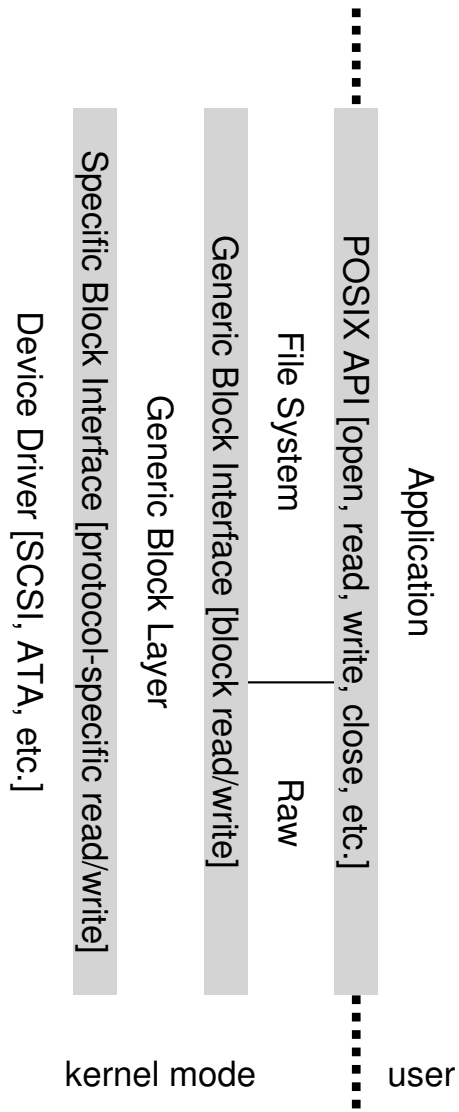
CSL 301
OPERATING SYSTEMS

Lecture 24
I/O

Instructor
Dr. Dhiman Saha



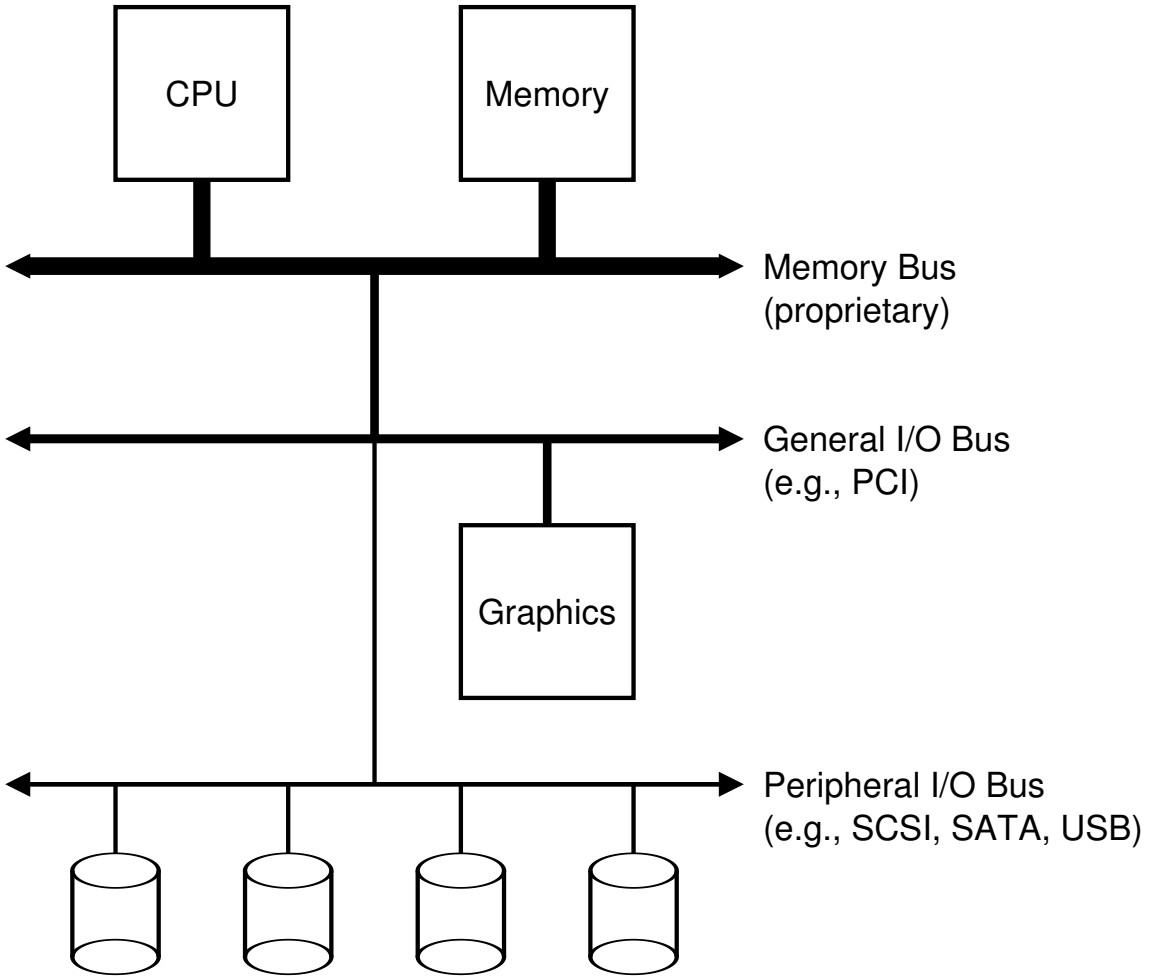
The File System Stack

Persistence

I/O

- How should I/O be integrated into systems?
- What are the general mechanisms?
- How can we make them efficient?

I/O devices connect to the CPU and memory via a bus



Device Model

- Block device e.g disk
- Character device e.g keyboard
- The device interface (Abstracts out the internal details)
  - Commands supported
  - Current status of the device e.g busy
  - Data that needs to be transferred

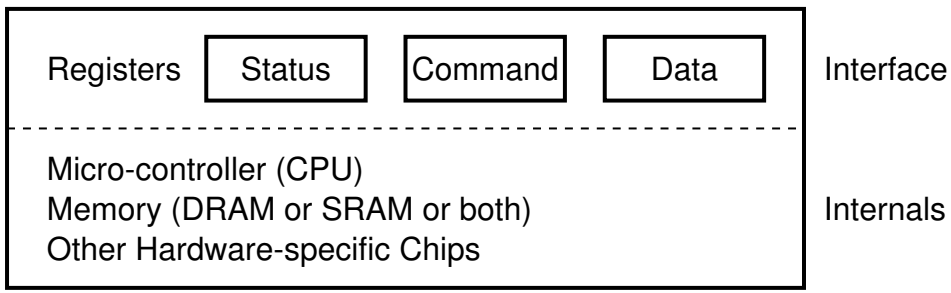


Figure 36.3: **A Canonical Device**

How to deal with read/write to device

Two Approaches

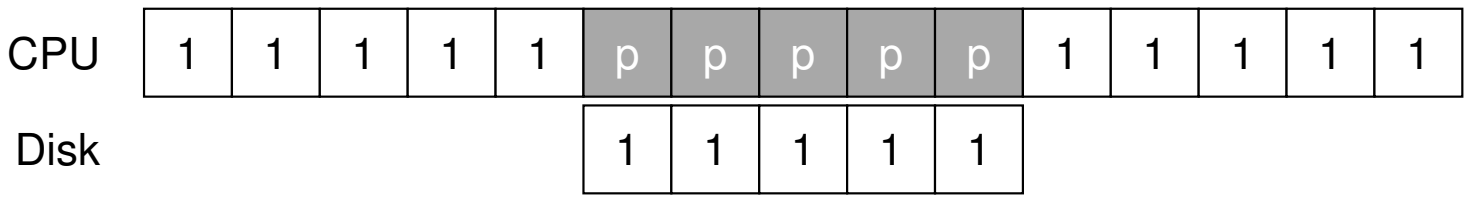
- Explicit instructions
  - e.g., `in` and `out` in `x86`
  - r/w to specific registers like status and command
- Memory Mapped I/O
  - Registers are abstracted as memory locations
  - Certain parts of memory are reserved
  - OS r/w to those specific locations
  - Memory hardware does the translation

The Protocol

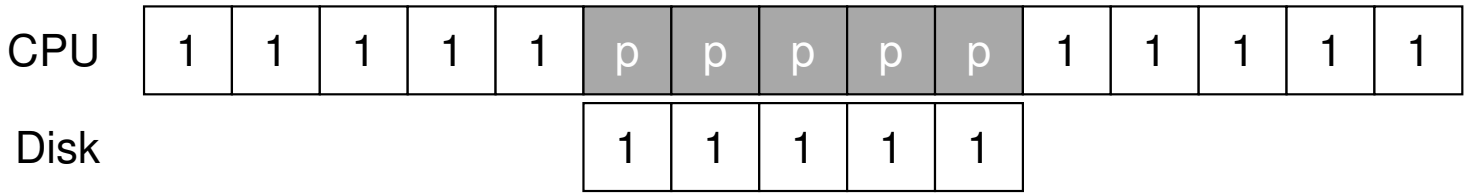
```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (Doing so starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

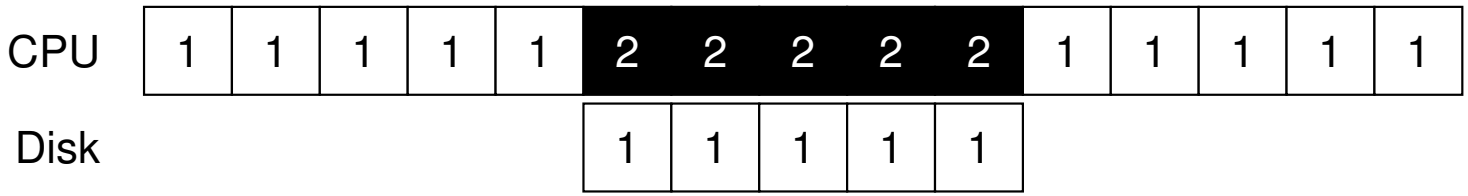
- Idea of polling
- Notion of Programmed I/O
- Both waste CPU cycles

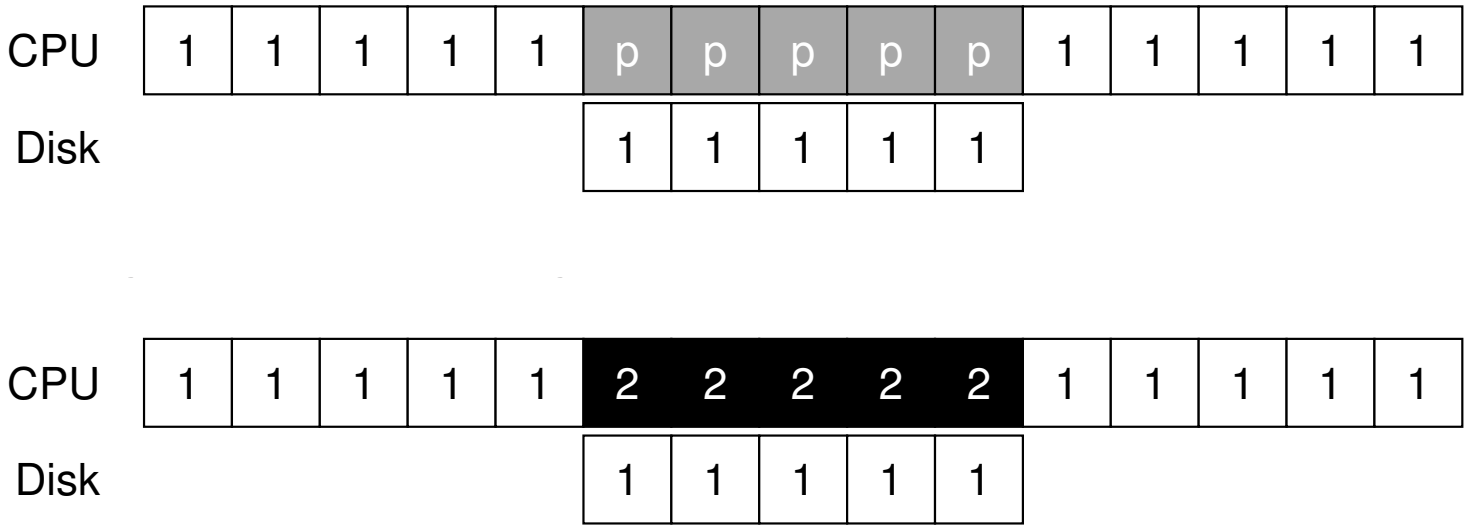
How can the OS check device status without frequent polling?


CPU
1 1 1 1 1 p p p p p 1 1 1 1 1
Disk
1 1 1 1 1

Polling Vs Interrupts







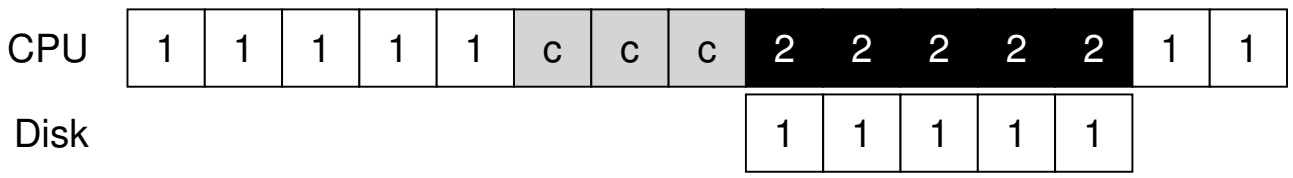
Issue	Interrupts
Livelock	

The Interrupt Handlers

- Already discussed
- Interrupt Descriptor Table
- Interrupt Number
- Interrupts allow for overlap of computation and I/O,

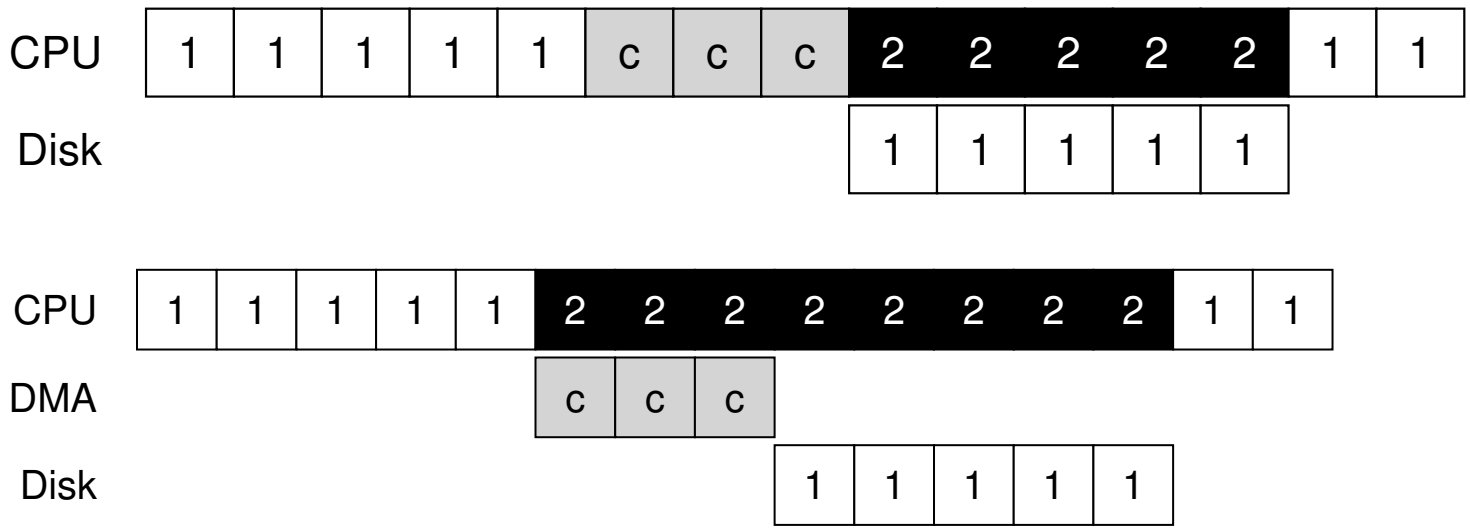
Programmed I/O

PIO



- CPU cycles still wasted in copying data to/from device registers.

Alternative Direct Memory Access (DMA)



The Need for Device Drivers

- Devices have specific interfaces.
- The OS needs to be general and device-neutral.
- Device drivers encapsulate the details of device interaction.

Device Driver

- A piece of software in the OS must know in detail how a device works.
- This piece of software is called a device driver,
- Any specifics of device interaction are encapsulated within

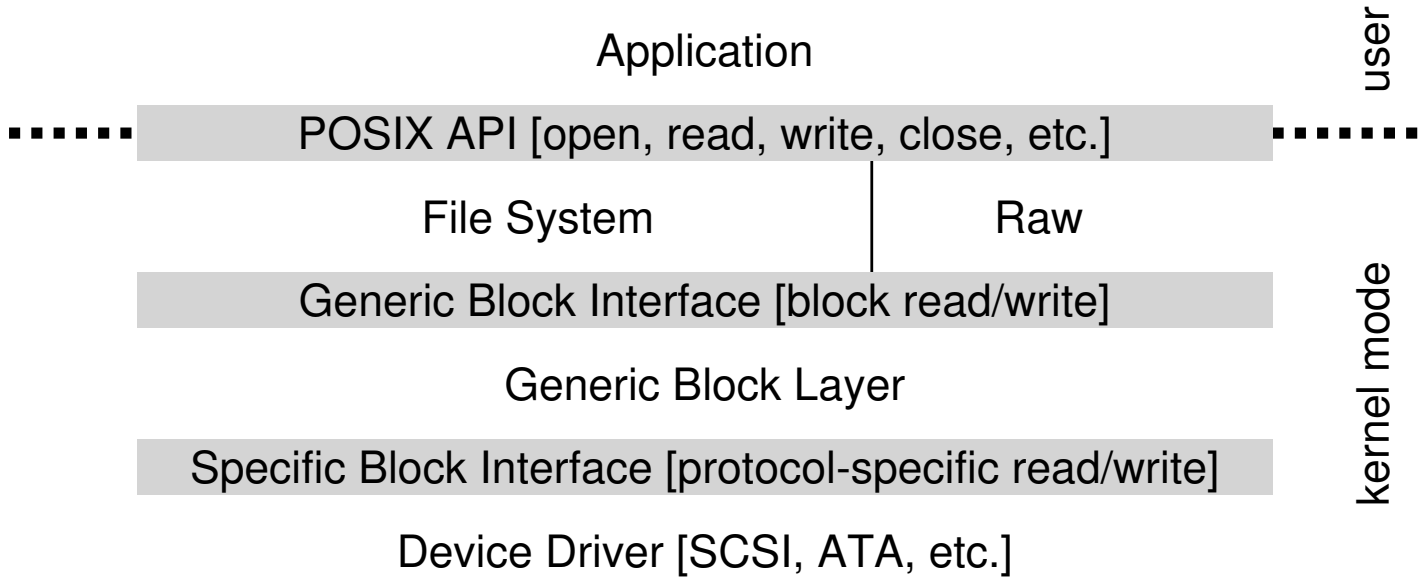
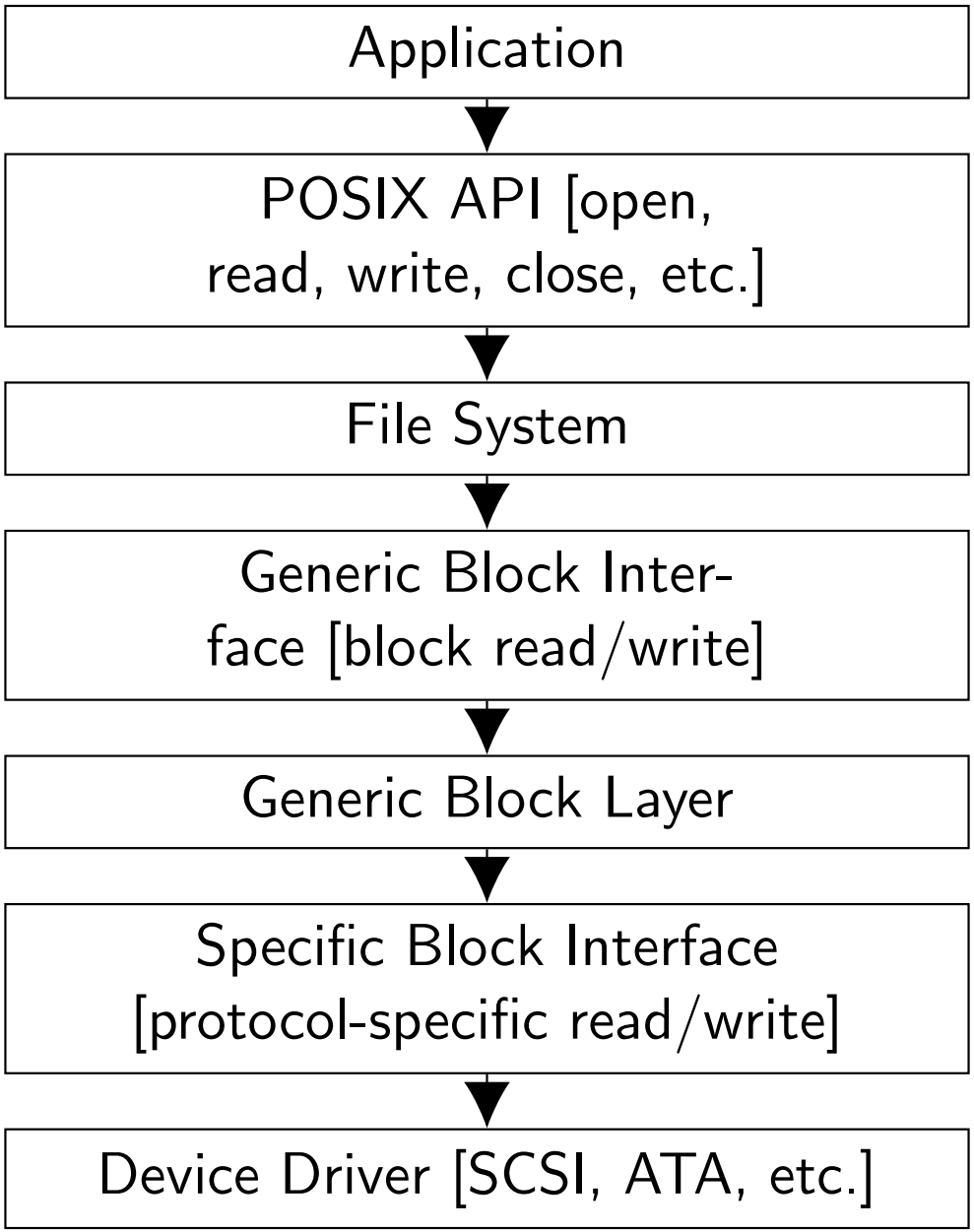


Figure 36.4: **The File System Stack**

The File System Stack



Summary

- Two techniques to improve device efficiency:
  - Interrupts
  - Direct Memory Access (DMA)
- Two approaches to accessing device registers:
  - Explicit I/O instructions
  - Memory-mapped I/O
- Device drivers encapsulate low-level details to make the OS device-neutral.