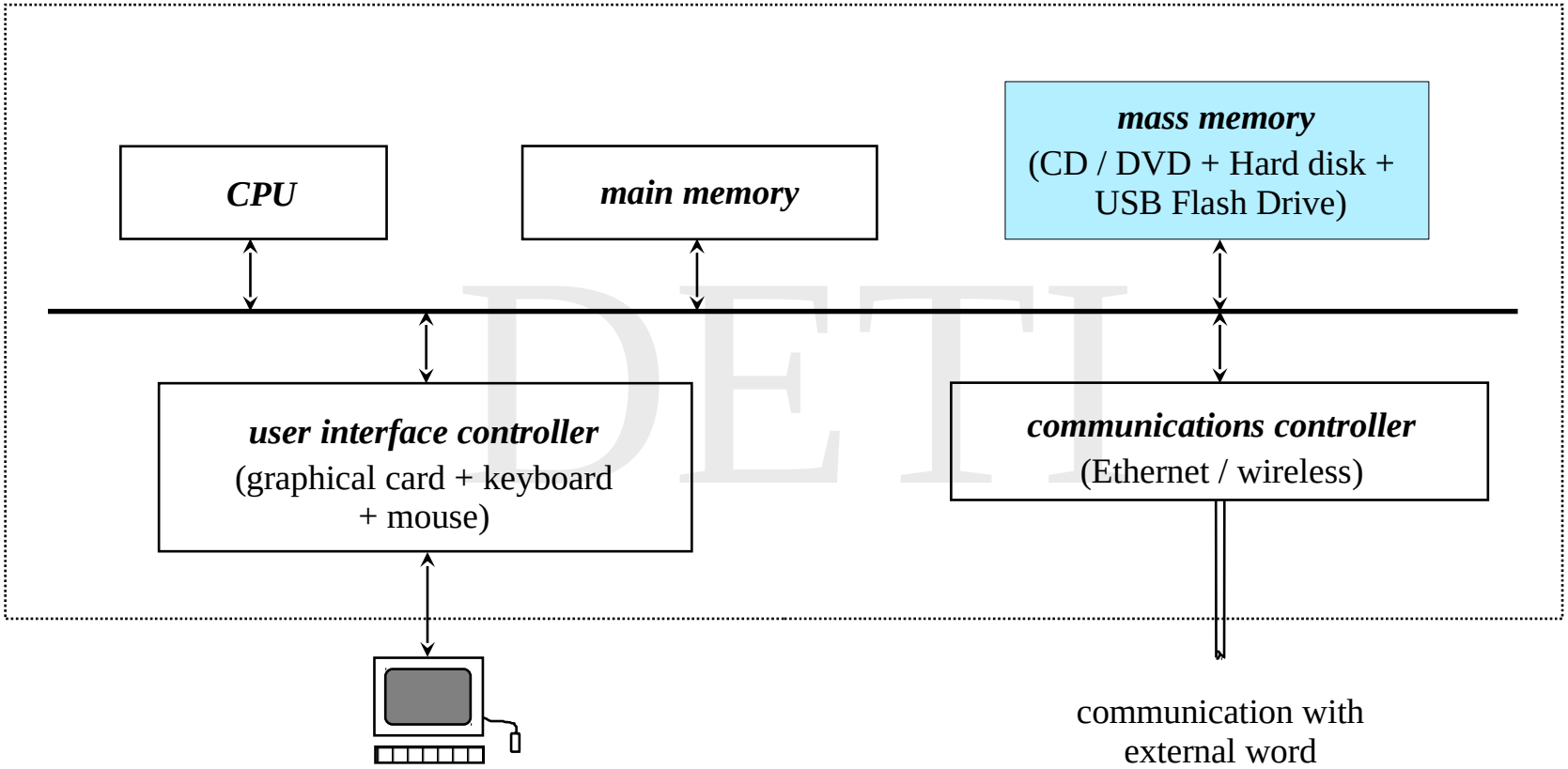
Sistemas de Operação / Operating Systems

File systems in a nutshell

Artur Pereira

Typical computational system



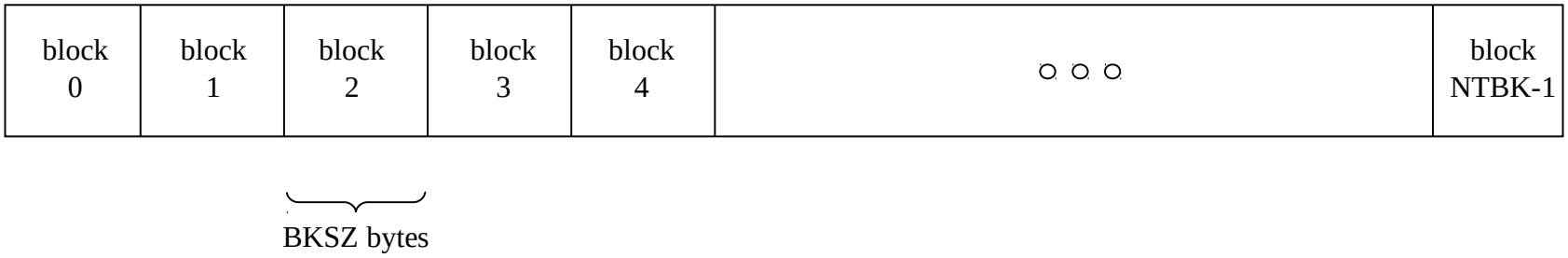
Types of mass memory devices

Type	Technology	Capacity	Type of use	transfer rate
		(Gbytes)		(Mbytes/s)
CD-ROM	mechanical / optical	0.7	read	0.5
DVD	mechanical / optical	4 – 8	read	0.7
HDD	mechanical / magnetical	250 – 4000	read /write	480
USB FLASH	semiconductor	2 – 256	read /write	30 (r) / 15 (w)
SSD	semiconductor	64 – 512	read /write	500

Operational abstraction of mass memory

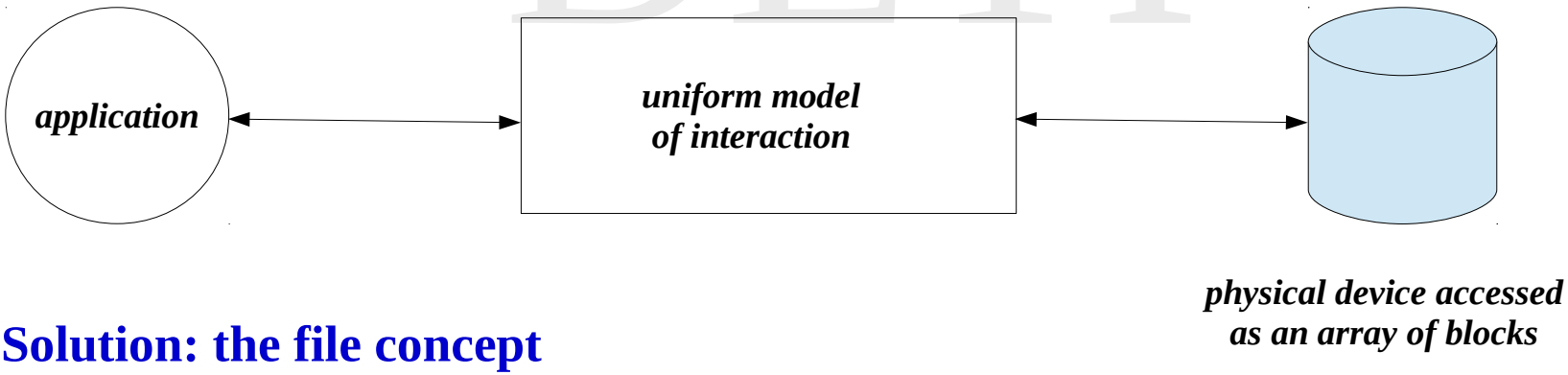
- Mass memory can be seen in operational terms as a very simple model
  - each device is represented by an array of NTBK storage blocks, each one consisting of BKSZ bytes (typically BKSZ ranges between 256 and 32K)
  - access to each block for reading or writing is done in a random manner
- Note that:
  - a block is the unit of interaction

    thus, a single byte can not be accessed directly



User abstraction of mass memory

- The direct manipulation of the information contained in the physical device can not be left entirely to the responsibility of the application programmer
- The inherent complexity of its structure and the need to impose quality criteria related to the efficiency of access, integrity and sharing require the creation of a uniform model of interaction



- **Solution: the file concept**

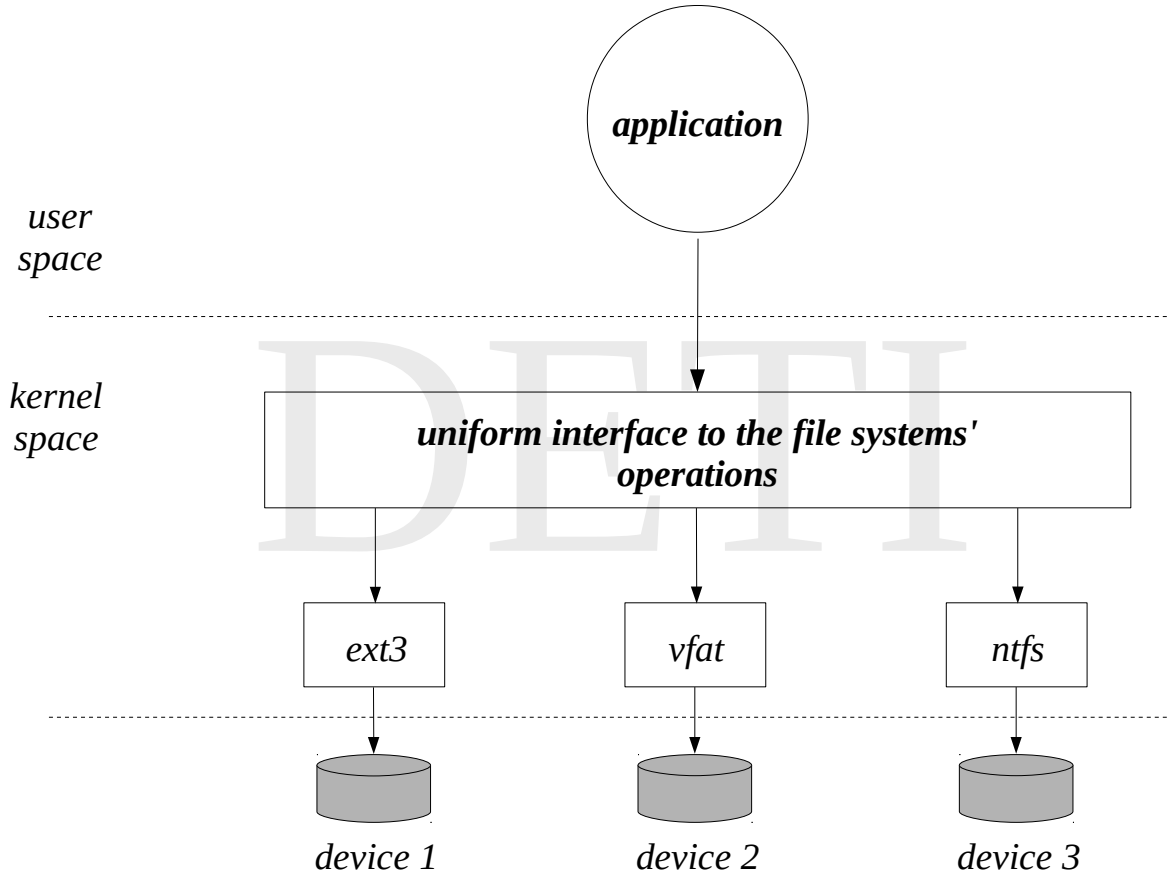
The file concept

- *file* is the *logical unit of storage in mass memory*
  - meaning that reading and writing information is always done within the strict scope of a file
- basic elements of a file
  - *name* – the generic way of referring to the information
  - *contents* – the information itself, organized in a sequence of bits, bytes, lines or registers, whose precise format is defined by the creator of the file and which has to be known by whoever accesses it
- From the point of view of the application programmer, a file is understood as an abstract data type, characterized by a set of *attributes* and a set of *operations*

The file concept

- The role of the operating system is to implement this data type, providing a set of operations, so-called *system calls*, which establishes a simple and secure communication interface for accessing the mass memory
- The part of the operating system that is dedicated to this task is called *file system*
- Different implementations of the file data type lead to different types of file systems
- Nowadays, operating systems implement different types of file systems, associated with different physical devices, or even with the same
  - This feature facilitates interoperability, establishing a common means of information sharing among heterogeneous computational systems

The file concept



Types of files

- From the operating system point of view, there are different types of files:
  - *ordinary or regular file* – a file from the user point of view
  - *directory* – file used to track, organize and locate other files and directories
  - *shortcut (symbolic link)* – file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution
  - *character device* – a special file representing a device handled in bytes
  - *block device* – a special file representing a device handled in blocks
  - *socket* – a special file used for inter-process communication
  - *named pipe* – another special file used for inter-process communication
- text files, image files, video files, application files, etc., are all regular files.

device file

é um ficheiro especial que serve de interface (plataforma-local onde dois sistemas trocam informação) para um device driver e que aparece no file system como se fosse um arquivo comum. Estas special files permitem que as aplicações interagem com o device usando o seu device driver através das das chamadas ao sistema de escrita-leitura.

character device:

tipos de ficheiros que costumam estar na directoria /dev e que lidam com pequenas quantidades de leitura e de escrita.
exemplo: portas, sound cards, etc…

block device:
estas ja lidam com grandes quandidades, sendo necessário blocks e buffering.

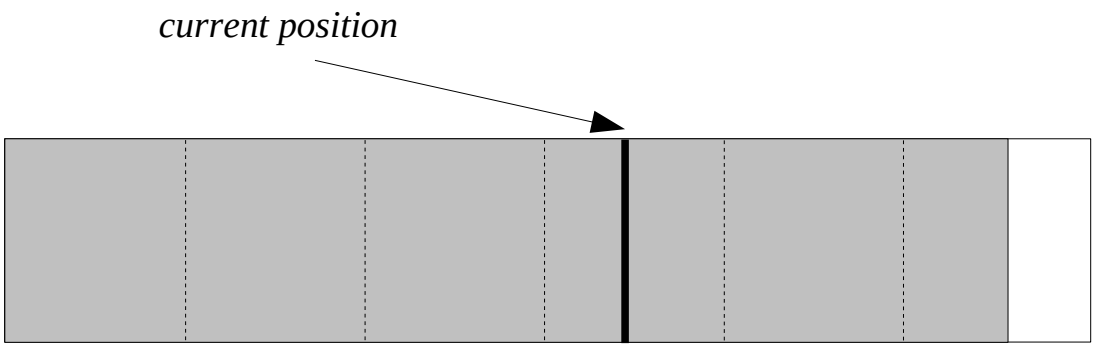
Attributes of files

- Common attibutes of a file
  - *type* – one of the referred above
  - *name* – the way users refer to the file
  - *internal identification* – the way the file is known internally
  - *size(s)* – size in bytes of information; space occupied on disk
  - *ownership* – who the file belongs to
  - *permissions* – who can access the file and how
  - *access and modification times* – when the file was last accessed or modified
  - *location of data in disk* – ordered set of blocks/clusters where the file contents is stored
    - Remember that the block is the unit of interaction with the disk

Operations on files

- Common operations on regular files
  - *creation, deletion*
  - *opening, closing*
  - *reading, writing, truncation*
  - *positioning* – in order to allow random access

quit, force quit.



Operations on files

- A directory can be seen as a set of (*directory*) *entries*, each one representing a file (of any type)
- Common operations on directories
  - *creation, deletion* (if empty)
  - *opening* (only for reading), *closing*
  - *reading* (directory entries)

- Common operations on shortcuts (symbolic links)
  - *creation, deletion*
  - *reading* (the value of the symbolic link)

- Common operations on files of any type
  - *get attributes* (access and modification times, ownership, permissions)
  - *change attributes* (ownership only root or admin)
  - *change ownership*

Unix operations on files

- As referred to before, the operations are *system calls*
- Common system calls on regular files
  - mknod, unlink, creat, open, close, read, write, truncate,