CSS503 A

Lecture #20

2019-06-05

Recap, Final Exam Review

Recap

- Operating Systems: kernel vs userland
- processes - essentially: running program
  - kernel view: data structure
  - system calls: user request for kernel service
  - process creation & termination
  - inter-process communication
    - pipes
- threads: concurrent paths of execution within single process
  - deadlocks & race conditions
  - mutual exclusion & signaling
  - message passing (abstraction)
- file systems
  - data structure on top of block devices
  - disk allocation algorithms
    - contiguous
    - linked list
    - File Allocation Table
    - indexed

Recap (cont.)

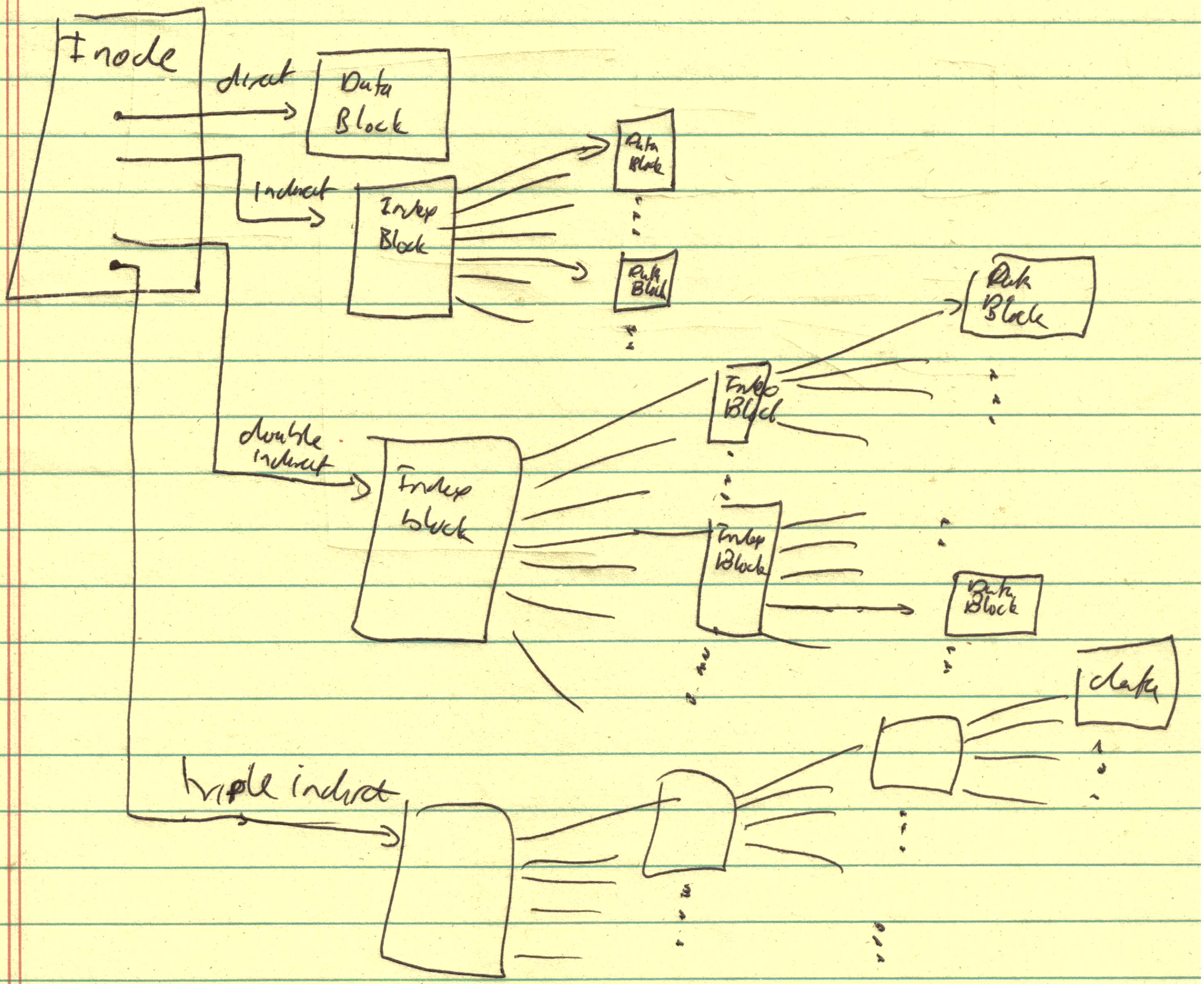
- networking
  - layered protocols: OSI 7-layer model
  - Ethernet: data link layer
  - Internet protocol
  - TCP & UDP
  - applications

    } basically, 4 layers
  - distributed systems
    - client-server
    - datacenter-wide operating system:
      manage resources of entire datacenter
  - remote procedure call: abstraction
    - not entirely dissimilar to message passing

Final Exam

- 4 questions @ 8 points each
    - File AllocationTable
    - indexed filesystem
    - networking
    - virtual memory

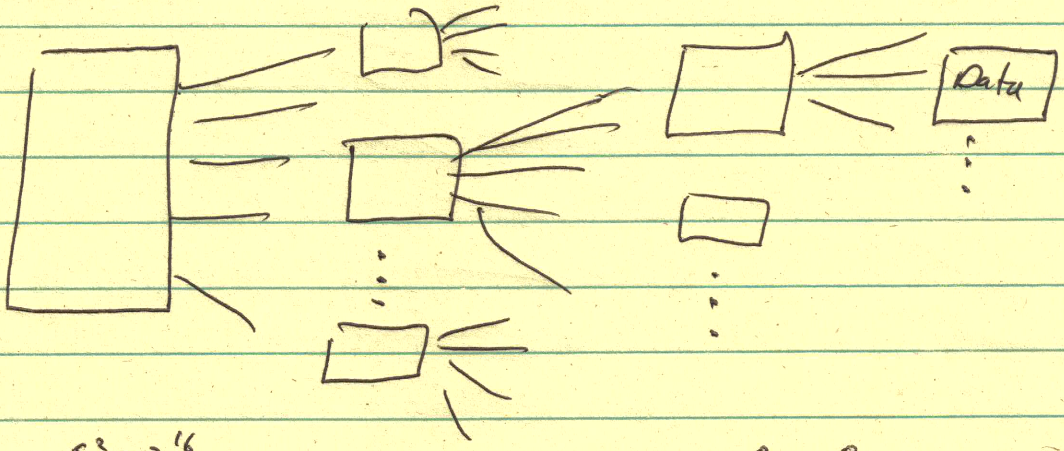
Direct vs Indirect Indexing



Indirect Indexing

- indirect index: trie-like data structure (aka radix tree)
    - squat, with high fan-out

- suppose index block holds 256 (2^8) (radix) block pointers
    - each level is digit in base 256



$256^2 = 2^{16}$
$= 64k$

$256^1 = 2^8$

$256^0 = 2^0 = 1$

triple indirection

Indirect Indexing (cont.)

$$\text{level1 offset} = \text{block\_num} \div 256^2$$
$$= \text{block\_num} >> 16$$

$$\text{level 2 offset} = (\text{block\_num} \div 256) \bmod 256$$
$$= (\text{block\_num} >> 8)\ \&\ \substack{255 \\ \text{0xff}}$$

$$\text{level1 offset} = \text{block\_num} \bmod 256$$
$$= \text{blocknum}\ \&\ \text{0xff}$$

$$\left\{\begin{aligned} \text{level2\_block} &= \text{get\_block}(\text{level1}[\text{offset}_1]) \\ \text{level3\_block} &= \text{get\_block}(\text{level 2}[\text{offset}_2]) \\ \text{data} &= \text{get\_block}(\text{level\_3}[\text{offset}_3]) \end{aligned}\right.$$

Combination Direct & Indirect Indexing

Suppose

- index block holds M block addresses (e.g. 64, 128, 256)
- d direct pointers (e.g 3, 5, 10)
- i indirect pointers (e.g 3, 4, 5)
- i2 double indirect pointers (e.g 2, 3, 4)
- i3 triple indirect pointers (e.g 1, 2, 3)

- to find block address for block number

```
if blocknum < d
    return direct pointer blocknum

blocknum -= d
if blocknum < $i * M$
    index block = indirect pointer $\frac{\text{blocknum}}{M}$
    return index block [ blocknum mod M ]

blocknum -= $i * M$
```

Combination Direct & Indirect Indexing (cont.)

```
if blocknum < $i2 * m^2$
    indexblock1 = doubleindirect-ptr $\frac{blocknum}{m^2}$
    indexblock2 = indexblock1 ( $\frac{blocknum}{m}$ mod m )
    return indexblock2 ( blocknum mod m )

blocknum -= $i2 * m^2$

indexblock1 = triple-indirect ( $\frac{blocknum}{m^3}$ )
indexblock2 = indexblock1 ( $\frac{blocknum}{m^2}$ mod m )
indexblock3 = indexblock2 ( $\frac{blocknum}{m}$ mod m )
return indexblock3 ( blocknum mod m )
```

Bitwise Operations

- multiplication/division/modulus by powers of 2
  $\Rightarrow$ more efficient to use shift & mask operations

- mask: & with 1 - bits to keep
  0 - bits to clear

e.g. $x \bmod 2^k \equiv x \,\&\, (2^k - 1)$

- $x * 2^k = x << k$

- $x / 2^k = x >> k$

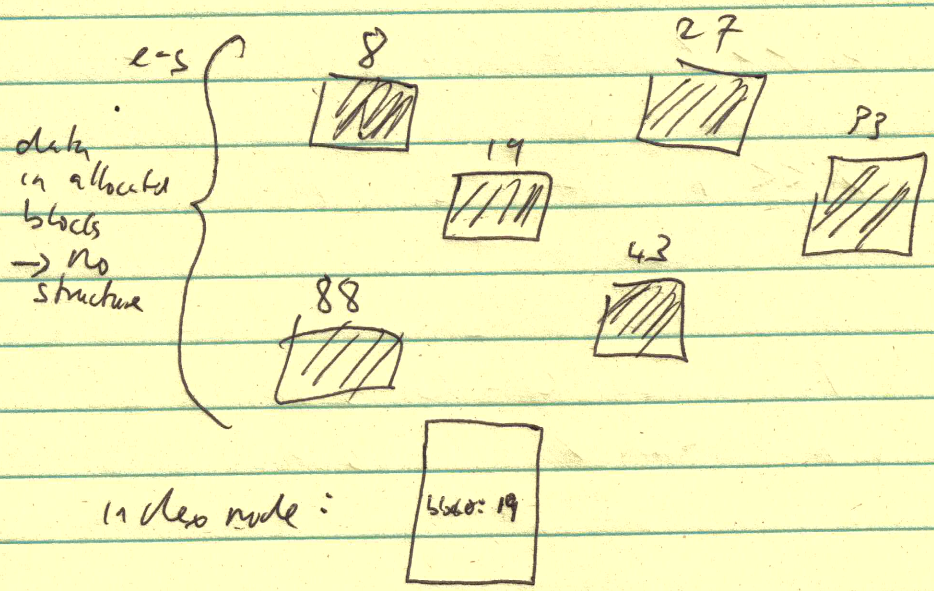
Boolean identities

$x \,\&\, 0 = 0$ clear $\quad x \mid 0 = x$ keep

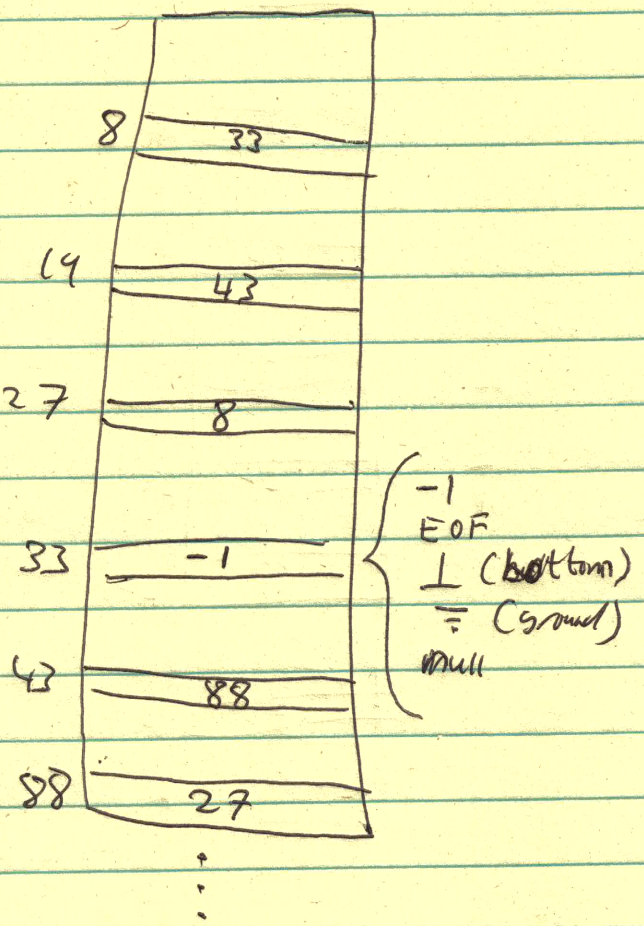
$x \,\&\, 1 = x$ keep $\quad x \mid 1 = 1$ set

File Allocation Table

- blocks contain data only
- index node contains address of block 0 (1st block)
- global FAT is an array of block addresses
  - block addresses are integers
  - array indexes are integers
  - ⇒ there is a 1:1 correspondence between block addresses & FAT array indexes



File Allocation Table (cont.)



19 → 43 → 88 → 27 → 8 → 33

- array value is array index of next node in list

- array value is also block address of next block in file