Review

File Systems

1. Key abstractions
   - File
   - Filename (don't need to remember sector number, just name)
   - Directory Tree
2. Access Rights
   - Permissions - Read/Write/Execute for user/group/other
   - Access Control Lists allow for more specific permissions
   - If you don't have execute permissions on a directory, you can't pass through it and modify what's inside
3. Developer's Interface
   - Position/Cursor-based
     - fd = open("file.txt",...);
     - read(fd, ...);
     - write(fd, ...);
     - lseek(fd, ...);
     - close(fd, ...);
   - Memory-mapped - treat a file as a chunk of memory
     - fd = open("file.txt",...);
     - buf = mmap(..., fd, ...);
     - munmap(buf, ...);
     - close(fd, ...);
4. Allocation strategies
   - Platter, track, sector (block)
   - Figures of Merit
     - Simple and fast creation
     - Flexible size
     - Efficient use of space
     - Fast sequential access
     - Fast random access
5. File Allocation Table
   - File is represented as a linked-list of blocks
   - FAT: Indexed by block number
     - Busy: True or False
     - Next: Next block (-1 if end of chain)
   - Directory Table Format (directories treated as files)
     - Filename, starting block, metadata
   - Pros:
     - Easy to create a new file
     - Good for removable storage (copying data; sequential)
     - Spatial efficiency
   - Cons:
     - Bad for random access (tracing pointers)
6. Extended File System
   - Inode structure
     - 12 direct pointers to data
     - 1 indirect pointer to table to data
     - 1 2x indirect pointer
     - 1 3x indirect pointer
   - Directories point to other inode structures
   - Pros:

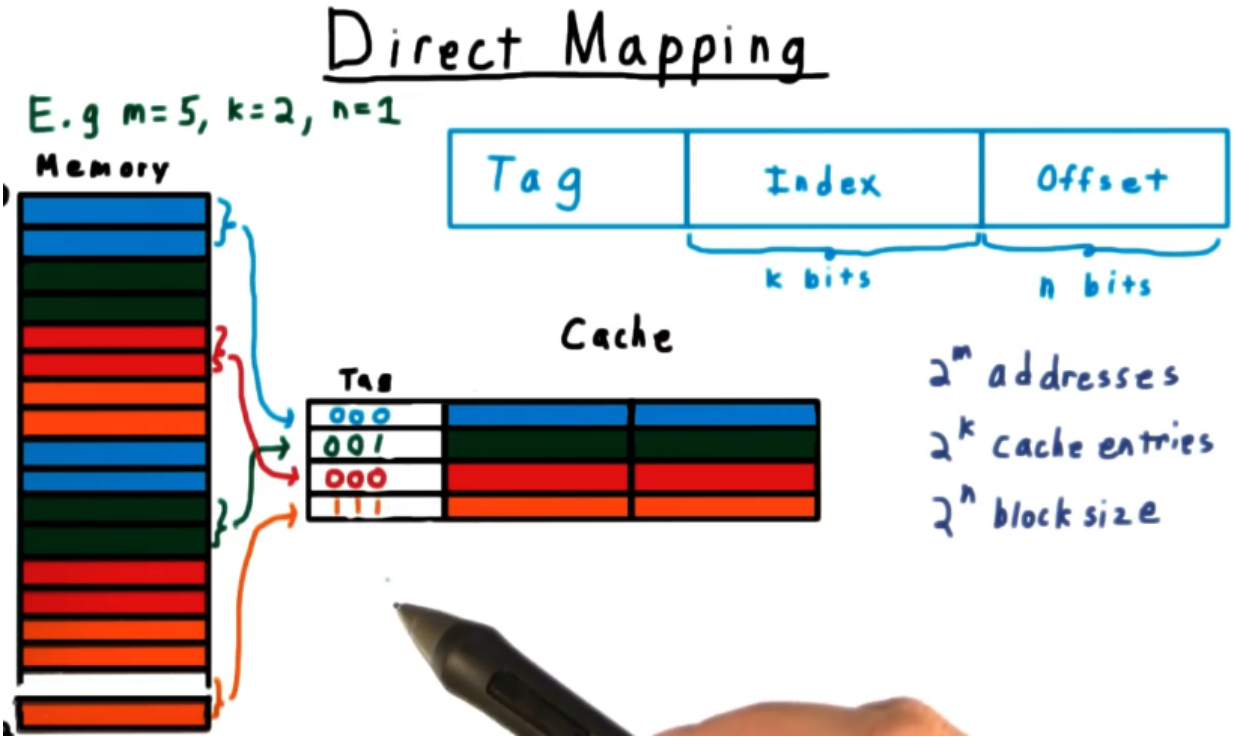
    - Much better random access due to tree structure vs FAT
  - Cons:
    - Slightly slower access than FAT due to levels of indirection
7. Unified Buffer Cache
  - Cache in main memory to avoid having to go to disk as often
  - Dirty bits allow for the delay of writeback for more opportune times
    - If a file has a short lifetime, writeback might not be necessary
  - fsync/msync to flush cache to disk
8. Journaling
  - Sudden power failure means anything in memory will be lost
  - Sequential access thousands of time faster than random
  - Store changes on disk sequentially (journal) then write back later
    - Two writes; more time spent writing
9. Direct Memory Access
  - Disk can talk to CPU directly instead of only through memory

Optimization	Reduces Writes	Improves Performance	Improves Recovery
Buffer Cache	X	X	
Journaling		X	X
DMA		X	

Memory Systems

1. Naive Memory Model
  - Memory
    - Fast
    - Random access
    - Temporary
  - Disk
    - Slow
    - Sequential access
    - Durable
2. Memory Heirarchy (smaller/faster at top of hierarchy)
  - Registers
  - L1 Cache
    - Instruction/data cache
  - L2 Cache
  - L3 Cache
  - Main memory
  - Disk
3. Locality and Cache Blocks
  - Locality: Assume an application will need data near what it already accessed
  - Temporal: Keep data in cache right after it's used
  - Spatial: Instead of a single memory address, fetch entire block
    - 64 bytes, typically
4. Direct Mapping
  - 2^m addresses
  - 2^k cache entries
  - 2^n block size
  - Address: Tag, k bits for index, n bits for offset

Tag	Index	Offset
	k bits	n bits



Cache Design

5. Set Associative Mapping
   - In a direct mapping, the address is only associated with one location in the cache
   - In a set-associative mapping, we map an address to multiple locations in the cache
     - Must check two tags to see if there's a cache hit
     - Less likely to encounter the problem of constantly evicting a line
   - Introduces replacement policies; typically use least recently used

Tag	Index	Offset
	k-j bits	n bits

6. Fully Associative Mapping
   - The address can go in any location in the cache
   - There is no index anymore, all tag
     - This means the hardware must check tags for every entry
     - Hardware considerations: Typically 64-256 entries

Tag	Offset
	n bits

7. Write Policy
   - Hit:

    - Write-through: Write to the cache and memory
    - Write-back: Only write to the cache
  - Miss:
    - Write-allocate: Write to cache and memory
    - No-write-allocate: Only write to memory
8. Virtual Address Abstraction
   - Each process requires the ability to use the entire system's memory
   - However, processes can't access another processes memory
   - Solution: Each process receives virtual memory addresses
     - The OS maps these virtual addresses to physical locations
9. Address Translation
   - Operating system must translate a virtual page number to a physical frame number
   - Virtual pages are typically 4KB
10. Page Table Implementation
    - Typically a hierarchical implementation to minimize unused addresses
    - 32-bit VA
    - 32-bit PA
    - 4K pages -> 12-bit offset
    - 10 bits to index into page directory
    - 10 bits to index into page table
    - 12 bits to locate the page in the page table
    - Top-level page table is always in memory (not paged)
11. Accelerating Address Translation
    - Translation-lookaside buffer caches VA->PA translations
    - When a context switch occurs, the mappings in the TLB are no longer valid. You can either...
      - Flush TLB on context switch (costly)
      - Store and check address space ID
12. Page Table Entries
    - Access control: read/write/execute
    - Valid/present: is the page in memory?
    - Dirty: Has the page been written to? If not, don't write to disk on eviction
    - Control caching
13. Page Fault
    - What happens when a process accesses a virtual address that isn't mapped to physical memory?
    - OS page fault handler engages
      - Check if request is valid; if not, seg fault
      - Find a page in memory to store the data
      - If there are no free pages, must evict one (page replacement)
      - Load page into memory from disk
      - Update page table and other data structures
      - Restart the process

```
def load(virtual_addr):
    try:
        physical_addr = translate_addr(virtual_addr)
    except page_fault:
        handle_page_fault(virtual_addr)
        return load(virtual_addr)

    if is_in_cache(physical_addr):
        return read_from_cache(physical_addr)
    else:
        return read_from_memory(physical_addr)


def translate_addr(virtual_addr):
    if is_in_tlb(virtual_addr):
        return read_from_tlb(virtual_addr)
    elif is_access_violation(virtual_addr):
        raise access_violation
    elif is_present(virtual_addr):
        return read_from_page_table(virtual_addr)
    else:
        raise page_fault

```

Memory-Management Pseudo-Code

14. Virtually Indexed, Physically Tagged Caches
    - Only the page offset bits are important for accessing the cache
    - This means we can use the virtual page number to access the TLB and the page offset to access the data cache simultaneously
      – Provides a speedup

Multithreaded Programming

1. Process-Thread Relationship
    - Each process has a stack, heap, globals, constants, and code
    - Each thread has its own stack, but shares everything else
2. Joinable and Detached Threads
    - Detached Threads Termination Cases
      – Any thread makes an exit call, or main reaches the end of its code
      – Our thread returns or calls pthread_exit
    - Joinable threads won't terminate until another thread joins them
      – pthreads are joinable by default
3. Thread Patterns
    - Team model: Pool of worker threads fetching requests from queue
    - Dispatcher model: Dispatcher thread passes requests to available worker
    - Pipeline model: Break work into subtasks

4. Producer-Consumer Pattern
    - One thread producers work and adds it to a shared buffer
    - Another thread removes work from the queue and completes it
5. Mutex Lock
    - Atomic structure for ensuring threads don't simultaneously access the same resource
    - Supported by hardware
6. Mutex vs Synchronization
    - Mutex: Prevent two threads from accessing the same memory
    - Synchronization: Controlling where two threads are in their execution
7. Conditional Wait Variables
    - Prevents one thread from spinning until another thread finishes
    - Releases the mutex while waiting for a signal
    - Must spin on the condition, not if, to truly ensure the condition is met

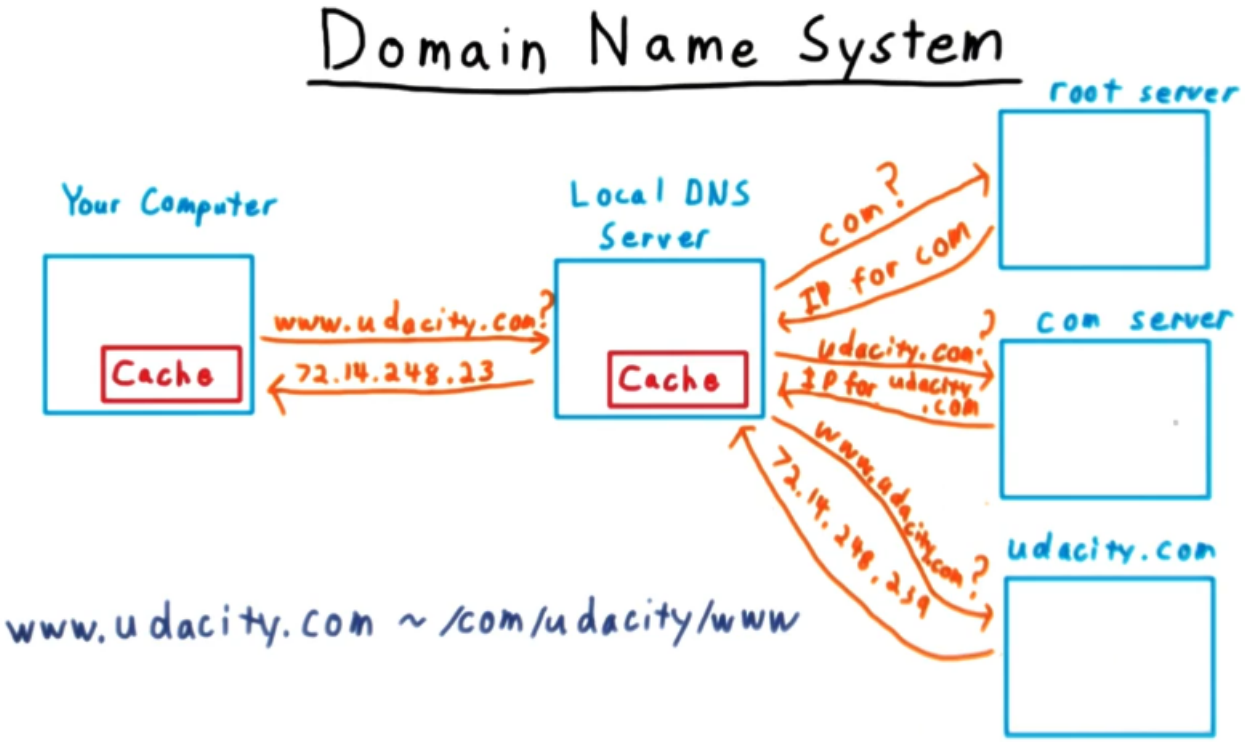
Networking

1. Interconnection Layers
    - Application: Application-specific protocols (HTTP, SMTP, etc.)
    - Transport: Detecting when data is missing, out of order, etc.
    - Network: End-to-end communication on the internet (IP addresses)
    - Link: Communication between peers on a local network (peer-to-peer)
    - Physical: Hardware that creates packets (NIC, modem, Ethernet, etc.)
2. Physical
    - NIC: Direct Memory Access device; doesn't have to go through CPU
    - Only interrupts CPU when data is done being sent or received
3. Link
    - Break large chunks of data into manageable chunks (packets)
    - NICs are identified by a unique 48-bit MAC address (media access control)
    - Every machine on a LAN will receive a message, only the ones that have MAC addresses matching the intended recipient will do anything with it
    - Collision avoidance: Two machines sending messages simultaneously
    - Collision avoidance approaches
        - CSMA/CD: Carrier Sense Multiple Access with Collision Detection
        - Token Ring/Bus
        - Switched Ethernet: Routes frames to destination MAC address; all frames go through the switch

Strategy	Efficient (light loads)	Resilient- Heavy Loads	Fair
CSMA/CD	X		
Token Ring		X	X
Ethernet	X	X	X

4. Network
    - IP address: 32-bit address
    - CIDR Notation: IP/# of bits for network ID
        - v.w.x.y/z -> z represents number of bits that are fixed
        - MIT: 18.0.0.0/8 -> 18.x.y.z are MIT's (2^24)
        - GT: 130.207.0.0/16 -> 130.207.x.y are GT's (2^16)
    - IPv6 uses 128-bit IP addresses; IPv4 uses 32-bit
    - Highest in range is broadcast: send to entire subnet
    - Lowest in range refers to subnet as a whole
    - LANS and NAT
        - Local network has a set of private IPs that router knows about

  - NAT: Network Address Translation
  - ARP: Address Resolution Protocol
- Internet Routing
  - Routing table: Translates IP address to next hop
  - Go through many hops to reach destination
  - netstat -nr
- Autonomous Systems
  - A bunch of discrete nodes that are interconnected
  - BGP: Border Gateway Protocol
- Domain Name System



Domain Name System

5. Transport
   - Ports: Specify which process the frame is intended for
     - Allows multiple processes to receive data simultaneously
   - Transport layer is only active at the end-points of the graph
   - NAT sends the response to whatever port sent it (same for IP)
   - Transmission Control Protocol Wishlist
     - Reliability
     - Cope with out-of-order delivery
     - Flow/congestion control
   - TCP sends acks and tracks how much data has been sent and received
     - Will only send so many packets without receiving an ack
   - User Datagram Protocol
     - Ignores the reliability and out-of-order protection of TCP
     - VoIP: User prefers degradation in quality over delay
     - Streaming: Don't mind occasional packet loss

Contents	UDP	TCP
Port	X	X
Window size		X
Seq/Ack		X

6. Putting it all together
    - Application uses DNS to convert host name to IP address
    - Routing table points system to modem (can acquire through ARP request)
    - Router swaps system port/IP for modem port IP (NAT)
    - Modem wraps packet in link layer frame and passes to WAN
    - To respond, end user swaps src/dest port and IP and sends packet