Operating Systems: I/O Systems

Neerja Mhaskar

Department of Computing and Software, McMaster University, Canada

Acknowledgements: Material based on the textbook Operating Systems Concepts (Chapter 12)

Overview

- Management of I/O devices forms a major component of operating system design and operation.
- The kernel of an operating system uses device-driver modules to use varied and different I/O types.
  - **Device drivers -** are modules that can be plugged into an OS to handle a device or category of similar devices.
- A device controller is in charge of one or more devices.
- Device drivers communicate with device controllers for I/O requests for a device (See slides on I/O structure in Chapter 1).

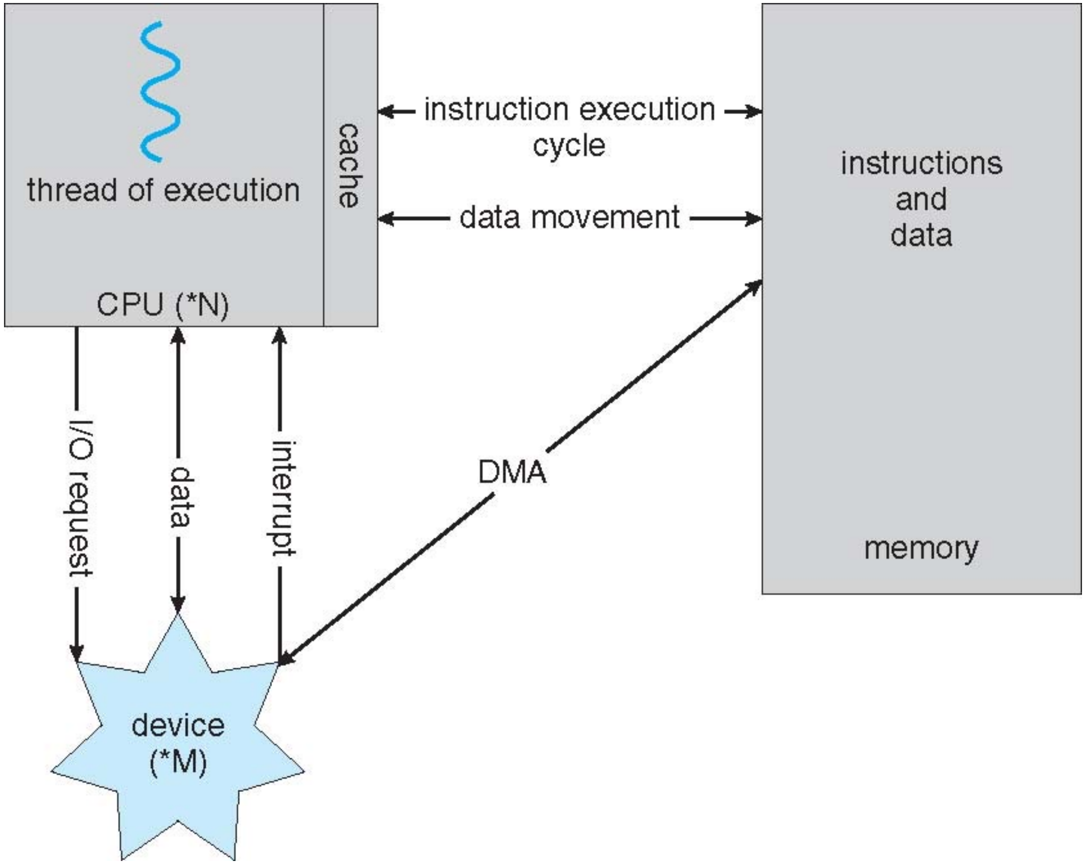
Polling and Interrupts

- The host and device controllers communicate with each other to process an I/O request.
- This involves the host checking if the device is available to accept an I/O request or provide data for an I/O request.
- **Polling or Busy waiting** occurs if host is constantly checking to see if device is available.
  - Polling is efficient if device is fast.
  - Polling inefficient if host rarely finds device ready for service. In which case the device uses **interrupts** to notify the host.

Direct Memory Access (DMA) Structure

- For devices that transfer large quantities of data (such as disk controllers), it is wasteful to tie up the CPU, transferring data in and out of registers one byte at a time.
- Instead, a special-purpose processor called **DMA** is used
- CPU gets an I/O request, it issues a command to the DMA controller providing it with the necessary information.
- The DMA controller handles this request (by directly accesses main memory without CPU's intervention)
- Notifies CPU about it by an interrupt.

Direct Memory Access (DMA) Structure Illustration



A von Neumann architecture

Application I/O Interface

- User application access a wide variety of different devices.
- The detailed differences in I/O devices are abstracted away by identifying a few general kinds.
- Applications are presented with a **common interface** (standardized set of functions) for each general kind.
- Devices are broadly grouped by the OS into the following categories:
  - **Block Devices** (e.g., hard disks)
  - **Character Devices** (e.g., keyboard)
  - **Network Devices**
  - **Clocks and Timers**

Kernel I/O Subsystem

Kernels provide many services related to I/O. Some are listed below.

- **Scheduling** - determine a good order of I/O requests for execution.
- **Buffering** - store original data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch

Kernel I/O Subsystem Continued…

- **Caching** - faster device holding always a copy of data
- **Spooling** and **Device reservation** are facilities provided by the kernel I/O subsystem to deal with concurrent device access
- **Spooling** – A **spool** is a buffer that holds output for a device that cannot accept interleaved data streams
    - E.g.: Printer
- **Device reservation** – The kernel provides exclusive access to a device
    - System calls for allocation and de-allocation
    - However, the programmers need to watch out for deadlock

Kernel I/O Subsystem Continued…

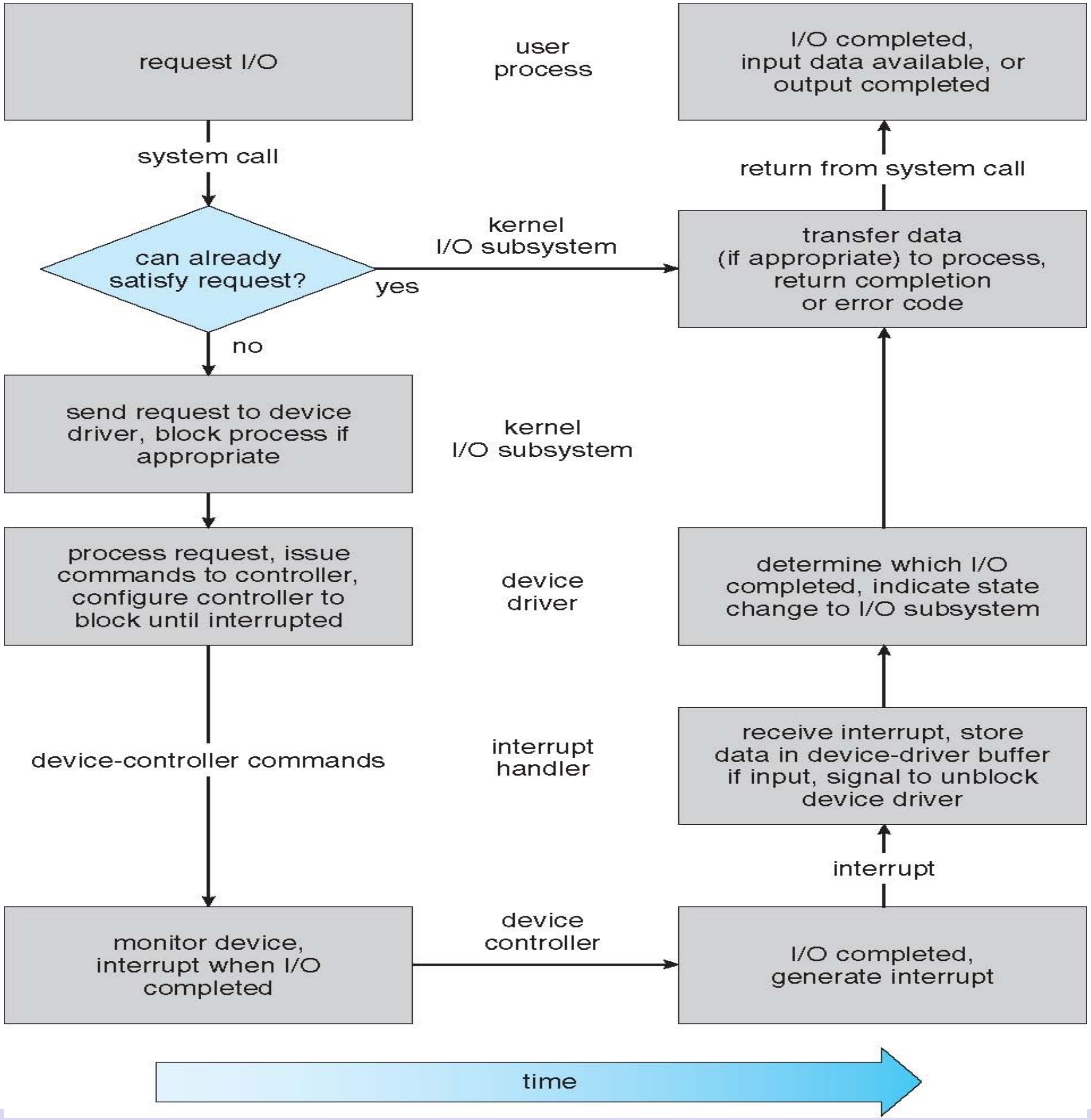
- **Error Handling**
  - Operating systems can often compensate effectively for transient failures. For example, retry a read or write.
  - System error logs hold problem reports
- **Protection**
  - **All I/O instructions are defined to be privileged**
  - I/O must be performed via system calls

Computer System Operation: I/O Structure

To start an I/O operation,

- The device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as “read a character from the keyboard”).
- The controller starts the transfer of data from the device to its local buffer.
- Once the transfer of data is complete, the device controller sends an interrupt signal to CPU.
- CPU transfers control to ISR which handles this request.

Life Cycle of An I/O Request (for your reference only)


request I/O
user process
I/O completed, input data available, or output completed
system call
return from system call
can already satisfy request?
kernel I/O subsystem
yes
transfer data (if appropriate) to process, return completion or error code
no
send request to device driver, block process if appropriate
kernel I/O subsystem
process request, issue commands to controller, configure controller to block until interrupted
device driver
determine which I/O completed, indicate state change to I/O subsystem
device-controller commands
interrupt handler
receive interrupt, store data in device-driver buffer if input, signal to unblock device driver
interrupt
monitor device, interrupt when I/O completed
device controller
I/O completed, generate interrupt
time