Lecture 17: Communication with I/O Devices

Mythili Vutukuru
IIT Bombay
Input/Output Devices

- I/O devices connect to the CPU and memory via a bus
  - High speed bus, e.g., PCI
  - Other: SCSI, USB, SATA

- Point of connection to the system: port
Simple Device Model

• Block devices store a set of numbered blocks (disks)
• Character devices produce/consume stream of bytes (keyboard)
• Devices expose an interface of memory registers
  – Current status of device
  – Command to execute
  – Data to transfer
• The internals of device are usually hidden

Figure 36.2: A Canonical Device
How does OS read/write to registers?

• How does OS read/write to registers like status and command?
  
• Explicit I/O instructions
  – E.g., on x86, \texttt{in} and \texttt{out} instructions can be used to read and write to specific registers on a device
  – Privileged instructions accessed by OS
  
• Memory mapped I/O
  – Device makes registers appear like memory locations
  – OS simply reads and writes from memory
  – Memory hardware routes accesses to these special memory addresses to devices
A simple execution of I/O requests

- 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

- Polling status to see if device ready – wastes CPU cycles
- Programmed I/O – CPU explicitly copies data to/from device
Interrupts

- Polling wastes CPU cycles

Instead, OS can put process to sleep and switch to another process

- When I/O request completes, device raises interrupt
Interrupt handler

• Interrupt switches process to kernel mode

• Interrupt Descriptor Table (IDT) stores pointers to interrupt handlers (interrupt service routines)
  – Interrupt (IRQ) number identifies the interrupt handler to run for a device

• Interrupt handler acts upon device notification, unblocks the process waiting for I/O (if any), and starts next I/O request (if any pending)

• Handling interrupts imposes kernel mode transition overheads
  – Note: polling may be faster than interrupts if device is fast
Direct Memory Access (DMA)

- CPU cycles wasted in copying data to/from device
- Instead, a special piece of hardware (DMA engine) copies from main memory to device
  - CPU gives DMA engine the memory location of data
  - In case of read, interrupt raised after DMA completes
  - In case of write, disk starts writing after DMA completes
Device Driver

• **Device driver**: part of OS code that talks to specific device, gives commands, handles interrupts etc.

• Most OS code abstracts the device details
  – E.g., file system code is written on top of a generic block interface