Devices, Input, and Output

Goals of Kernel I/O routines

- Character code independence
- Device independence
- Efficiency
- Uniform treatment of devices

Character Code Independence

- Kernel I/O subsystem must translate character codes from various devices to uniform internal representation
  - Example: end-of-line can be <NL> (\n), <CR> (\r), <CR><NL> (\r\n), . . .
- Kernel does this right after characters arrive in memory but before they are given to process, or before they are written to the device
  - So programmer need not worry about this
- Internal codes vary; examples:
  - ASCII
  - UNICODE-16, UNICODE-32 (supersets of ASCII)
  - EBCDIC

Device Independence

- Process should not depend on any particular device
  - It *may* depend on a type of device, though
- Operating system should be free to assign any device of right type as appropriate
- Example: process should be able to say “lpr” to print, not lpr0” to print
  - First refers to any printer, second to printer 0
- As far as possible, programs should be independent of type of device

Device Independence

- Sometimes different types of devices require different interfaces
- Linux: reading, writing use same system calls for all devices
- But different types have other, different system calls
  - General interface: *ioctl*(2) with different arguments for different device types
  - Example: for terminal, ioctl(td, FIONREAD, &argp) puts number of bytes in input buffer associated with terminal device with file descriptor fd
  - Example: for network device, ioctl(nd, SIOCATMARK, &torf) returns true if next input from network device with file descriptor fd is marked urgent

Other

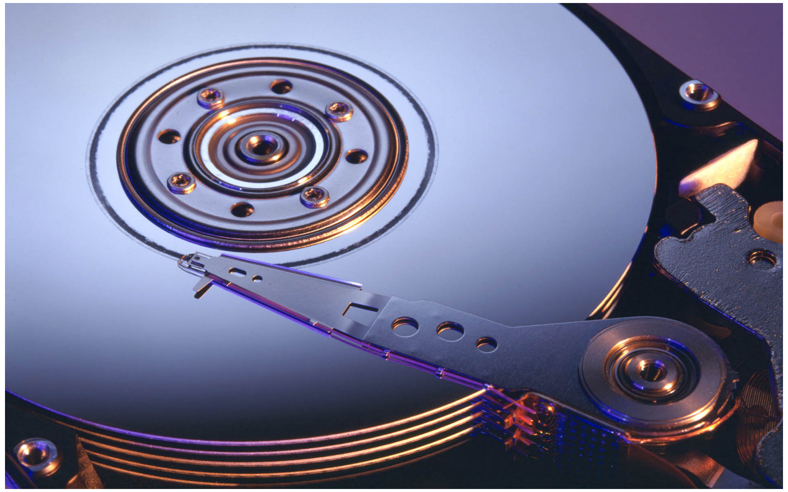
- Efficiency: I/O devices often a bottleneck, so this should be minimized
- Uniform treatment of devices: keep device handling simple, error free for processes
  - May be difficult in practice to handle all devices alike

Device Hardware

- Drums (legacy devices)
- Disks
- SSDs
- Tapes
- CDs
- DVDs
- Communication lines

Disks and Drums

- Disks made up of platters stacked and a spindle in the middle:



- Arms are over each platter, but image shows only top one

Image from: https://allhdwallpapers.com/wp-content/uploads/2016/07/Hard-Disk-Drive-1.jpg

Disks and Drums

- Platters divided into rings called tracks
- Cylinder: same track on all disks in the pack
  - Think of chopping out one track on each platter by cutting straight down
  - Relevant because the arm moves all the heads over the tracks with the same number in the different platters
- Operating system needs to know how disk formatted
  - Number of sectors per track
  - Number of bytes per sector
- Zone is collection of sectors; all have the same physical size
  - So more sectors per zone on the inner tracks

Data Transfer

- Data transferred in blocks
  - Typical block size is 1024 to 8096 bytes
  - Data *always* transferred in multiples of a block
- Rotational latency depends on RPM
  - 15,000 rpm has 2ms average rotational latency
  - 7200 rpm has 4.16ms average rotational latency
- Data transfer rates depend on location
- Higher for outer tracks where there are more sectors per rotation
  - 7200 rpm has average sustained disk to buffer rate up to 1030 Mbits/sec

Sectors

A sector contains:

- data
- a bit indicating whether or not the sector is usable
- error checking information
- (possibly) the sector number

Some disks contain a bad sector map or bad block list, showing which sectors are not usable

Formatting a Disk

- This creates a file system that the operating system can interact with
  - Any bad sectors found during this are marked as unusable
  - Critical information often duplicated, for example using RAID disks for data
  - Errors may be recurring or transient
- If enough sectors are unusable, the operating system can reject the disk

Accessing a Disk

- To access data, operating system seeks for the data by positioning the disk arm on the track on which the data sits
- Three latencies (delays) to consider:

1. *seek latency*: how long does it take the heads to get to the desired cylinder?
2. *rotational latency*: once the heads are over the right track, how long does it take the right sector to rotate under the heads?
3. *transfer latency*: once the heads are over the right sector, how long does it take to transfer data to or from the disk?

Floppies and Drums

- Floppy: one platter; rotates slower than other disks
  - Size 3.5”, capacity ranging from 360KB to 720KB to 1.44MB (last was most popular after it came out)
  - Hole in one corner with a tab which, if moved to block the hole, enabled writing
  - Also came in 8”, 5.25”, and other sizes
- Drum: cylinder divided into circular tracks
  - Like a disk, but with 1 head per track
  - No seek latency
  - Not used now

Solid State Drive (SSDs)

- Stores data persistently, like a disk, but without spinning platters
- SSD contains controller that does (among other things):
  - Bad block mapping
  - Read, write caching
  - Error handling
- Usually smaller than disk drives
- Read latency much lower than hard disk drives

Tapes

- Used primarily for archiving data, data transfer
- Very portable among different operating systems, architectures
- Characteristics:
  - Access is sequential; to read something in the middle of the tape, you have to go past everything before it on the tape
  - Usually in cassettes now; standards dictate how many tape heads there are
    - LTO Ultrium can handle 8, 16, or 32 heads
  - Capacity varies from 100 GB to 18 TB
  - Seek times range from 10 to 100 seconds
  - Data transfer rates can exceed those of hard drives

Compact Disks (CDs)

- Physical: 1.2mm thick, 12 cm diameter disk of polycarbonate plastic
- Data stored in indentations on surface (“pits”, separated by “lands”) in a spiral track. Read using a laser through the bottom; the pits vary the intensity of the light reflected, which is translated into bits.
- Capacity is usually 700 MB data per side
- Track broken into *frames* (24 bytes) grouped into *sectors* of 98 frames
    - A data CD can have 2048 or 2336 bytes per sector.
    - Sectors grouped into *tracks* and a CD can have up to 99 tracks.
- Data transfer rates: from 150KB/sec to 4.8MB/sec

Digital Video Disks (DVDs)

- Also Digital Versatile Disk
- Like CD, data stored on indentation on surface, but laser is narrower
- Holds ~ 4.7GB (single layer) or ~8.5GB (double layer)
- Each sector has 2418 bytes; 2048 bytes are user data
- Data transfer rates: from 1.4mB/sec to 33.2MB/sec

Communication Lines

- Simplex lines: transmit data in one direction
- Half-duplex lines: transmit data in either direction, but only one direction at a time
- Duplex lines: transmit data in both directions simultaneously
- Synchronous transmissions: sender, receiver share common clock and know when to sample communication line for transmission
- Protocols are conventions for formatting information, interpreting messages, etc.

Protocols

- Each transmission is preceded by header containing prearranged patterns of bits enabling the receiver to adjust its clock to match that of the sender
- Transmissions consist of frames (packets) each of which is preceded by header
- Header pattern cannot appear in message
  - If transmission is bit-oriented, put an extra bit in; it will be stripped at the receiving end (called *bit stuffing*)
  - If transmission is character-oriented, add a byte (the *escape character*) instead of a bit

Example

- BISYNC protocol (used to communicate between mainframes)

  <SYN><SYN><SOH>header<STX>text<ETX>checksum

- Escape character is <DLE>
- If <ETX> character appears in text, put a <DLE> in front of it
  - Then receiver will strip out <DLE> and treat <ETX> as an ordinary character
- Notes:
  - <SYN> is SYNchronize, ASCII character 22
  - <SOH> is Start Of Header, ASCII character 1
  - <STX> is Start of TeXt, ASCII character 2
  - <ETX> is End of TeXt, ASCII character 3
  - <DLE> is Data Link Escape, ASCII character 16

Device Interface

- Lowest level of interaction between machine, I/O device
  - Device driver sits right above it
- Interface mechanism is *device register* used for
  - Transferring status information from device to CPU
  - Transferring instructions from CPU to device
  - Transferring data between CPU and device

Device Controller

- Sits between device, CPU
  - Usually on the device, but could be on the machine
- Monitors device status, other things
- Accepts instructions from the CPU and executes them
- Handles accepting (writing) or returning (reading) data

Channels

- Subsidiary CPUs that use a different machine language
  - Instructions called *commands*
  - Sequences of commands are *channel programs*
- Typically use direct memory access (DMA)
- *Scatter-gather* refers to the ability to scatter input data to or gather output data from many locations

Device Drivers

- Three functions
  - Try to put regular structure on the parts of the operating system that interact with devices
  - Provide standard interface to rest of kernel
  - Serve both the rest of operating system and devices

Organization of a Device Driver

- Two parts
  - Upper: interacts with rest of operating system
  - Lower: interacts with device itself
- Upper part:
  - Accepts requests from operating system, eg., memory manager asks operating system to write out page
  - Transforms this into entries in a pending work list
- Lower part:
  - Invoked on interrupt, or when something added to pending work list
  - Disable interrupts from the same device
  - Do the work on the work list
  - Re-enable interrupts

Example: Clock Device Driver

- Two types of clock devices: line clock, programmable clock
- Line clock: generates interrupt every tick
  - May have register that counts ticks since last reset
  - May have a backup battery
  - May have a register counting ticks missed if CPU doesn't service the interrupts
- Programmable clock: has a count register that can be set by software
  - Decrement by 1 every time interval (ms, tick, etc.)
  - When count register is 0, generates an interrupt

On Clock Interrupt . . .

- System software time structure incremented
- If clock is used for scheduling
  - Decrement the remaining time field of the current process
  - If time field is 0, invoke scheduler
- Do any accounting
- If no programmable clock
  - Decrement counter for next alarm
  - If this counter is 0, any kernel routine waiting for an alarm is invoked
- If current process being profiled, figure out where it is (look at its PC) and update the profiling counters

Example: Disk Device Driver

- Must provide an illusion of a linear array of sectors that are numbered like elements of an array
- Sector s on track t in cylinder c is numbered

$$a = ((c \times (\#tracks/cylinder) + t) \times (\#sectors/track) + s$$

  so rather than referring to (c, t, s), kernel can refer to a
- Also must reduce effect of latencies of accessing disk
  - Overlap I/O and computation
  - Arrange large objects only one seek is needed to read/write them
  - Order outstanding disk requests

Ordering Disk Requests: Assumptions

- Only 1 disk drive
- All I/O requests are for single equal-size blocks
- Requested blocks distributed over disk
- Disk has only 1 moveable arm with all heads on it
- Seek latency is linear in the number of tracks crossed
  - Ignores disk controller using sectors from tracks at end of disk to replace bad sectors
- Disk controller does not introduce appreciable delays
- Read, write requests are equally fast

Key Points for Evaluating Disk Access Policies

- How long must requests wait as a function of load
  - Frequency of requests, measured in requests per time unit
- Mean, variance of waiting time for each request
  - Low mean, high variance means some requests will take a long time

Disk Access Policies

- First cone, first serve (FCFS)
- Pickup
- Shortest seek time first (SSF, SSTF)
- SCAN
- N-Stop SCAN
- C-SCAN