I/O (Lecture 15)

Architecture

- Bus-style architecture
- PCIE Architecture

Communication methods CPU/I/O

1. Explicit I/O instructions
    1. `inb` , `outb` (b stands for bytes), `inw` `outw` ..
    2. `inb(0x1F7, 0x20)` 0x1F7 -> port , 0x20 -> command
2. Memory-mapped I/O
    1. special memory addresses connected to device registers
3. Direct memory access (DMA)
    1. device can transfer data directly to/from memory without CPU
    2. CPU will run for other tasks during the transfer

I/O synchronization methods

1. Busy waiting
2. Polling
3. Interrupts

Device Driver

- the middle layer between hardware and the Os kernel

More CPU Stuff (Lecture 16)

Memory mapping (**mmap**)

- Direct file access through memory operations
- mmap: moving at least one page at a time

Context switching (switching from one process to another)

- switching the view of the memory
- switching the registers

User-level threading

- kernel only see as one process
- custom library to do these

Two methods:

- cooperative: `yeild`
- preemptive: uses signal/timers

Disk (Lecture 17)

Physical strcuture of a disk

- platters
- spindle
- head
- acutator arm

Geometry

- Track
- Sector
- Cylinder

Performance Factors

- seek time
- rotation delay
- transfer time

optimizations:

- read-ahead caching
- write caching
- track skewing
- Disk scheduling

File System (18-19)

What is a file system:

- persistence for data

- maps human-friendly names to disk blocks

operations:

- read/write/delete, ...

Goal: have as few disk accesses as possible with minimal overhead

Implementation

- Files -> inodes (metadata, data)
- Directory (filename -> inodes)
  - `/` root
  - `.` current dir
  - `..` parent dir

How to Allocate blocks?

- contiguous
- linked
- indexed allocation (multi-level indexed allocation)

FFS

- cylinder groups
- blocks size increase

Bitmap allocation

Crash recovery (Lecture 20)

Key: how do we maintain consistency?

Approach

1. ad-hoc (fsck)
2. Copy-on-write
3. journaling (redo/undo)
    1. txbegin/txend

Hard links/soft links

NFS (Lecture 21)

RPC: remote procedural call

NFS architecture
VFS

important property: **stateless**!

File handler (FS id, i-node, gen #)

How the semantics in file operations in NFS different from standard unix file operations semantics

"close-to-open consistency"

Stack smashing (Lec 22)

Buffer overflow

Nop

Defenses:

- stack canary
- address space layout randomization
- Write XOR execute
    - return-oriented programming

Trusting Trust (Lec 23)

Unix Security (Lec 24)

Setuid program

Summary (Lec 25)

- loading and execxuting programs
- power-up to terminal