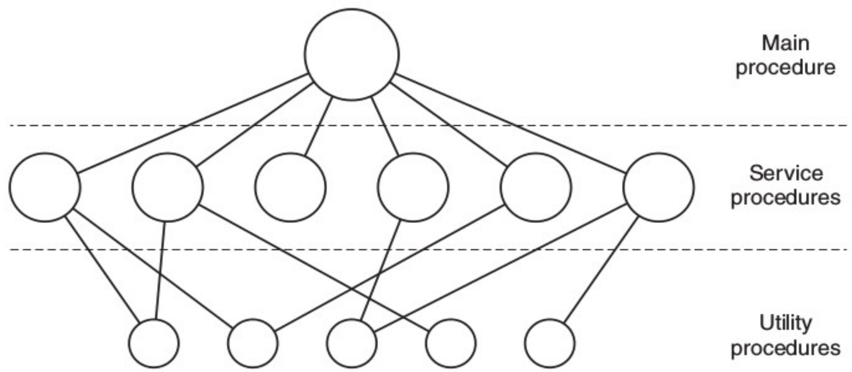
- A typical operating system should:
  - implement a graphical environment for user interaction
  - be multiuser (allow different users, even at the same time)
  - be multitask (run several programs at the same time)
  - implement virtual memory
  - allow access, in an almost indistinguishable way, to local and remote file systems and other input/output devices
  - allow connection to remote machines
  - contain a large collection of device drivers
  - allow dynamic connection of devices (*plug and play*)
- Thus:
  - it is a quite complex program, with a huge dimension (millions lines of code)
  - its design and implementation requires a huge effort in order to work properly

Design approaches

- In terms of design of operating systems, the following approaches have been tried:
  - Monolithic systems
  - Layered systems
  - Microkernels systems
  - Client-server model
  - Virtual machines
  - Exokernels

Design approaches

- Monolithic approach:
  - The most common approach
  - A single program running in kernel mode, composed of
    - An entry point that invokes the requested service procedure
    - A set of service procedures that carry out the system calls
    - A set of auxiliary procedures
  - Every part of the operating systems can 'see' the others
    - very efficient but hard to test and modify



Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Design approaches

- Layered approach:
  - A modular approach, based on a hierarchy of layers, each one on top of another
  - Interaction is only possible between adjacent layers
  - Not that simple to design
    - requires a careful design on the role of each layer
  - Simple to test and modify but less efficient
    - there is an overhead due to communication between layers
  - Designers can choose the user-kernel boundary, i.e., what layers run in kernel mode

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

Figure 1-25. Structure of the THE operating system.

Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Design approaches

- Microkernel approach:
  - A modular approach, based on small, well-defined modules
    - one of them, the microkernel, runs in kernel mode
    - the others run in user mode
  - Typically, the microkernel implements process management, memory management and a basic communication mechanism
  - The other modules run as system processes, running in user mode and communicating using the microkernel communication mechanism
    - These modules can be launched at start up or dynamically at run time (plug and play)

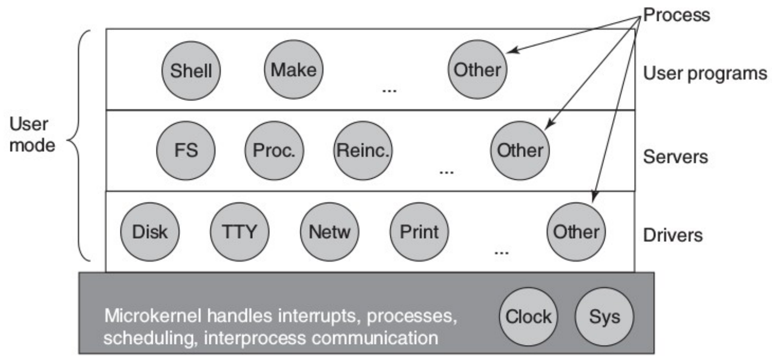
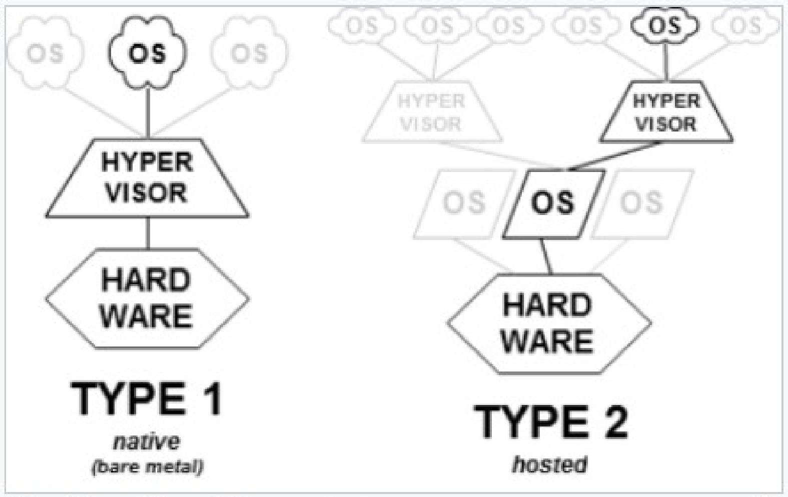


Figure 1-26. Simplified structure of the MINIX system.

Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Design approaches

- Virtual machines (hypervisors):
  - Creates different virtual operating platforms, where guest OSs can be installed
  - There are two types
    - type-1 or native hypervisor: runs directly in the host's hardware
    - type-2 or hosted hypervisor: runs on top of the host's operating system
  - Type-1 examples:
    - z/VM, Xen, Hyper-V, VMware ESX
  - Type-2 examples:
    - VirtualBox, VMware Workstation, Parallels
  - Hybrid examples:
    - KVM, bhyve



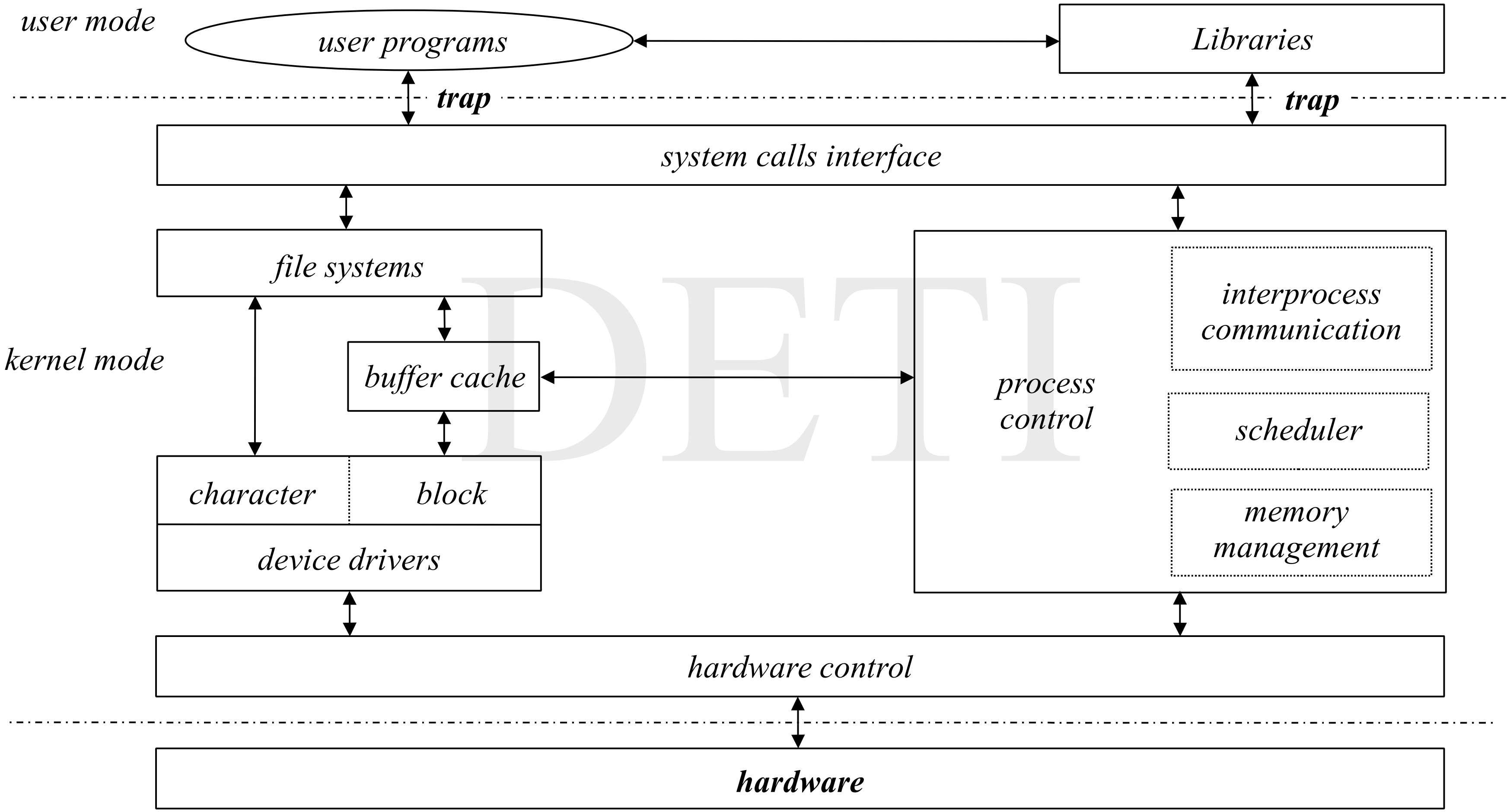
Type-1 and type-2 hypervisors

wikipedia: hypervisor (oct, 2017)

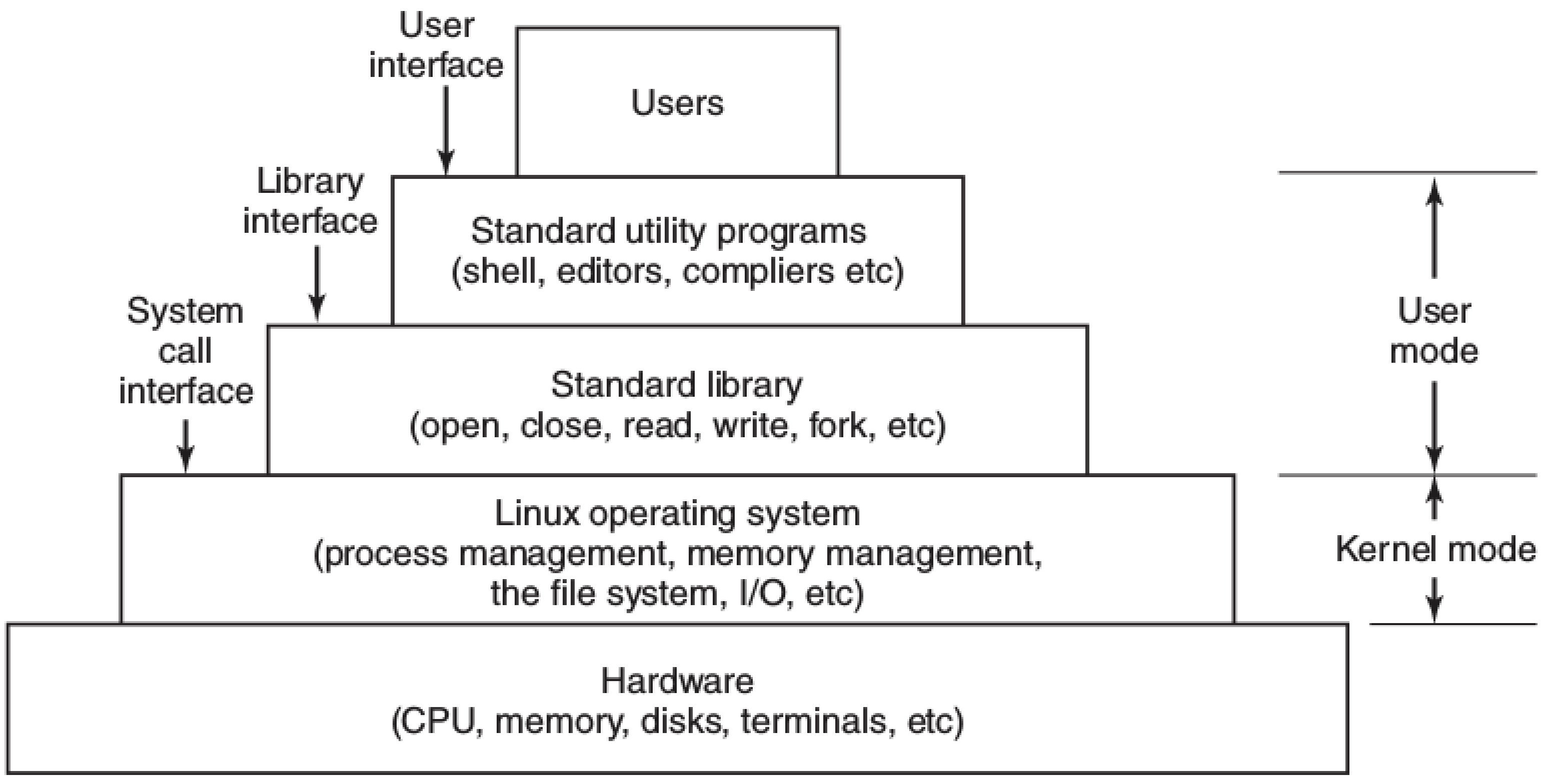
Design approaches

- Client-server approach:
  - A modular approach, based on clients and servers
    - typically, communication is based on message passing
  - A low level microkernel can be present
  - Can be generalized to a multimachine system
- Exokernels:
  - Based on a tiny kernel, running in kernel mode, providing few hardware abstractions
    - resources are partitioned, instead of cloned
    - resources are allocated to virtual machines and their usage is controlled
  - Allows the implementation of custom abstractions
  - More efficient, since it saves a layer of mapping

Traditional Unix internal structure

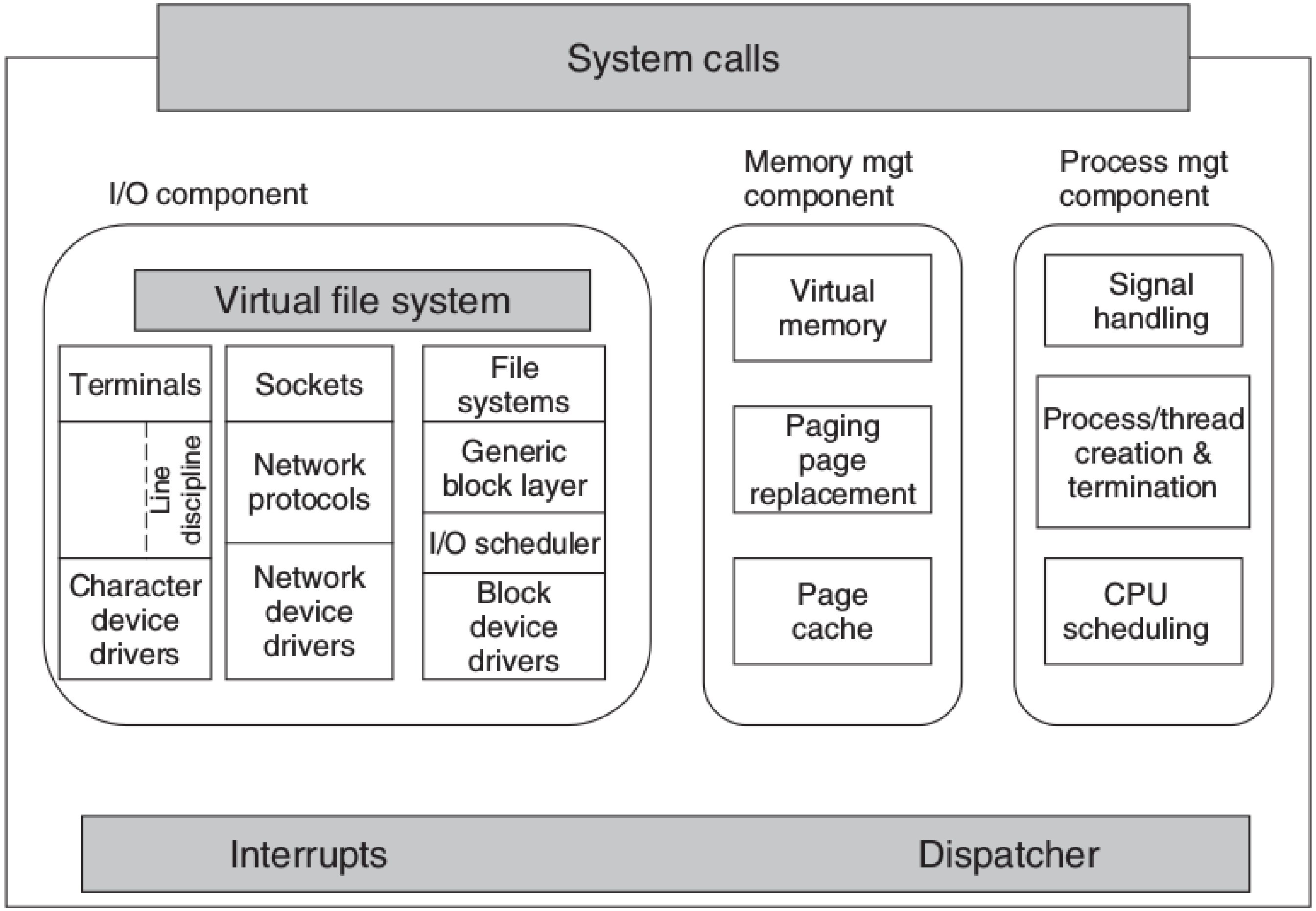


Linux global structure



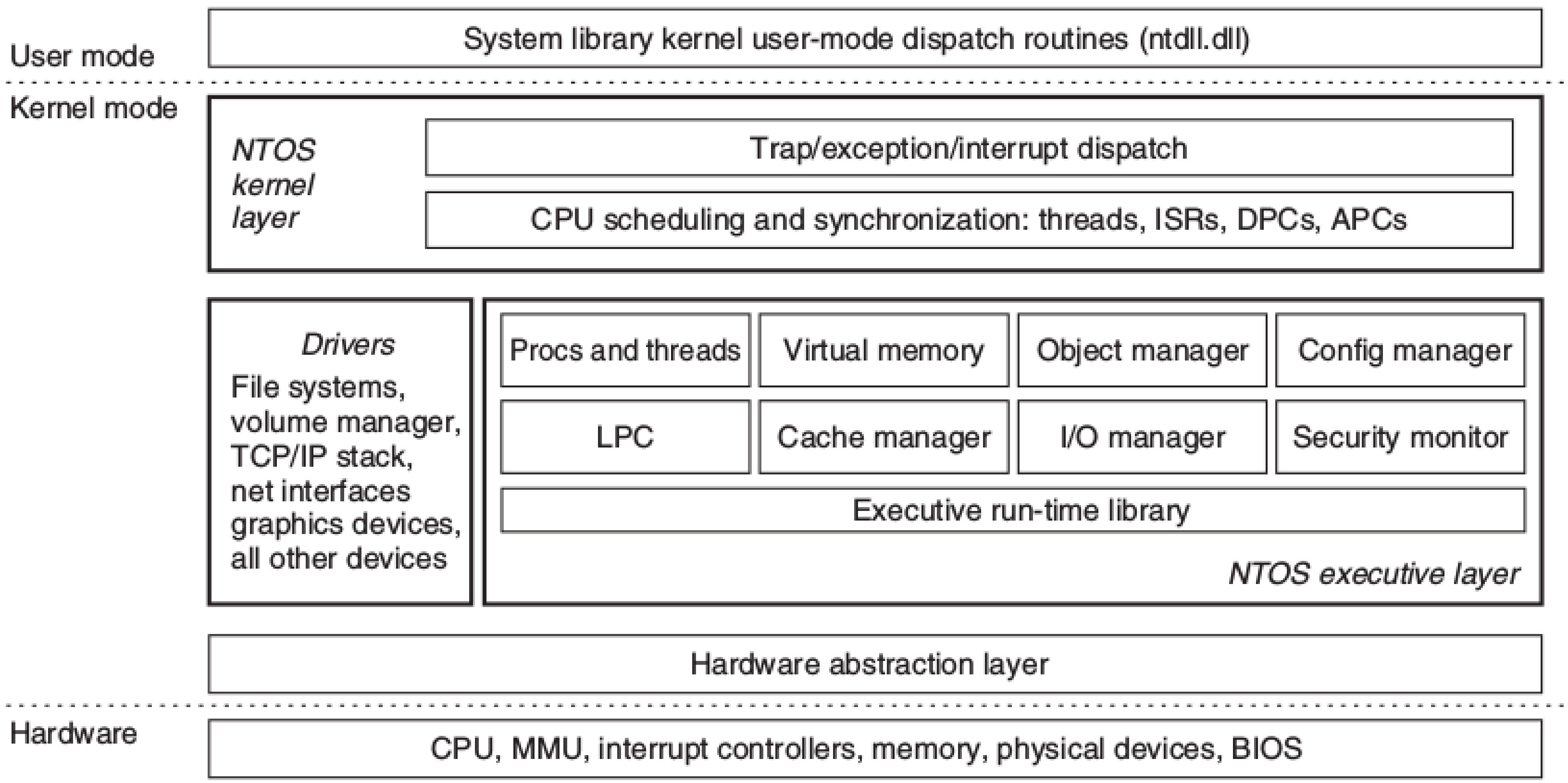
Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Linux kernel structure



Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Windows kernel structure



Modern Operating Systems, Andrew Tanenbaum & Herbert Bos

Bibliography

Operating Systems Concepts; Silberschatz, Galvin & Gagne; John Wiley & Sons, 9th Ed

Chapter 1: *Introduction*

Chapter 2: *Operating-System Structures*

Modern Operating Systems; Tanenbaum & Bos; Prentice-Hall International Editions, 4rd Ed

Chapter 1: *Introduction*

Operating Systems; Stallings, Prentice-Hall International Editions, 7th Ed

Chapter 1: *Computer System Overview*

Chapter 2: *Operating System Overview*