
東南大學
SOUTHEAST UNIVERSITY

# OPERATING SYSTEM CONCEPTS

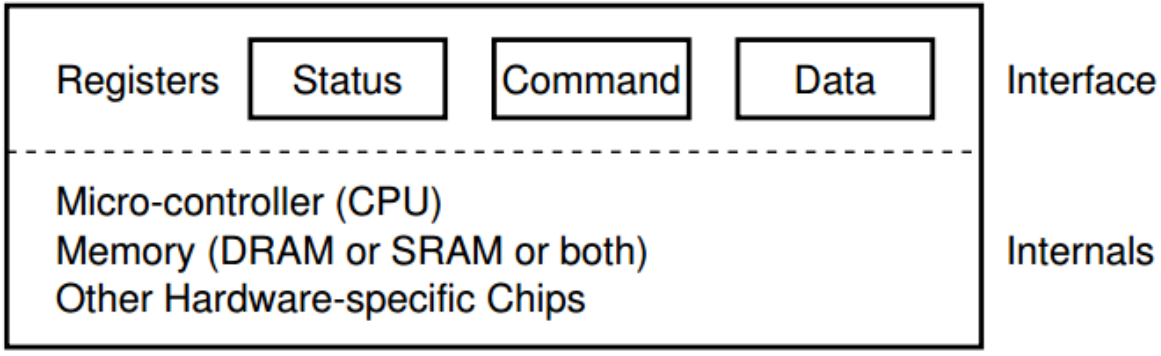
## Chapter 13. I/O Systems

A/Prof. Kai Dong

# Contents

# A Canonical Device

- A device has two important components.
  - The first is the hardware interface it presents to the rest of the system
  - The second part of any device is its internal structure, which is implementation specific.




- Three registers:
  - Status — to be read to see the current status of the device;
  - Command — to tell the device to perform a certain task;
  - Data — to pass data to , or get data from the device.

# A Canonical Protocol

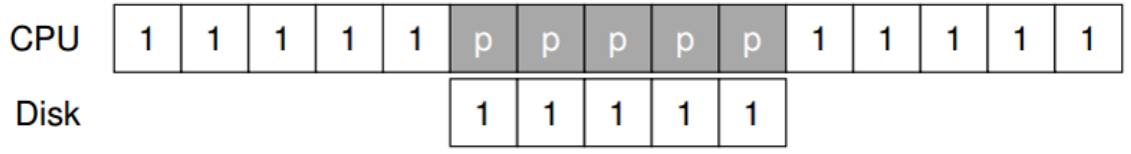



- Polling is inefficient

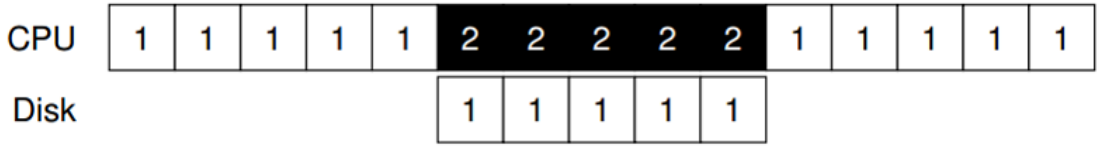
```
1  While (STATUS == BUSY)
2          ;       // wait until device is not busy
3  Write data to DATA register
4  Write command to COMMAND register
5          (Doing so starts the device and executes the command)
6  While (STATUS == BUSY)
7          ;       // wait until device is done with your request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

# Interrupts

- Interrupt-driven
- Instead of polling the device repeatedly, the OS can issue a request, put the calling process to sleep, and context switch to another task. When the device is finally finished with the operation, it will raise a hardware interrupt, causing the CPU to jump into the OS at a pre-determined interrupt service routine (ISR) or more simply an interrupt handler.
  - Without interrupts




  - Interrupt-driven

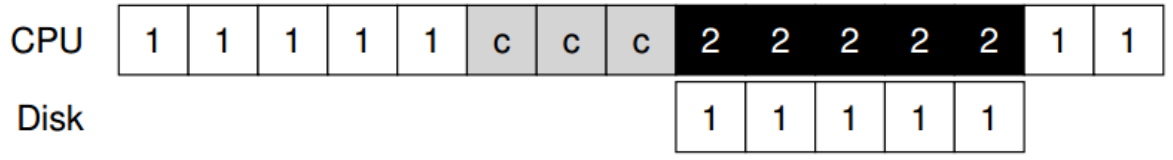


# Interrupts

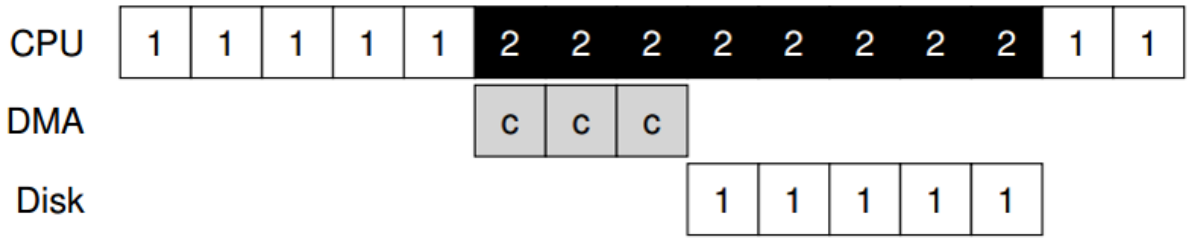

- Take the device speed into account
  - If a device is fast, it may be best to poll; if it is slow, interrupts are best.
  - If the speed of the device is not known, or sometimes fast and sometimes slow, it may be best to use a hybrid that polls for a little while and then, if the device is not yet finished, uses interrupts.
- Livelocks
  - In networks, when a huge stream of incoming packets each generate an interrupt, it is possible for the OS to livelock, that is, find itself only processing interrupts and never allowing a user-level process to run and actually service the requests.
  - Slashdot effect
  - Coalescing: multiple interrupts can be coalesced into a single interrupt delivery

# Direct Memory Access

- One remaining problem:
- Suppose a transfer of a large chunk of data to a device
  - The CPU is overburdened




- Direct Memory Access (DMA).
  - A DMA engine is essentially a very specific device within a system that can orchestrate transfers between devices and main memory without much CPU intervention.



# Direct Memory Access




- How to communicate with devices?
  - I/O instructions.
    - » E.g., the in and out instructions in x86.
    - » Such instructions are usually privileged.
  - Memory mapped I/O.
    - » The hardware makes device registers available as if they were memory locations.

# Device Driver

- Device driver
  - A piece of software at the lowest level in the OS which know in detail how a device works.

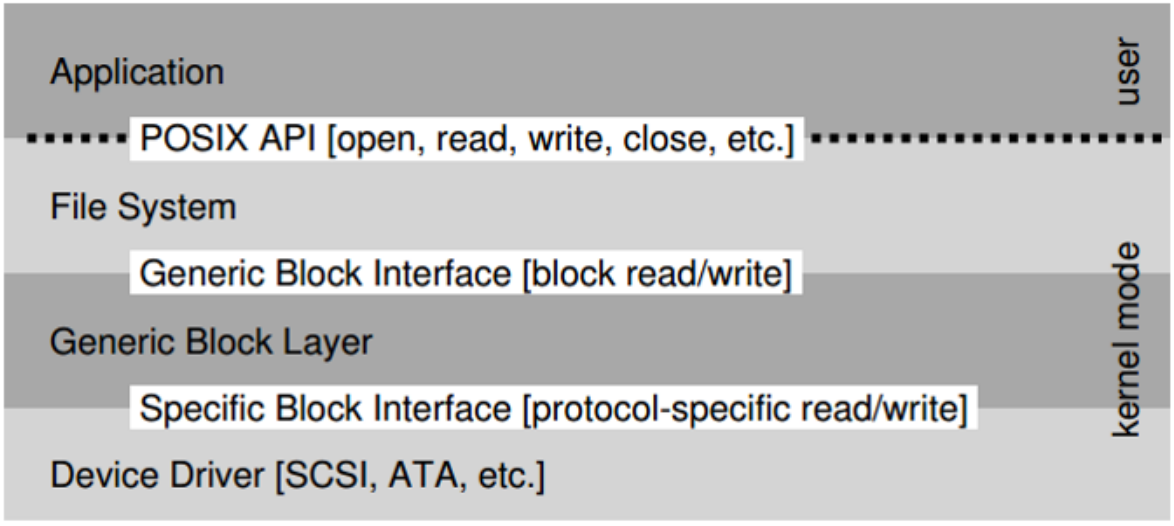



Figure 36.3: **The File System Stack**