Input-Output Management

E. Campo M. Knoblauch Ó. López J. Clemente

Departamento de Automática
Universidad de Alcalá

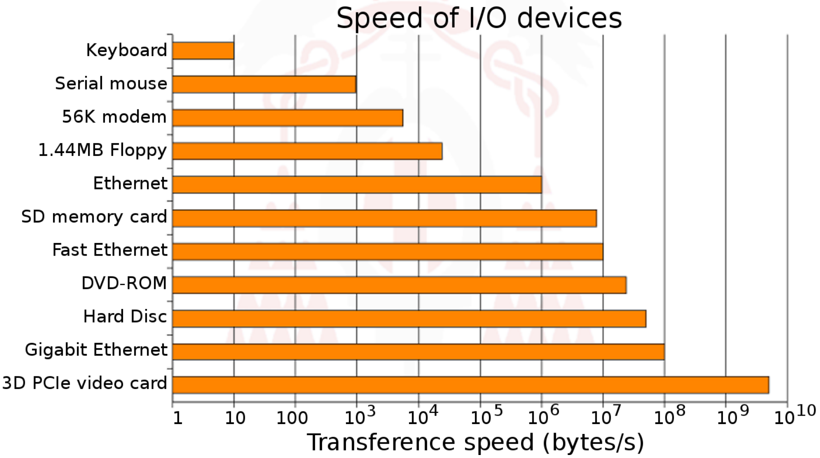
/gso>

Index

Issues with I/O

- Extensive use of I/O
  - More I/O concurrence → better usage of the system
- Every I/O device has its own idiosyncrasy
  - Diverse functionality
    - Storage (Discs)
    - User interface (Keyboard, mouse)
    - Communications (Network cards, modem)
    - Etc.
  - Ways to access them
  - Different degrees of autonomy and intelligence
- Devices need help form the kernel
- Different speed

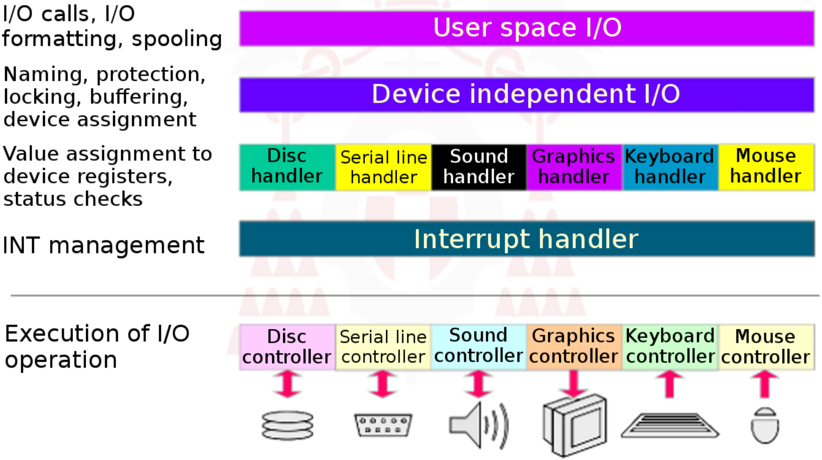
Speeds of I/O devices



I/O management

- The operating system is the link between the user application and the I/O hardware
- Goals of the I/O subsystem of the O.S.:
  - Hide HW from applications
  - Uniform access interface (uniform names)
  - Device-independence
    - Ability to write programs that access I/O devices without knowing in advance the type of device
  - Error handling
  - Management of different device types
    - Shared or dedicated
    - Character mode or block mode
    - Sequential access or random access
    - Etc. . .

Layers of I/O software (1/2)

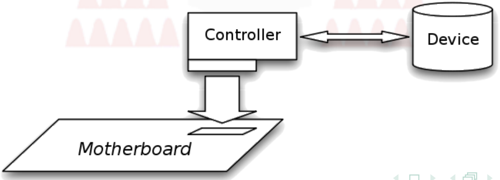


Layers of I/O software (2/2)

- I/O hardware can be updated and modified:
  - The O.S. provides mechanisms for updates
  - What happens if a new HW element is introduced?
    - Monolithic approach versus microkernel approach
- The I/O interface with the final user:
  - Uniform access
    - All devices look the same
  - Device-independent
    - Hides HW from upper levels
    - The user doesn't bother accessing every device
    - Same access primitives, regardless of device type

Elements of I/O

- Every I/O element is directly or indirectly connected to the system bus
- Usually divided in two parts:
  - Device controller or adapter
    - Printed circuit card, connected to an expansion slot of the computer
    - Has some state, control and data registers
  - The device itself
    - Connected to the controller through a slot, wire or similar, which can be standard (i.e. IDE or SCSI)



Interrupt Handlers

- Interrupts are inevitable in most I/O operations
  - They should be hidden as much as possible
  - The O.S. can/must carry out several tasks:
    - Store registers and PSW, and prepare a context and a stack in order to execute an ISR
    - Send an acknowledgement to the interrupt controller
    - Execute the ISR, which will interact with the controller that generated the interrupt
    - Sometimes, wake up a process implied in the I/O and invoke the scheduler
    - Prepare the context and recover the registers of the new process
    - Return from the interrupt and execute the new process

Device Handlers

- They contain specific code for every device
- Duties:
  - Initialize the device
  - Accept abstract read/write requests from the device-independent software
  - Record events
- Usually written by the manufacturer of each hardware device
  - Important: O.S. designers must provide a well defined model of what a device handler does and how should it interact with the O.S.
  - Standard interface for block mode devices and character mode devices

Mapping of Input-Output

- Communication between controller and CPU
  - Control registers: The CPU can read/write them to command the device, find out its state, etc.
  - Data buffers where the O.S. can read/write
- How?
  - Assign an I/O port to every register
    - Port: 8 or 16 bits integer
    - Specific instructions for reading/writing (IN/OUT)
  - Establish a correspondence between registers or memory positions
    - Accesses to these positions actually access the registers
    - Standard load/store instructions (i.e. MOV)
  - Hybrid
    - Ports for the registers, memory for the buffers

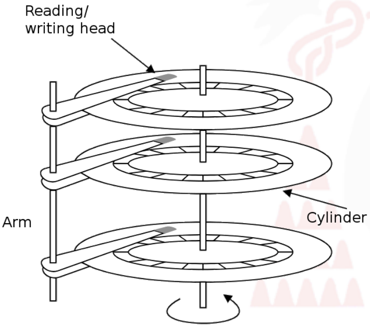
Techniques for performing I/O (1/2)

- Peripheral's operations are asynchronous with respect to those of the processor
- Different communication mechanisms:
  - Polling
  - Interrupts
  - Direct memory access

Techniques for performing I/O (2/2)

- Programmed I/O (Polling)
  - The CPU waits until the device is ready to send the next character (polling)
    - Simple, but with busy wait
- I/O by means of interrupts
  - The CPU starts the I/O operation, and it continues with an interrupt
    - The process that makes the I/O call gets blocked, thus avoiding busy wait
    - An interrupt is produced for every character
- I/O by means of DMA
  - It is the DMA controller who serves characters
    - The DMA controller basically carries out programmed I/O
    - The process that makes the I/O call gets blocked
    - Only one interrupt is produced, and busy wait is avoided

Hardware of a disc



- Disc divided in 512-bytes sectors, referenced by:
  - Cylinder, Head, Sector (CHS)
- Access times
  - Seek time
  - Rotational latency
  - Transference time
- Which one can be optimized?
- On modern discs, the internal structure is transparent. Sectors are referenced with Logic Block Addressing (LBA)

Disc handler

- Goal: read and write disc sectors
- Basic procedure:
  - Initialize DMA
  - Start the motor (floppy discs or hard discs with power saving)
  - Move the heads to the right position
  - Read or write data (completion is signalled by the controller with an interrupt)
  - Stop the motor (floppy discs)
- The operations to be carried out will depend on the degree of intelligence and autonomy of the disc controller

Scheduling of disc I/O (1/2)

- There are several scheduling possibilities
- In order to evaluate every algorithm, a list of requests is used:
  - 98, 183, 37, 122, 14, 124, 65 y 67 (with head starting at track 53)
- First Come - First Served (FCFS)
  - Simplest method to schedule and implement
  - Acceptable with small workloads (equitable)
  - Traverses 640 tracks

Scheduling of disc I/O (2/2)

- Shortest Seek Time First (SSTF)
  - Minimizes seek times
  - Response times are not equitable, external tracks are discriminated and might suffer starvation
  - Traverses 236 tracks
- SCAN
  - Also known as the “elevator algorithm”
  - The C-SCAN variant provides more uniform response times
  - Traverses 236 tracks
- Nowadays, it is the HW disc controller who optimizes the disc I/O requests

Bibliographic references

[Sánchez, 2005] S. Sánchez Prieto.
Sistemas Operativos.
Servicio de Publicaciones de la UA, 2005.

[Tanenbaum, 2009] A. Tanenbaum.
Sistemas Operativos Modernos.
Ed. Pearson Education, 2009.

[Stallings, 1999] W. Stallings.
Organización y arquitectura de Computadores.
Ed. Prentice Hall, 1999.

[Silberschatz, 2006] A. Silberschatz, P. B. Galván y G. Gagne
Fundamentos de Sistemas Operativos.
McGraw Hill. 2006