CS 112/212: Final Review

March 11, 2022

Topics

Covered in Midterm Review

- **Processes and Threads**
- **Virtual Memory**
- **Concurrency**
- **Synchronization**
- **Linking**

To be covered Today

- **Memory Allocation**
- **Device I/O**
- **File Systems**
- **Security**
- **Virtual Machines**

Memory Allocation

- **Dynamically give programs arbitrary size chucks of memory**
- **The core fight: minimize fragmentation**
  - Allocation have different sizes and life-times leaving “holes” in the memory space
  - Various allocation policies to try to mitigate
- **Can use garbage collection in languages that control pointers**

Ways for OS (drivers) to do IO

- **Memory-mapped device registers**
  - Regular memory read/write interface except access go directly to a device's registers
- **Memory-mapped device memory**
  - Regular memory read/write interface except access go directly to a device's internal memory
- **Special instructions (e.g. inb, outb)**
  - Communicates with devices using specified "port" numbers
- **DMA (Direct Memory Access)**
  - CPU offloads read/write of main memory to device/DMA engine

File systems

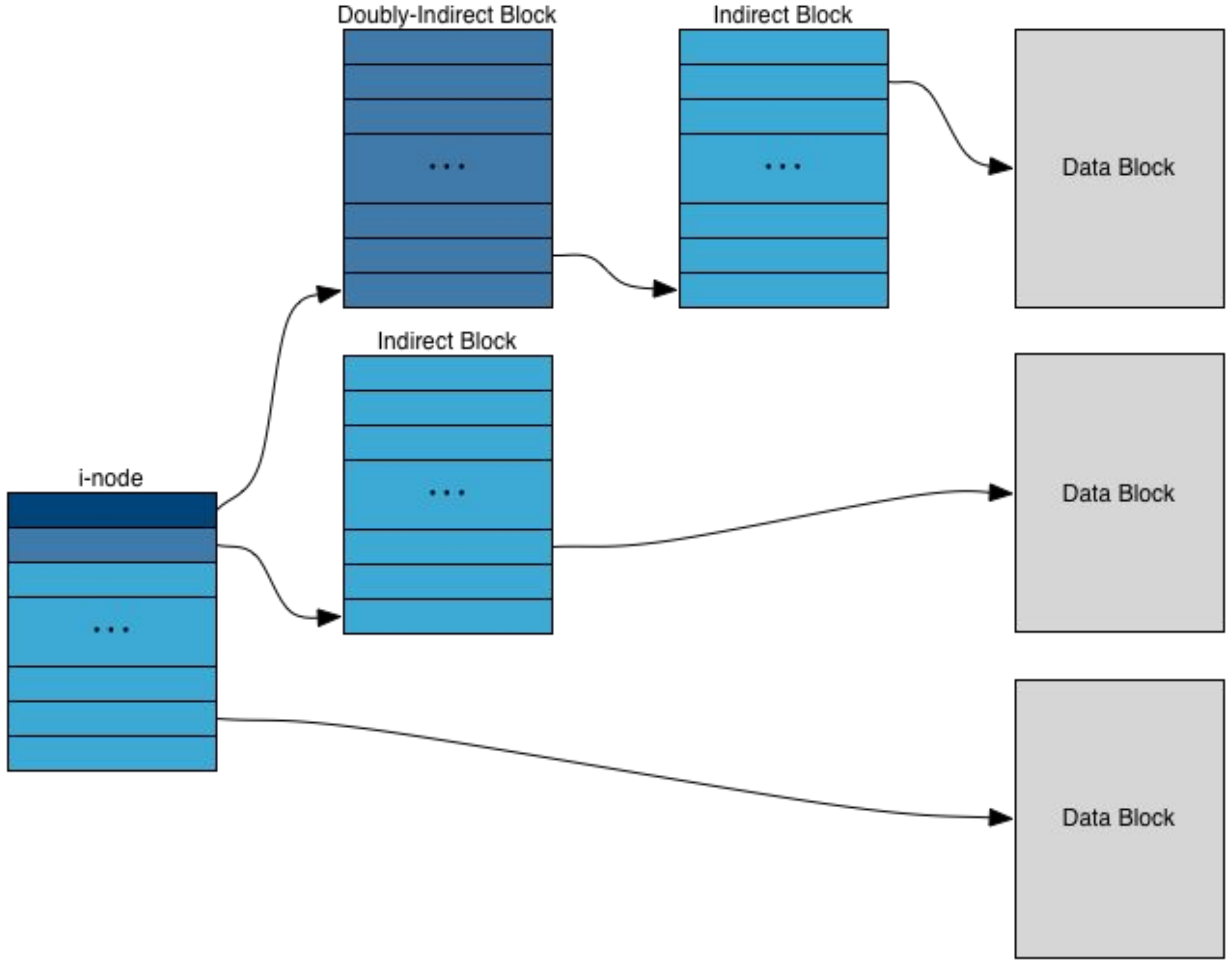
- **Need a way to persist and organize data between restarts**
- **Associates names with bytes on disk**
  - Want an organization and naming that humans can remember
- **Most file systems designed around disks**
  - Optimized for fast sequential access and slow random access
- **Need to handle unexpected crashes**

File systems on Disk

- **How do you track the blocks associated with a file?**
- **Contiguous allocation “extent-based”**
- **Linked files**
  - Each block contains the location of the next block
- **FAT (File Allocation Table)**
  - Like linked files but keep link information for all files in one (or two) blocks
- **Indexed Files**
  - Keep an index for each file (inode)

Muti-level indexed files

- **Files divided into blocks of 4 Kbytes**
- **Blocks of each file managed with multi-level arrays of block pointers**
- **File descriptor (i-node) = 14 block pointers, initially 0 ("no block")**
  - First 12 point to data blocks (direct blocks)
  - Next entry points to an indirect block (contains 1024 4-byte block pointers)
    - Last entry points to a doubly-indirect block
- **Maximum file length is fixed, but large**
- **Indirect blocks aren't allocated until needed**



File Naming and Directories

- Directory contains a mapping from name to an inode
- Directories are just files with a specified format
- Multiple directories can contain file names that point to the same inode (hard-links)
- Names can also point to a string that resolves at time of access (soft-links)

Handling Crashes

- **Machine could shut down at literally any point**
- **Need to make sure that the file system is never corrupted**
  - Ok with (some) data loss
  - NOT ok with corruption
- **Possible solution: Fix corruption (fsck)**
  - After crash fsck can be run to try to fix disk corruption and clean up the disk
  - Scans over the entire disk looking for orphaned files, leaked disk blocks, etc
  - Issue: need to make sure that no corruption can occur that is beyond repair

Minimizing Corruption

- **Ordered Updates**
  - Ensure write are permitted back to disk in an order that is recoverable
  - e.g. add the new inode before updating the directory
- **Soft Updates**
  - Update order may create cycles
  - Break cycles by temporarily roll back all changes that created the cycle
- **Journaling**
  - Allow operations the act as though they are atomic
  - Use a write-ahead log to persist the intent; replay the log if there is a crash

Networking

- **Allow two applications on different machines to communicate**
- **OS provides abstraction for communication**
  - Handles packaging, sending, unpacking, and delivering of information
- **TCP implemented by the kernel to provide a “reliable pipe” abstraction over an unreliable network**
- **The user-level interface provided is called a socket**
- **Endpoints are named by an IP-address and 16-bit port**

Network Layering

- **Networking protocols are organized in layers**
- **Application data wrapped in TCP layer**
  - Contains information for implementing reliable delivery
- **TCP packet wrapped in IP packet**
  - Contains information for routing packets between networks
- **IP packet wrapped in link layer protocol (typically ethernet)**
  - Contains information for delivering packets within a network
- **Layers are unwrapped to deliver data to the application**

Networking Implementation

- **mbuf used to store packet data**
  - Packets made up of multiple mbufs
  - mbufs are basically linked-lists of small buffers
- **protosw structure as abstract network protocol interface**
  - Goal: abstract away differences between protocols
  - In C++, might use virtual functions on a generic socket struct
  - Here just put function pointers in protosw structure

Basic Security

- **How do you limit access to resources (files, devices, etc.)?**
- **Access Control Lists**
  - Each “object” has an associated list of who has access
  - OS checks that a user is on the list before granting access to the object

Basic Security Issues

- **setuid: how to allow partial privileges?**
  - e.g. what to allow the user to change their own password in the password file but don't want the allow reading the password file
  - setuid allows a program to run at with the effective permissions of the files owner
- **TOCTOU (Time-of-check, Time-of-use) bug**
  - e.g. first check if you are allowed to execute, then execute
  - Problem: attacker can change the state between the check and the execution

Advanced Security

- **Discretionary Access Control (DAC)**
  - Prevents unauthorized access to resource
  - Does NOT prevent authorized access from leaking information
  - e.g. ACL
- **Mandatory Access Control (MAC)**
  - Prevents both unauthorized access and unauthorized disclosure
  - e.g. stop a infected virus scanner from leaking your data

Mandatory Access Control (MAC)

- **A security level or label is a pair(c,s) where:**
  - c=classification – E.g., 1=unclassified,2=secret,3=topsecret
  - s=category-set – E.g., Nuclear, Crypto
- **(c1,s1) dominates (c2,s2) iff c1≥c2 and s1⊇s2**
- **Subjects and objects are assigned security levels**
- **Prevent leaking classified by checking the dominates relationship**
  - e.g. kill any process that attempts to write to a with security level (c′,s′) if it has already read from a file with security level (c,s) where (c,s) dominates (c′,s′)

LOMAC (Low water Mark Access Control)

- **LOMAC's goal: make MAC more palatable**
- **Concentrates on Integrity**
  - More important goal for many settings
  - E.g., don't want viruses tampering with all your file
- **Security: Low-integrity subjects cannot write to high integrity objects**
- **Subjects are jobs (essentially processes)**
  - Each subject labeled with an integrity number (e.g., 1, 2)
  - Higher numbers mean more integrity

Advanced Security Issue: Side Channels

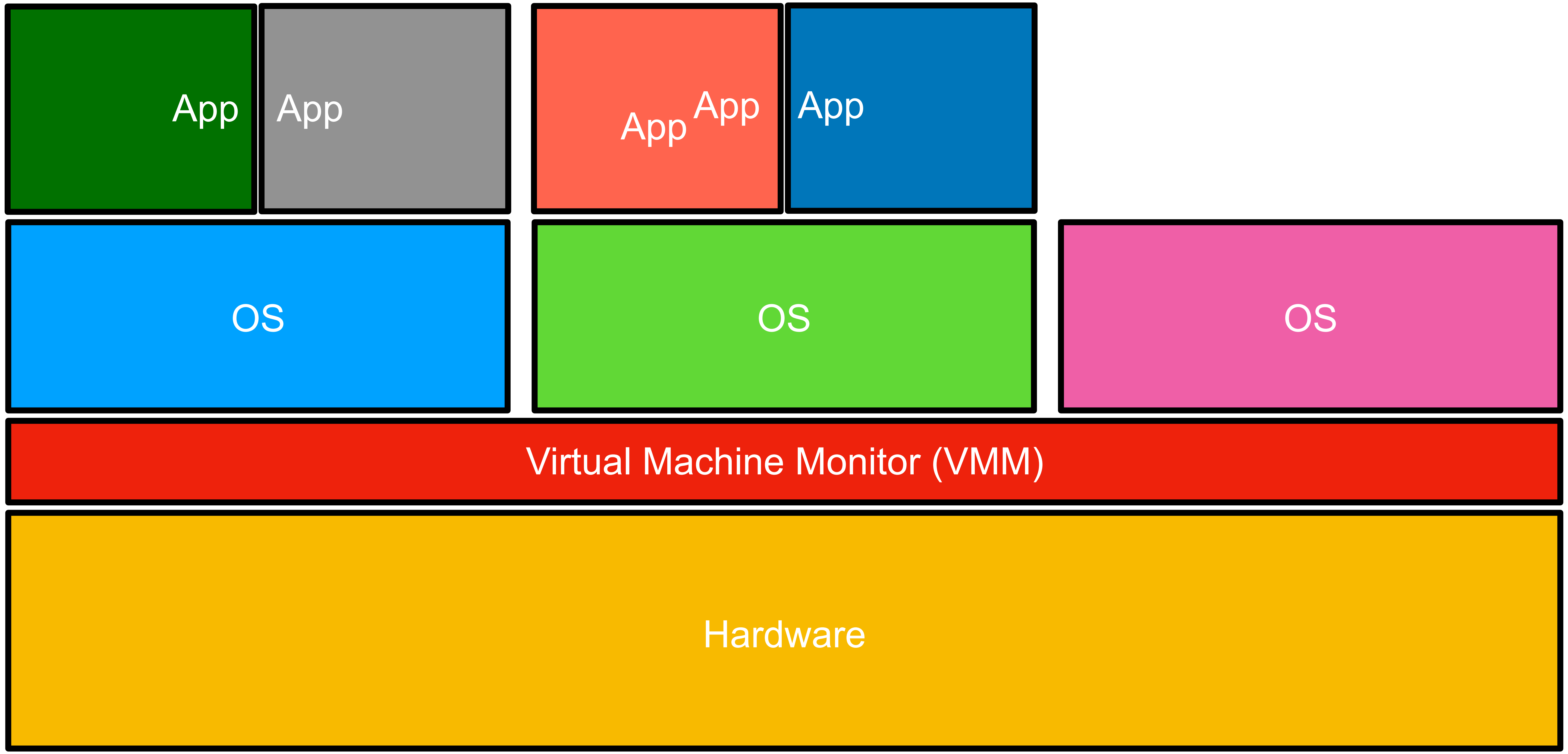
- **Even with access controls process can communicate in an unauthorized manner**
- **Covert storage channels**
  - e.g., high program inherits file descriptor-Can pass 4-bytes of information to low program in file offset
- **Timing channels**
  - e.g. use high and low CPU utilization to single 1s and 0s; monitor progress of busy loop to detect CPU utilization
- **In general, can only hope to bound bandwidth of covert channels**

Operating Systems vs Virtual Machines

- **OS and Virtual Machine allow sharing of hardware with protections**
- **OS exposes hardware through a process abstraction**
  - Makes finite resources (memory, # CPU cores) appear much larger
  - Abstracts hardware to makes applications portable
  - Protects processes and users from one another
- **Virtual machine exposes hardware through a hardware abstraction**
  - Makes hardware resources appear larger or smaller
  - Allows almost any software {OS + Apps} to run
  - Protects {OS + Apps} from each other

Virtual Machine

- **Thin layer of software that virtualizes the hardware**



Virtual Machines

- **Benefits**
  - Software compatibility: any OS/App can run (even really old ones)
  - Hardware sharing: allow multiple servers to run on the same hardware
- **Ways to virtualize**
  - Complete Machine Simulation (too slow)
  - Basics
  - Binary Translation
  - Hardware-assisted virtualization

VMM Basics

- **CPU Virtualization**
  - Guest OS to runs in user mode
  - Trap to VMM when Guest OS does sensitive things
- **Virtual Memory Virtualization**
  - Guest OS to controls Guest Virtual to Guest Physical Address mapping
  - VMM controls Guest Physical to Host Physical Mapping
- **I/O Device Virtualization**
  - Simulate device behavior

Virtual Machine Implementations

- **Binary translation**
  - Dynamically rewrite code to replace sensitive instructions with jumps into the VMM
  - Most instructions are not sensitive so they can be translated identically
- **Hardware-assisted virtualization**
  - Hardware supports "guest mode"
  - VMM transfers control to guest using new "vmrun" instruction
  - Hardware defines VMCB control bits to tell the CPU which instructions should cause guest mode to "EXIT"

Topics

- **Processes and Threads**
- **Virtual Memory**
- **Concurrency**
- **Synchronization**
- **Linking**

- **Memory Allocation**
- **Device I/O**
- **File Systems**
- **Security**
- **Virtual Machines**

Good luck!