Goals for Today

- Learning Objective:
  - Present final exam details + review content
- Announcements, etc:
  - MP3 Soft Extension: Submit by **TODAY** for -10pts
  - MP4 due **May 7th**
    - Deadline provides more time than necessary; wanted to give you flexibility
  - vSphere console was temporarily down, is back up now
  - Review Homework will be posted lated today.

Reminder: Please put away devices at the start of class

CS 423
Operating System Design: Final Exam Overview

Professor Adam Bates

Spring 2018

Final Exam Details

- May 4th, 1:30pm - 3:30pm
  - You will have **2 hours**
- Scantron Multiple choice
- 30-40 Questions
  - Questions per minute will be less than Midterm
- **Openbook**: Textbooks, paper notes, printed sheets allowed. *No electronic devices permitted (or necessary)!*
- **Content**: All lecture and text material covered after the midterm content (i.e., starting with Virtualization)

Final Exam Content

- Virtualization (Emulation, Binary Translation…)
- File Systems (Disk Scheduling, Directories, Reliability…)
- Security (Access control, Encryption, Attacks, Reference monitors)
- Guest Lectures (Hardware Attacks, Process VMs)
- Remaining Special Topics (Energy, Linux Audit)
- *Exam questions will not be <u>explicitly</u> cumulative, but I can't guarantee that content from before the midterm won't come up in some fashion.*

Virtualization

- Key Concepts:
  - Different purposes for virtualization
  - Different virtualization layers
  - Emulation versus Binary Translation
  - Dynamic Binary Translation Challenges + Optimizations
  - Challenges of Process VMs
    - e.g., Emulating Target Architecture
  - Interpretation/Emulation versus Translation

What's a virtual machine?

- Virtual machine is an entity that emulates a guest interface on top of a host machine
  - Language view:
    - Virtual machine = Entity that emulates an API (e.g., JAVA) on top of another
    - Virtualizing software = compiler/interpreter
  - Process view:
    - Machine = Entity that emulates an ABI on top of another
    - Virtualizing software = runtime
  - Operating system view:
    - Machine = Entity that emulates an ISA
    - Virtualizing software = virtual machine monitor (VMM)

Different views == who are we trying to fool??

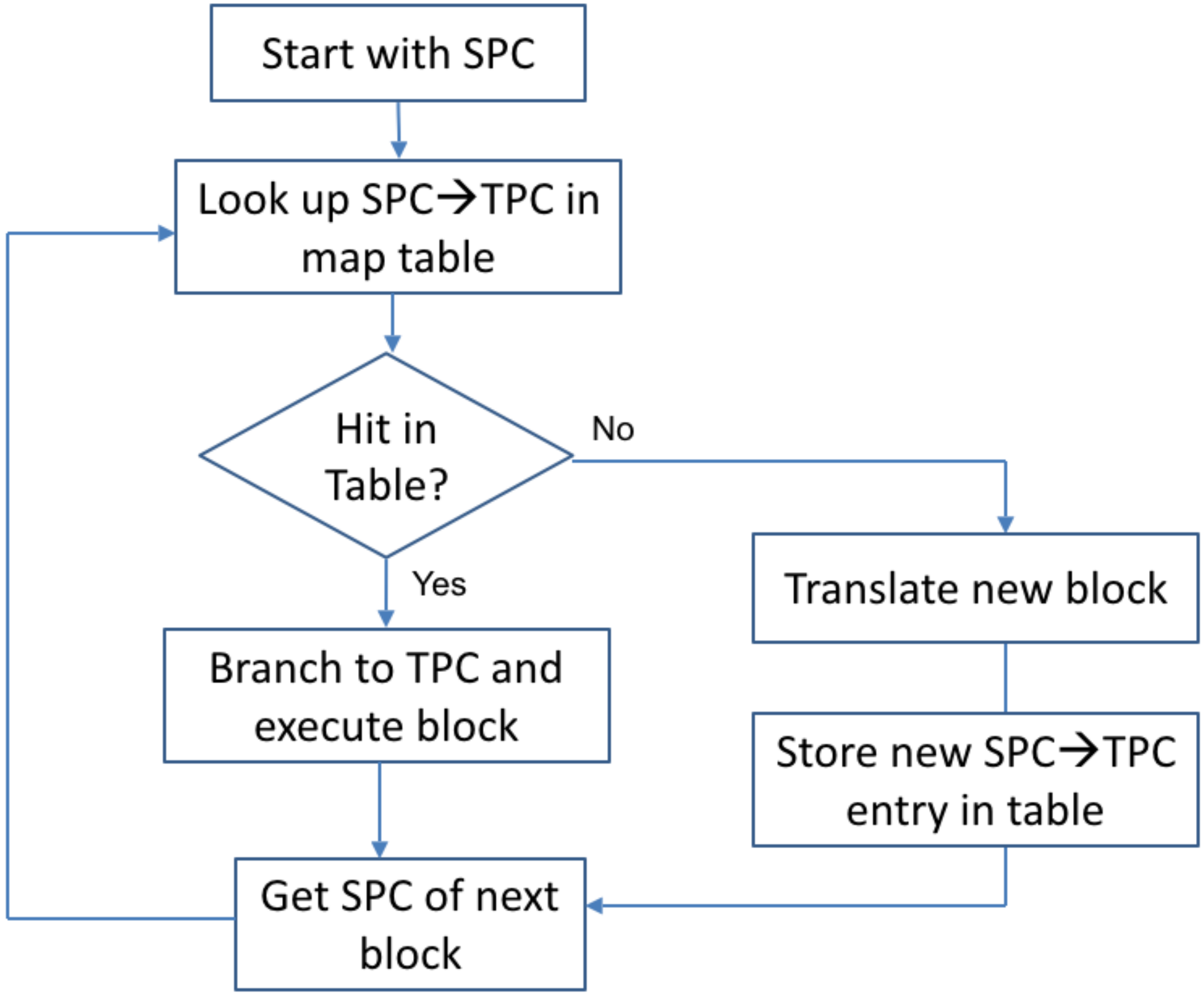
Purpose of a VM

- Emulation
  – Create the illusion of having one type of machine on top of another
- Replication (/ Multiplexing)
  – Create the illusion of multiple independent smaller guest machines on top of one host machine (e.g., for security/isolation, or scalability/sharing)
- Optimization
  – Optimize a generic guest interface for one type of host

Writing an Emulator

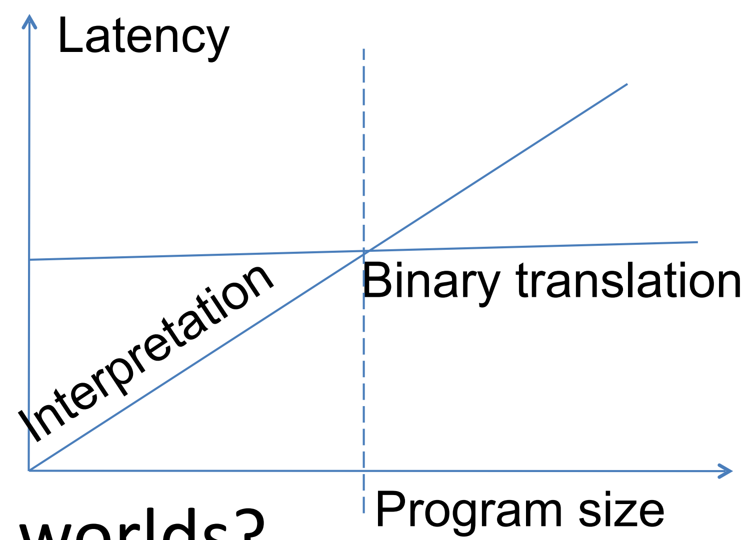
- Problem: Emulate guest ISA on host ISA
- Create a simulator data structure to represent:
  - Guest memory
    - Guest stack
    - Guest heap
  - Guest registers
- Inspect each binary instruction (machine instruction or system call)
  - Update the data structures to reflect the effect of the instruction

Dynamic Binary Translation



Instruction Emulation

- Interpretation versus binary translation?
  - Interpretation:
    - no startup overhead
    - High overhead per instruction
  - Binary translation:
    - High startup overhead
    - Low overhead per instruction
  - Can we combine the best of both worlds?
    - Small program: Do interpretation
    - Large program: Do binary translation

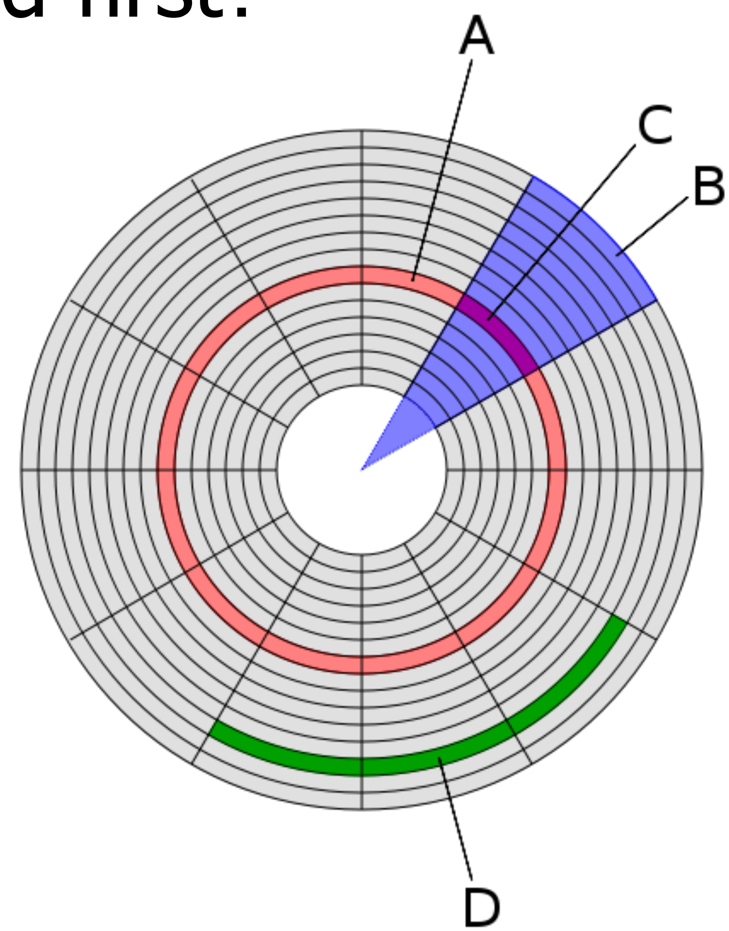


File Systems

- Key Concepts:
  - Disk Scheduling
    - Concepts + Modern Implementations
  - Data Layout on Disk
  - File Allocation Strategies
    - Concepts + Modern Implementations
    - Locality
  - Directory Structures
    - Representing Large Directories
  - Reliability
    - Transaction Concept + Implementations
    - RAID

Disk Scheduling

- Which disk request is serviced first?
  - FCFS
  - Shortest seek time first
  - Elevator (SCAN)
  - C-SCAN (Circular SCAN)



A: Track.
B: Sector.
C: Sector of Track.
D: File

Disk Scheduling Decision — Given a series of access requests, on which track should the disk arm be placed next to maximize fairness, throughput, etc?

Linux I/O Schedulers

- What disk (I/O) schedulers are available in Linux?

```
$ cat /sys/block/sda/queue/scheduler
noop [deadline] cfq
```

^ scheduler enabled on our VMs

- As of Linux 2.6.10, it is possible to change the IO scheduler for a given block device on the fly!
- How to enable a specific scheduler?

```
$ echo SCHEDNAME > /sys/block/DEV/queue/scheduler
```

  - SCHEDNAME = Desired I/O scheduler
  - DEV = device name (e.g., hda)

Disk Layout for a FS

Disk layout in a typical file system:

Boot block	Super block	File metadata (i-node in Unix)	File data blocks

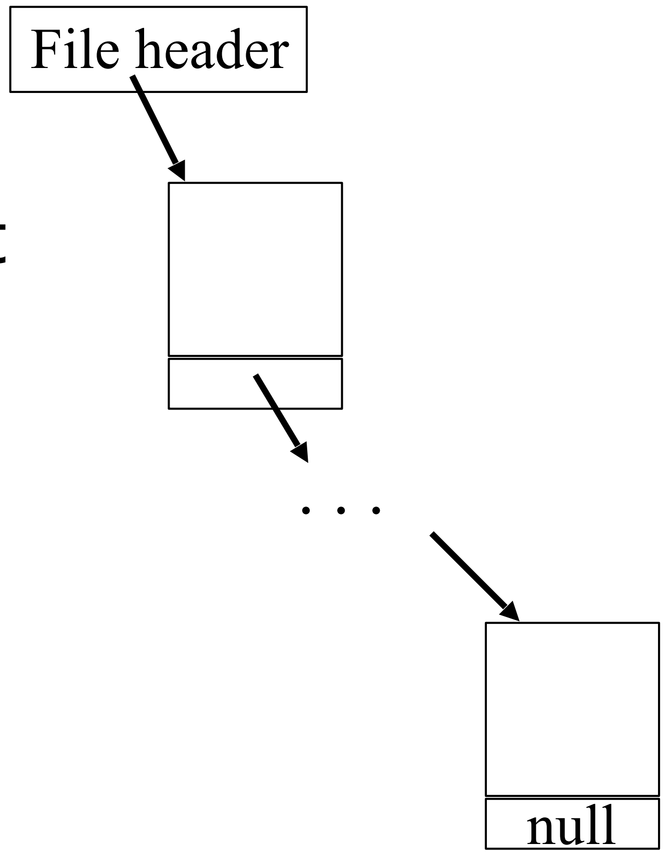
- Superblock defines a file system
  - size of the file system
  - size of the file descriptor area
  - free list pointer, or pointer to bitmap
  - location of the file descriptor of the root directory
  - other meta-data such as permission and various times
- For reliability, replicate the superblock

Contiguous Allocation

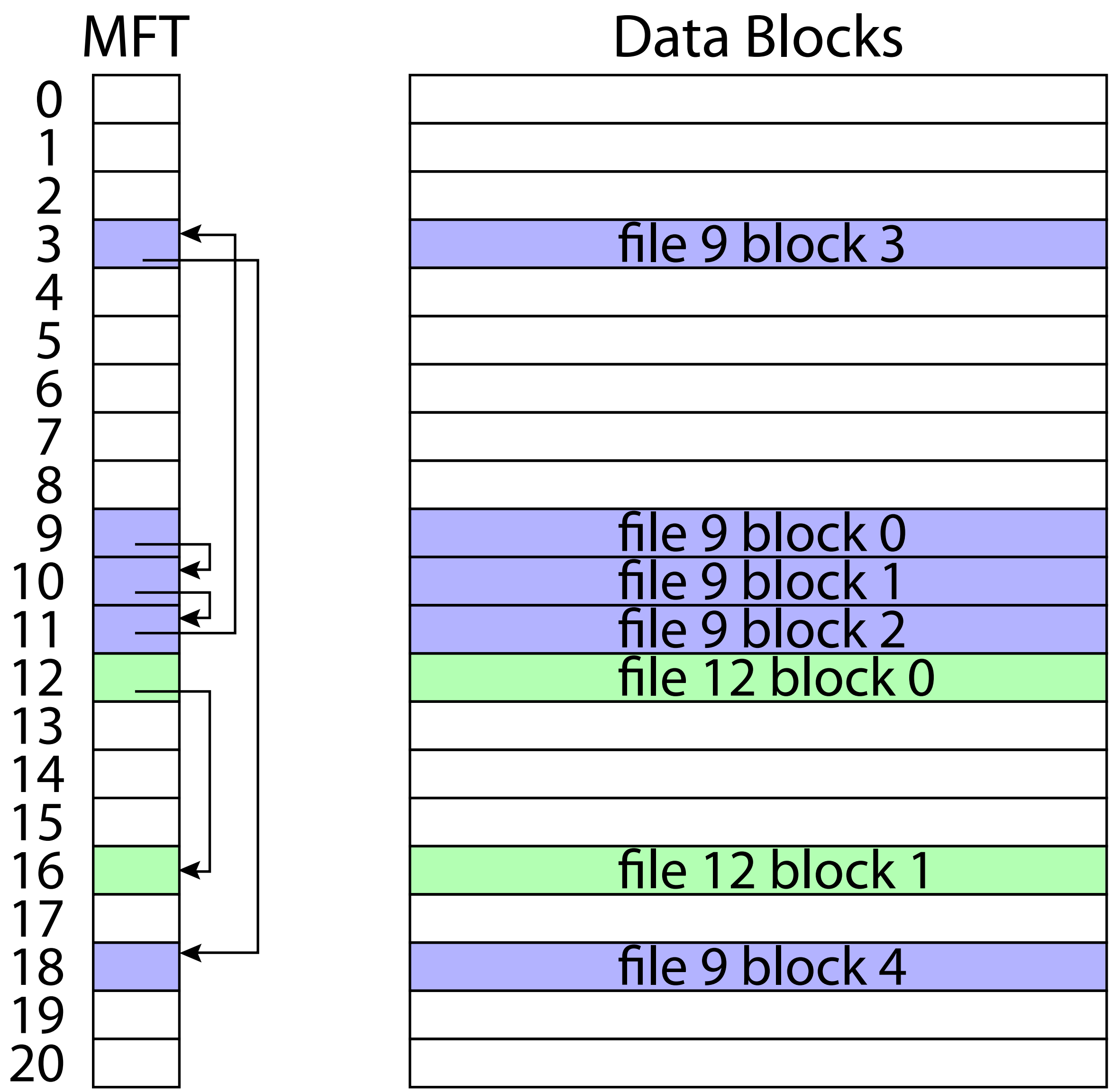
- Request in advance for the size of the file
- Search bit map or linked list to locate a space
- File header
  - first sector in file
  - number of sectors
- Pros
  - Fast sequential access
  - Easy random access
- Cons
  - External fragmentation
  - Hard to grow files

Linked Files

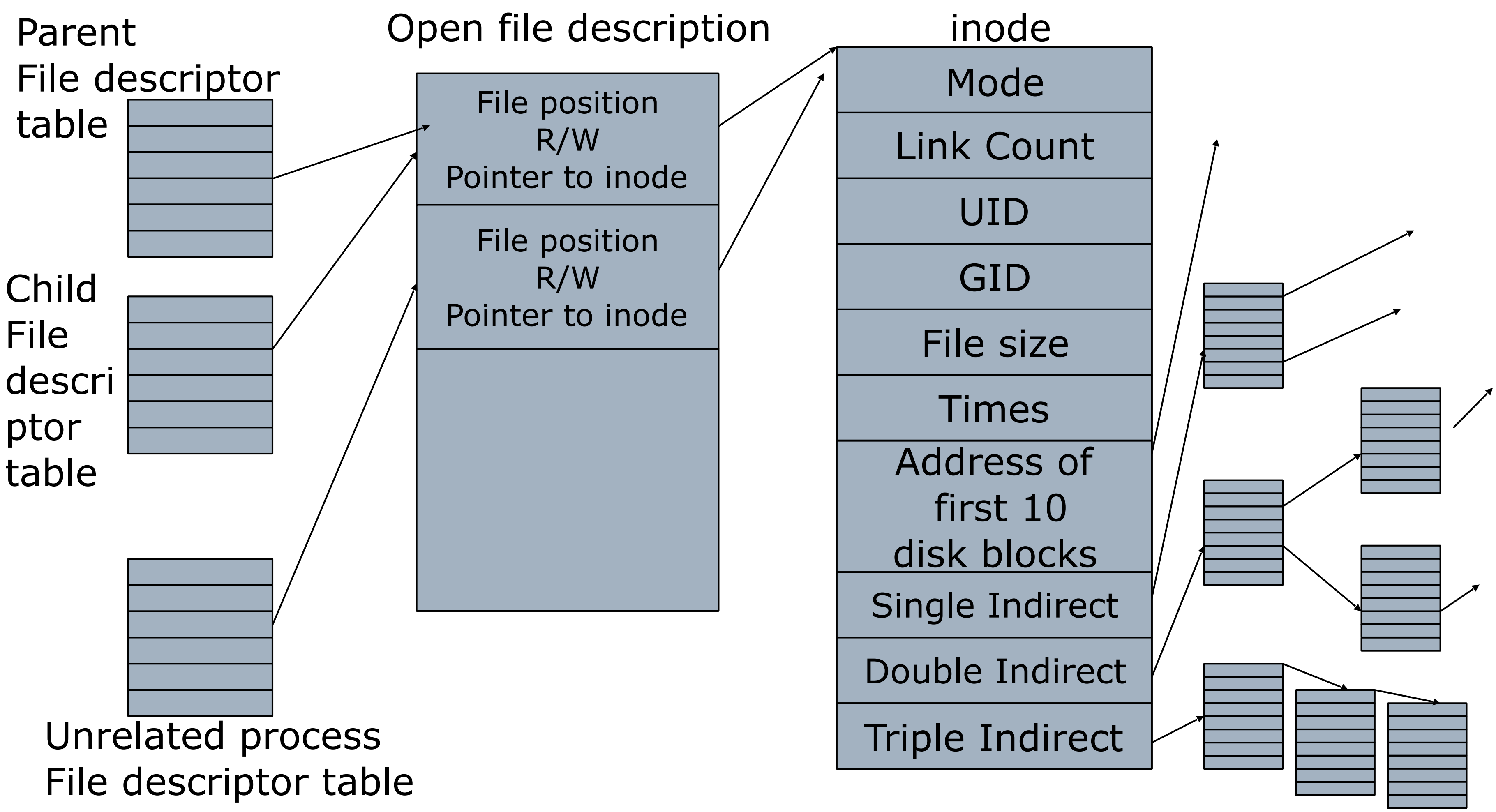
- File header points to 1st block on disk
- Each block points to next
- Pros
  - Can grow files dynamically
  - Free list is similar to a file
- Cons
  - random access: horrible
  - unreliable: losing a block means losing the rest



MS File Allocation Table (FAT)



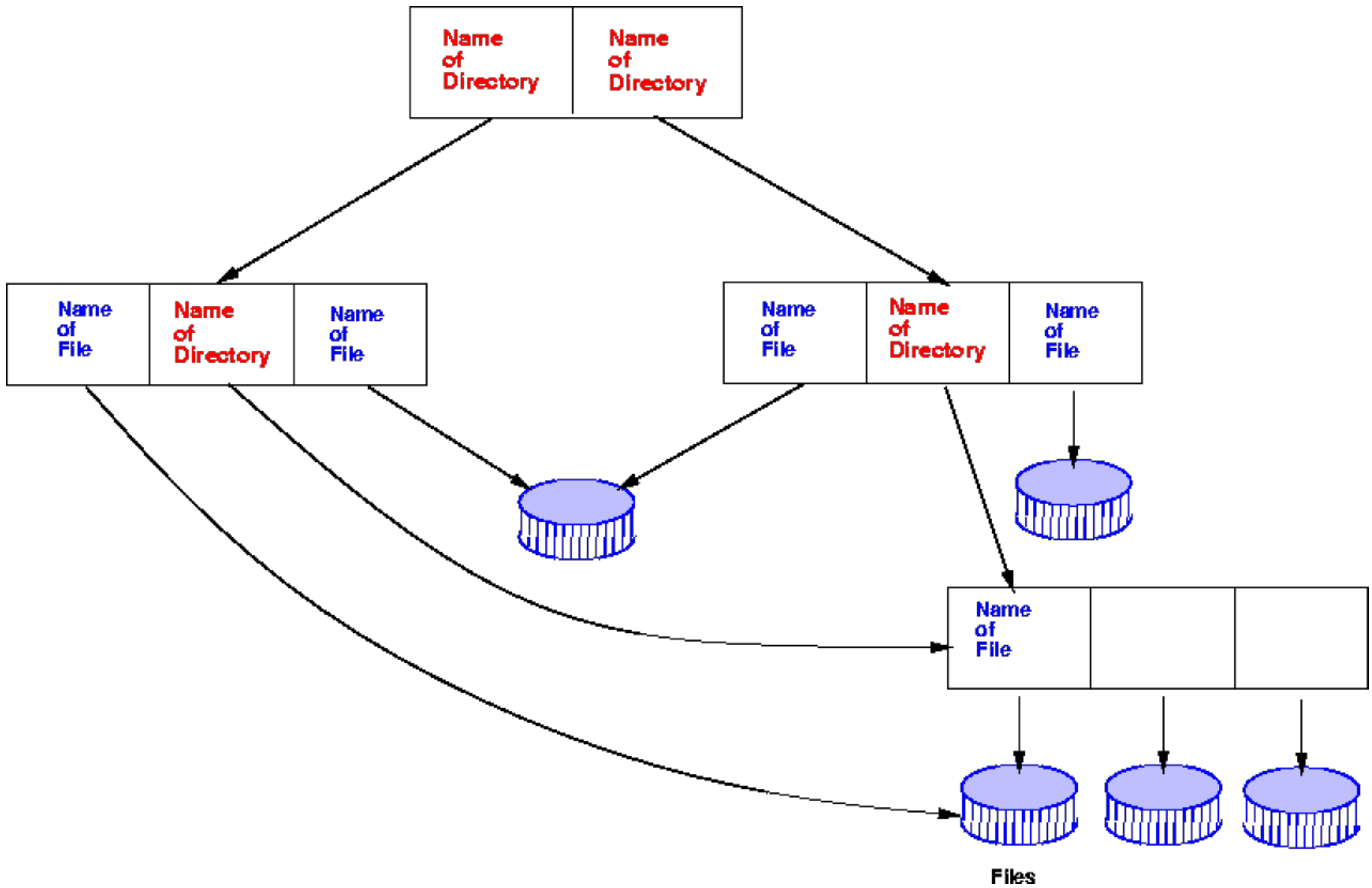
Berkeley FFS / UNIX FS



Berkeley FFS Locality

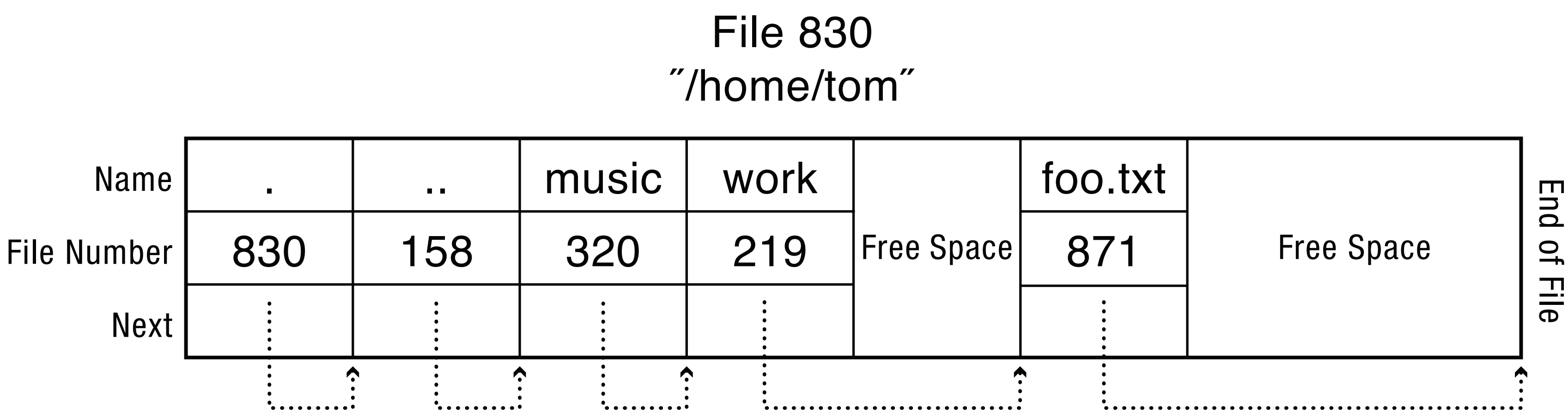
- How does FFS provide locality?
- Block group allocation
  - Block group is a set of nearby cylinders
  - Files in same directory located in same group
  - Subdirectories located in different block groups
- inode table spread throughout disk
  - inodes, bitmap near file blocks
- First fit allocation
  - Property: Small files may be a little fragmented, but large files will be contiguous

Acyclic Graph Structured Dir's



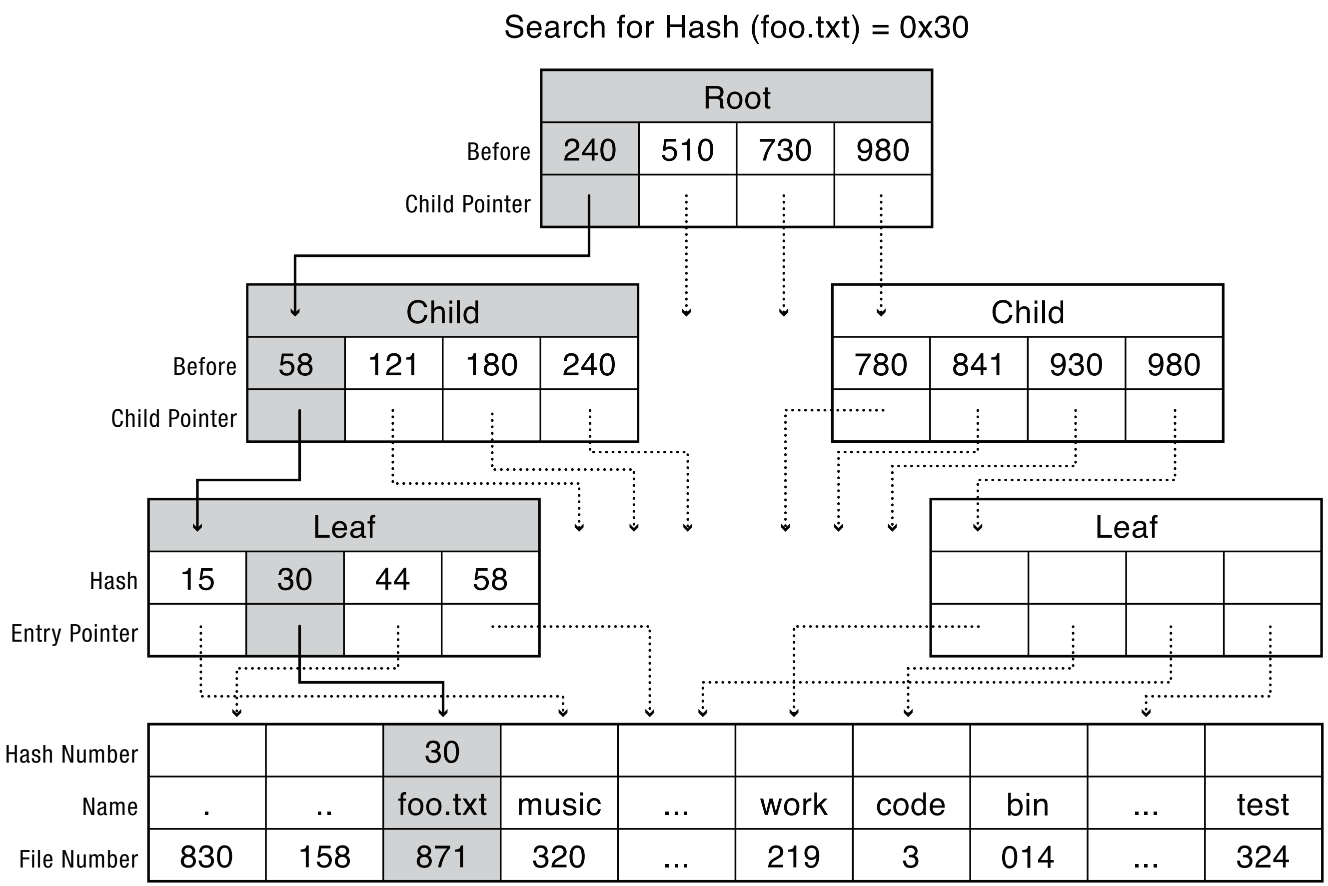
Directory Layout

- Represent directory as a list of files
- Linear search to find filename
- Suitable for small directories



B Trees

- Logarithmic search to find filename
- Suitable for large directories



Transaction Concept

- A <u>transaction</u> is a grouping of low-level operations that are related to a single logical operation
- Transactions are <u>atomic</u> — operations appear to happen as a group, or not at all (at logical level)
  - At physical level of course, only a single disk/flash write is atomic
- Transactions are <u>durable</u> — operations that complete stay completed
  - Future failures do not corrupt previously stored data
- (In-Progress) Transactions are <u>isolated</u> — other transactions cannot see the results of earlier transactions until they are committed
- Transactions exhibit <u>consistency</u> — sequential memory model

Reliability Attempt #1: Careful Ordering

Pros

- Works with minimal support from the disk drive
- Works for most multi-step operations

Cons

- Can require time-consuming recovery after a failure
- Difficult to reduce every operation to a safely-interruptible sequence of writes
- Difficult to achieve consistency when multiple operations occur concurrently (e.g., FFS grep)

Reliability Attempt #2: Copy-on-Write

Pros

- Correct behavior regardless of failures
- Fast recovery (root block array)
- High throughput (best if updates are batched)

Cons

- Potential for high latency
- Small changes require many writes
- Garbage collection essential for performance

Logging File Systems

- Instead of modifying data structures on disk directly, write changes to a journal/log
  - Intention list: set of changes we intend to make
  - Log/Journal is append-only

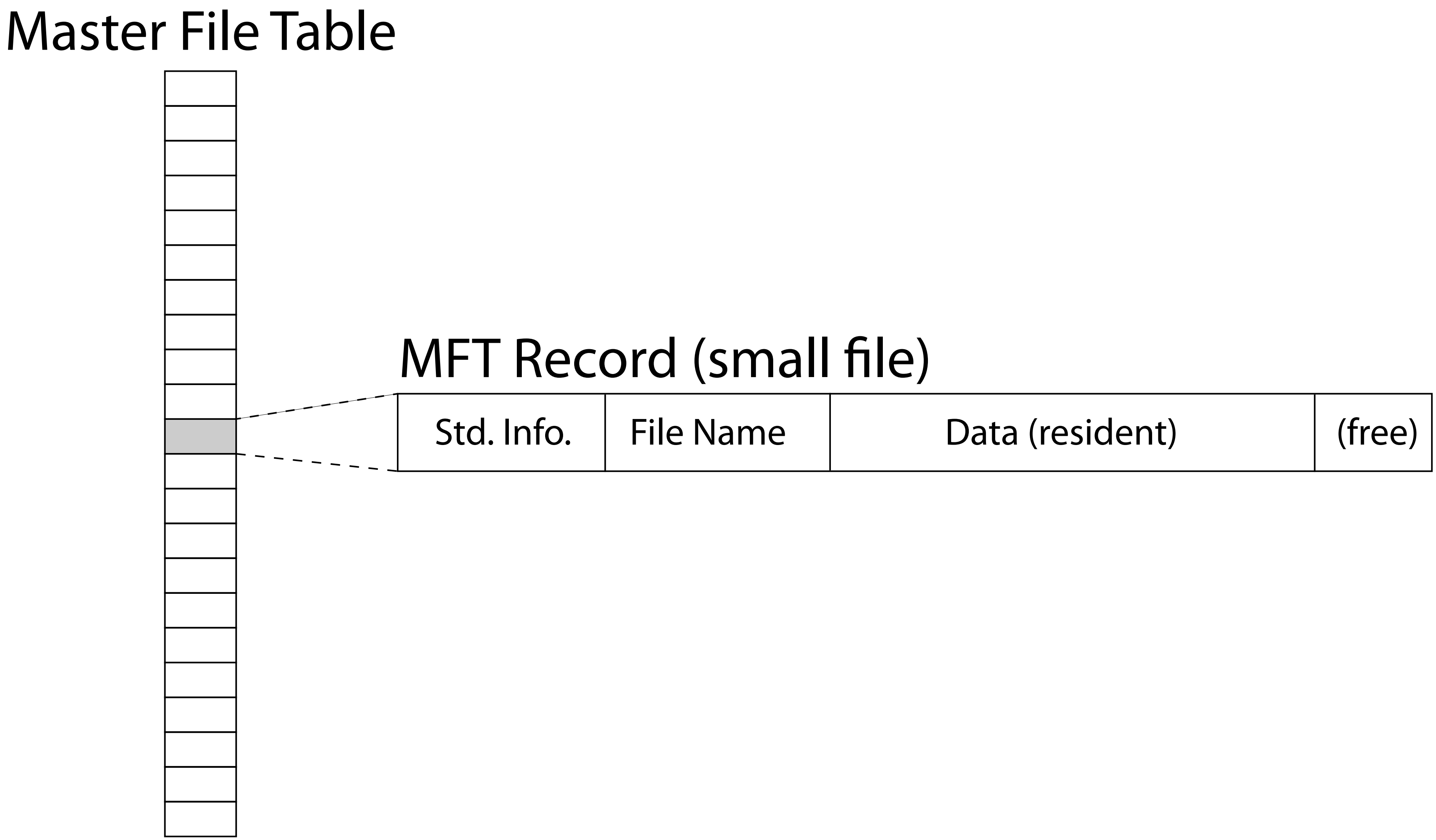
- Once changes are on log, safe to apply changes to data structures on disk
  - Recovery can read log to see what changes were intended

- Once changes are copied, safe to remove log

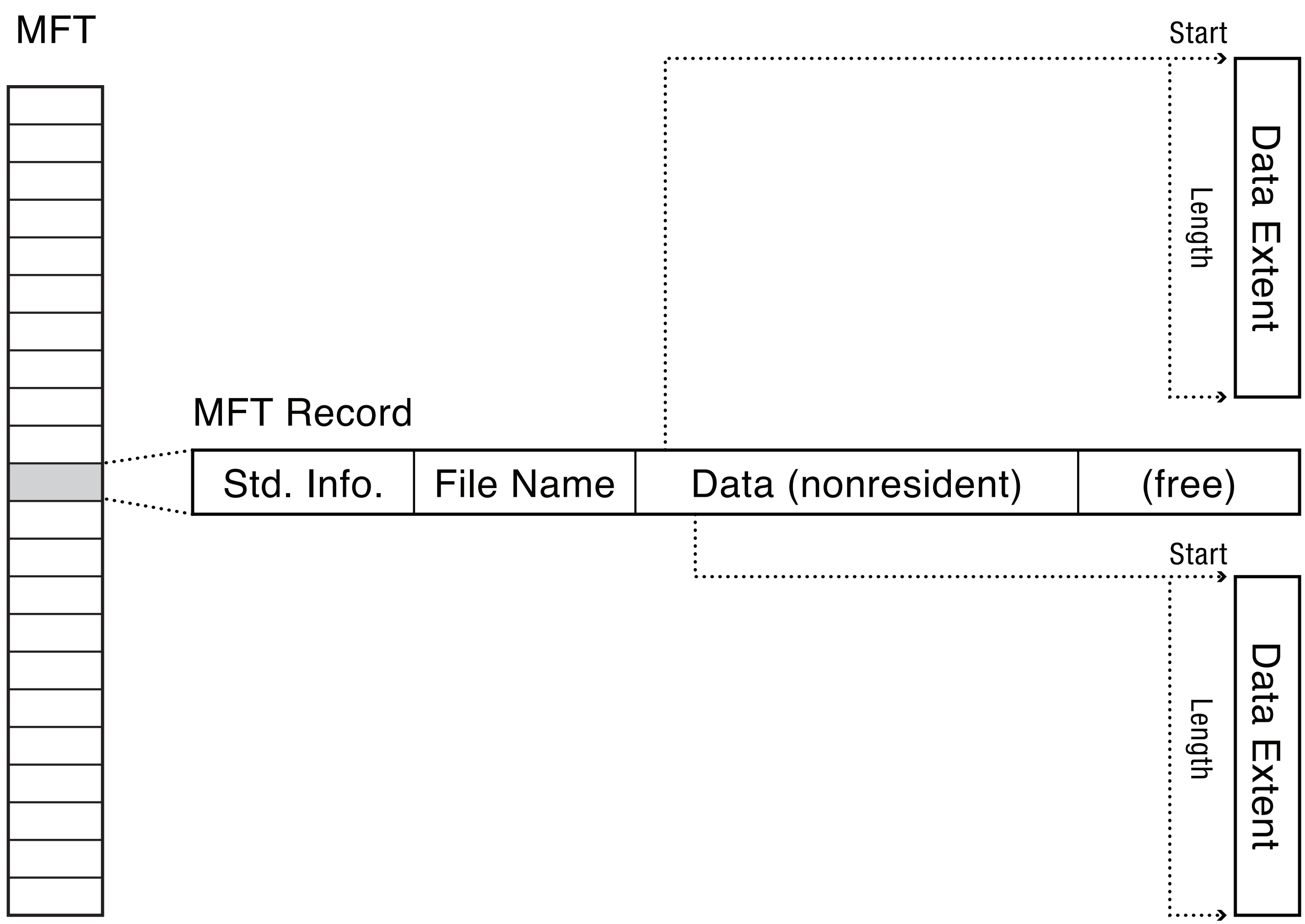
What about NTFS?

- Improved Metadata support
  - Flexible 1KB storage for metadata and data
- Scalability Features
  - MFT is optimized for 4KB resident data
  - Extents: a middle ground between contiguous and non-contiguous allocation.
    - Block pointers cover runs of blocks
    - Similar approach in linux (ext4)
- NTFS uses journalling for reliability

NTFS

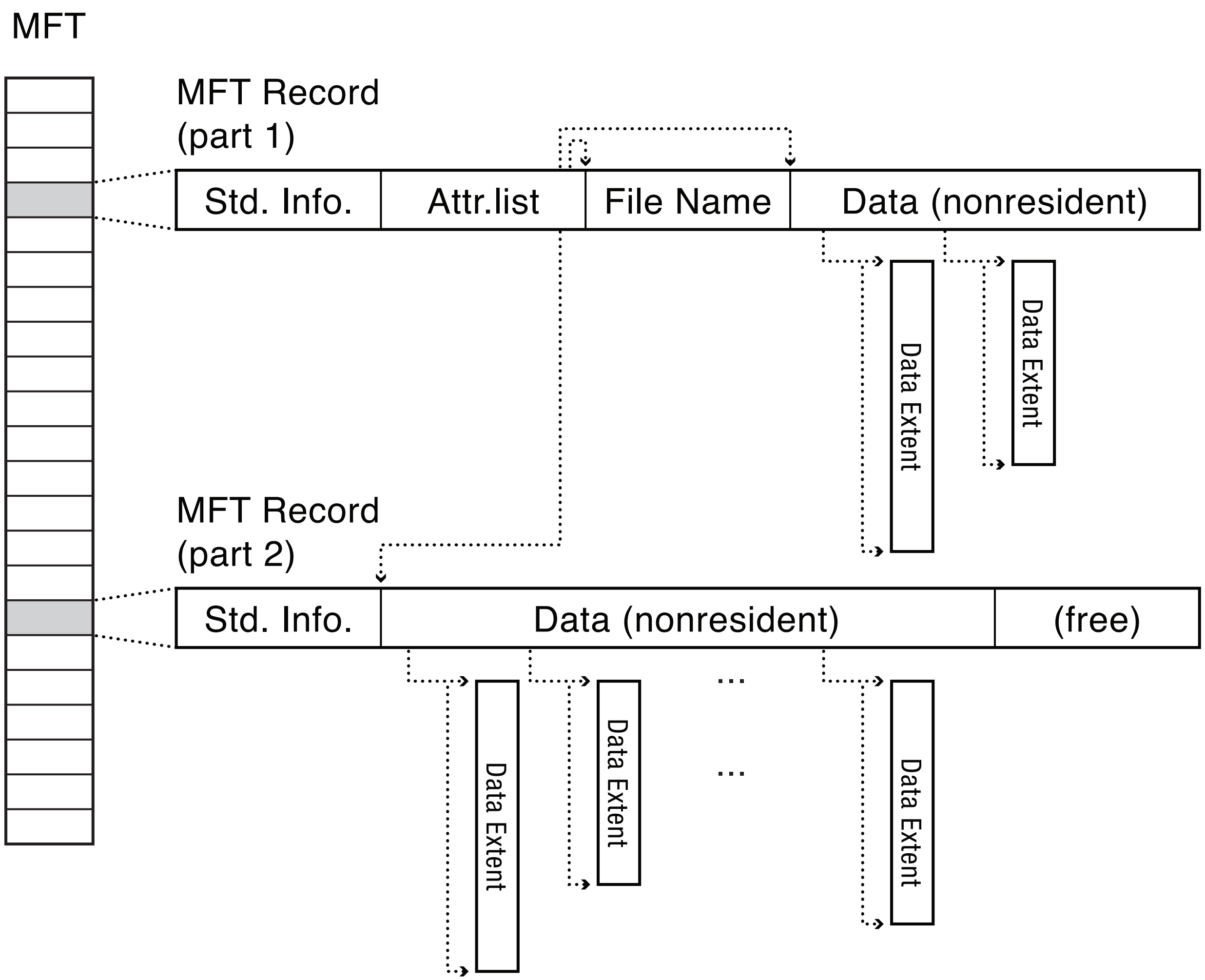

Master File Table
MFT Record (small file)
Std. Info.
File Name
Data (resident)
(free)

NTFS


MFT
MFT Record
Std. Info.
File Name
Data (nonresident)
(free)
Start
Length
Data Extent
Start
Length
Data Extent

NTFS Indirect Block

What if file is too large to fit all extent pointers in one data cluster?

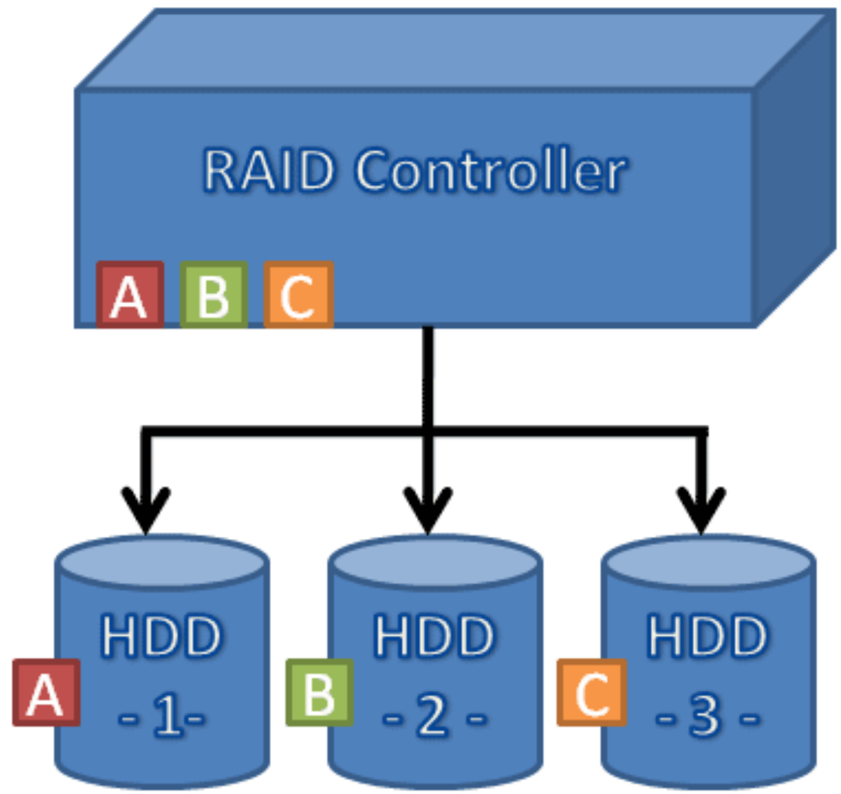


RAID

- RAID
  - multiple disks work cooperatively
  - Improve reliability by storing redundant data
  - **Striping** (**RAID 0**) improves performance with disk **striping** (use a group of disks as one storage unit)
  - **Mirroring** (**RAID 1**) keeps duplicate of each disk
  - Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy

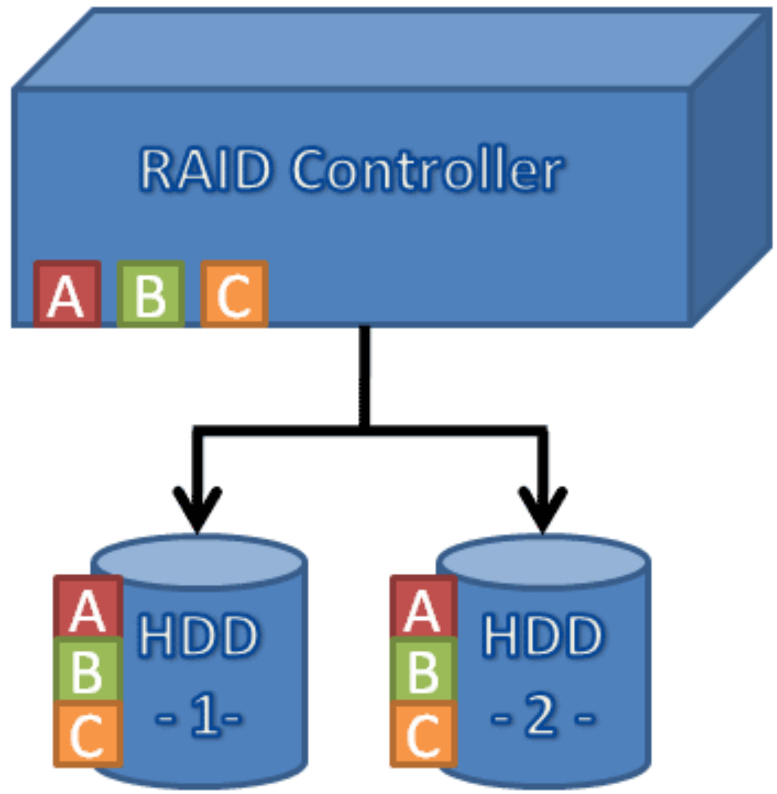
RAID Level 0

- Level 0 is nonredundant disk array
- Files are striped across disks, no redundant info
- High read throughput
- Best write throughput (no redundant info to write)
- Any disk failure results in data loss



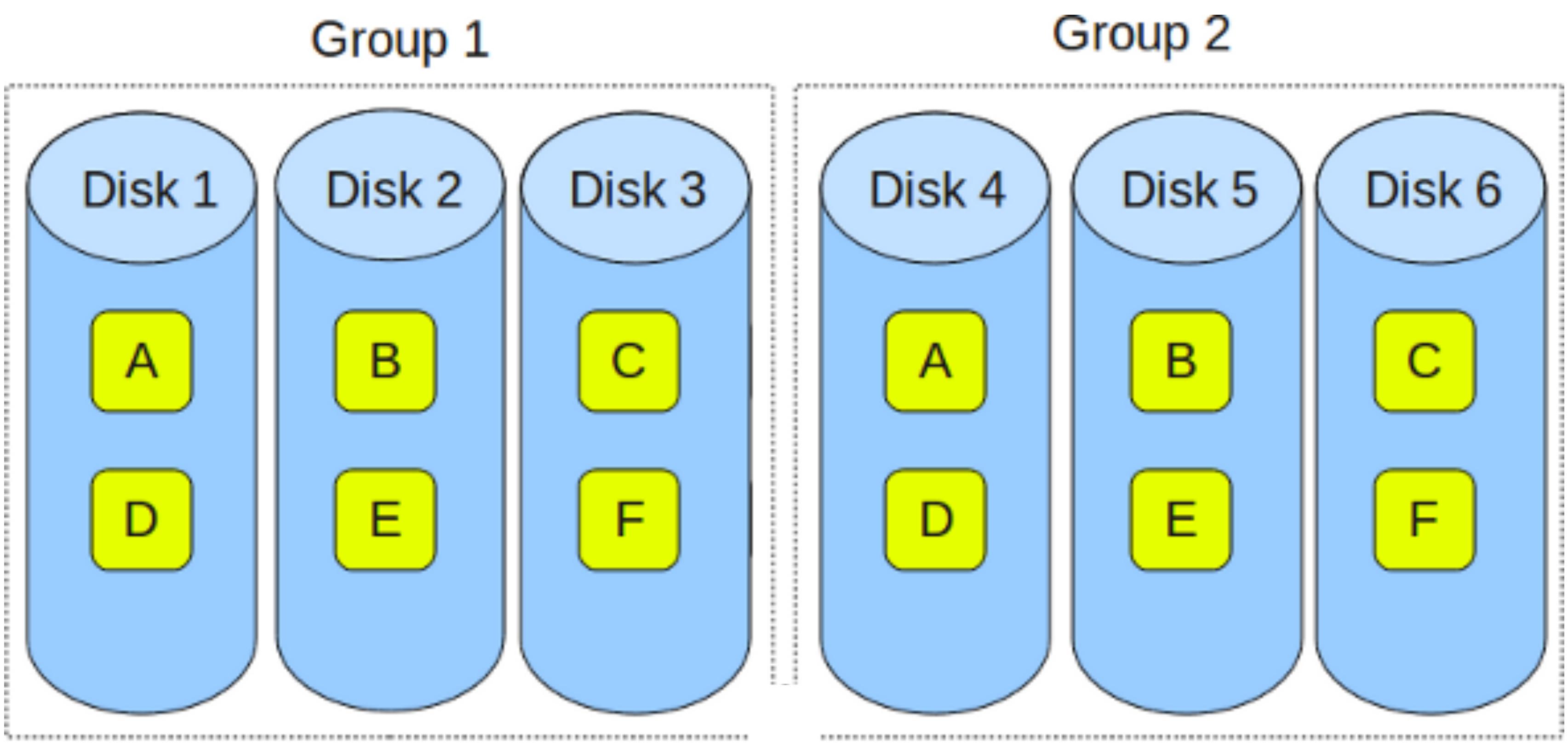
RAID Level 1

- Mirrored Disks
- Data is written to two places
  - On failure, just use surviving disk (easy to rebuild)
- On read, choose fastest to read
  - Write performance is same as single drive, read performance is 2x better
- Expensive (high space overhead)



RAID Level 0+1

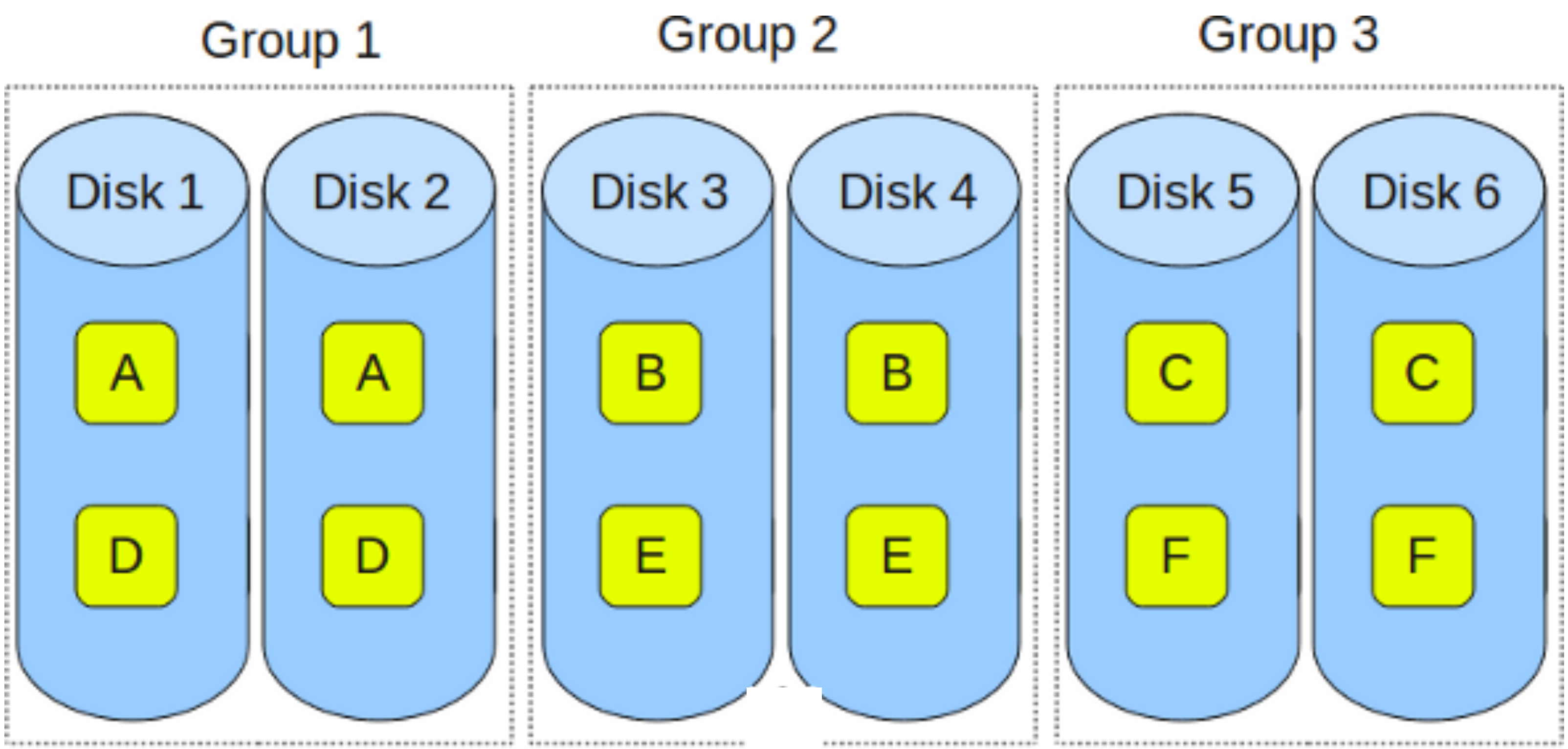
- Stripe on a set of disks
- Then mirror of data blocks is striped on the second set.



RAID 01 – Blocks Striped. (and Blocks Mirrored)

RAID Level 1+0

- Pair mirrors first.
- Then stripe on a set of paired mirrors



RAID 10 – Blocks Mirrored. (and Blocks Striped)

Security

- <u>Key Concepts</u>:
  - Least Privilege
  - Encryption — have a High-level / Block Box comprehension. You won't need to prove RSA.
  - Authentication + Passwords
  - Why do secure systems (epically) fail?
  - Cache Side-Channels (Fletcher lecture)
  - Access Control
    - e.g., DAC, Capabilities, Bell-LaPadula
  - Cryptography versus Access Control
  - Reference Monitors, LSM, SELinux

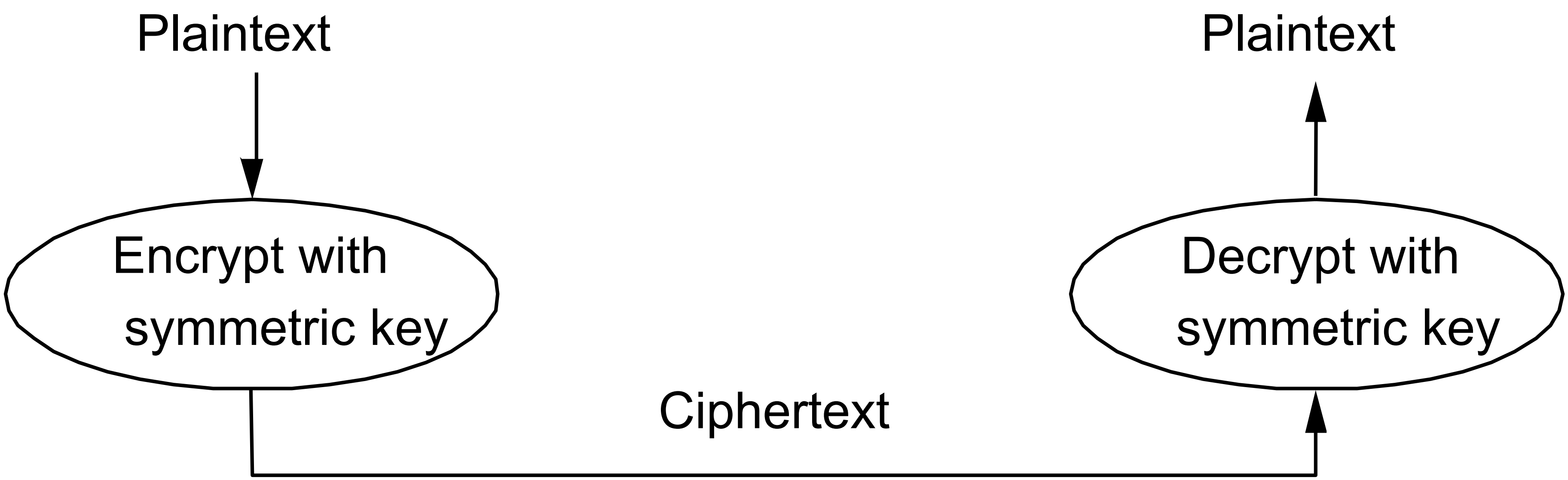
How to study for security?

- No corresponding chapter in textbook!?!?
- We won't be straying far from the lectures/slides
  - Google + Wikipedia concepts if you need further clarification (just evaluate credibility of sources).

Principle of Least Privilege

- Grant each principal the least permission possible for them to do their assigned work
  - Minimize code running inside kernel
  - Minimize code running as sysadmin
- Practical challenge: hard to know
  - what permissions are needed in advance
  - what permissions should be granted
    - Ex: to smartphone apps
    - Ex: to servers

Symmetric Key (DES, IDEA)



- Single key (symmetric) is shared between parties, kept secret from everyone else
  - Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
  - if K kept secret, then both parties know M is authentic and secret

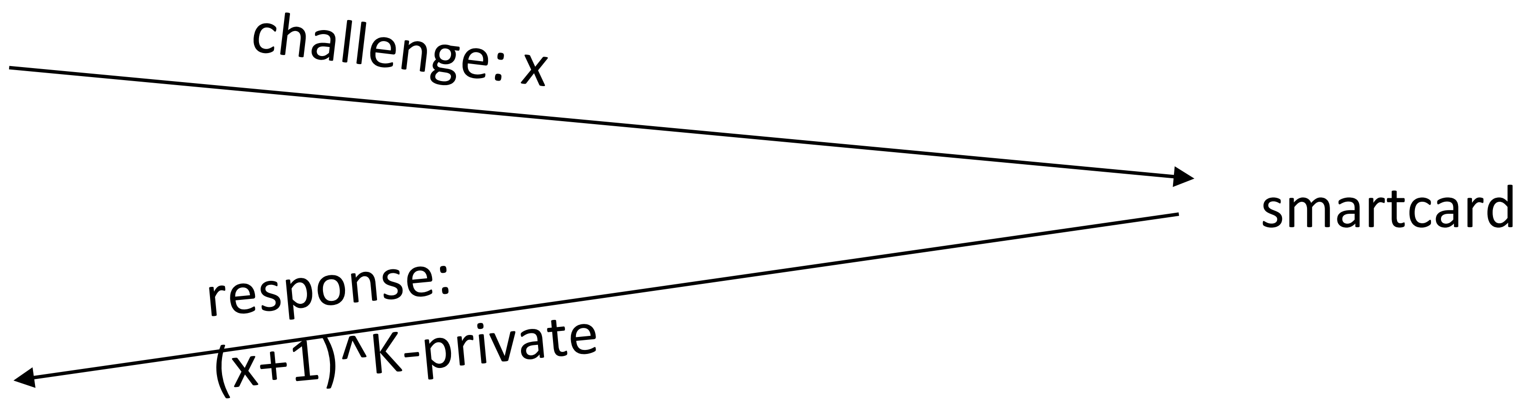
Public Key (RSA, PGP)

Keys come in pairs: public and private

- M = ((M)^K-public)^K-private
- Ensures secrecy: can only be read by receiver

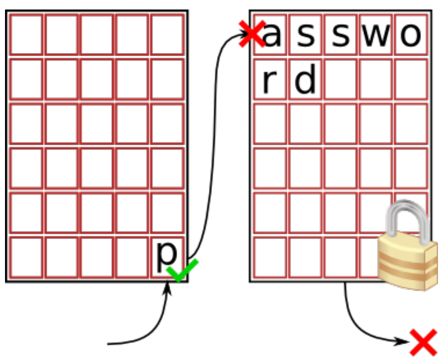
2-Factor Authentication

- Can be difficult for people to remember encryption keys and passwords
- Instead, store K-private inside a chip
  - use challenge-response to authenticate smartcard
  - Use PIN to prove user has smartcard



Ex1: Tenex Password Vuln

- Observation: Programs have *a lot* of control over how their virtual memory works.
- Attack #1: Trap-To-User Bit Exploit



Trap-To-User: Alert me if this 2nd page is accessed!

- Attack #2: Exploit timing side-channel

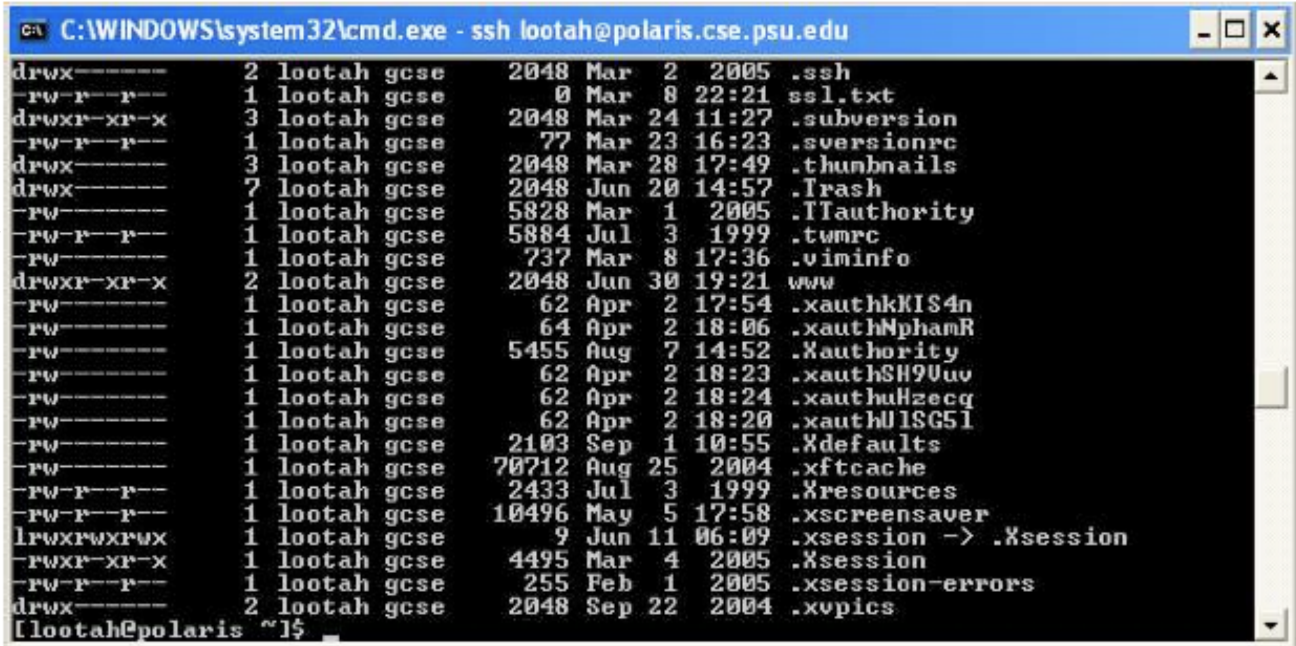
Processing time for password check was proportional to the number of correct characters at the front of the attacker's guess.

Reflections on Trusting Trust

- Thompson's Takeaway: You can't fully trust code that you didn't write yourself!
- Presented as a thought experiment during Thompson's Turing Award Lecture. Didn't really happen… we think??
- Hard to re-secure a machine after penetration. How do you know you've removed all the backdoors?
- It's hard to detect that a machine has been penetrated
- Any system with bugs is vulnerable
  - and all systems have bugs

Discretionary Access Control (DAC)

Access Mask defines permissions for User, Group, and Other




```
chmod u=rwx,g=rx,o=r myfile
chmod 754 myfile
```

<- Same thing

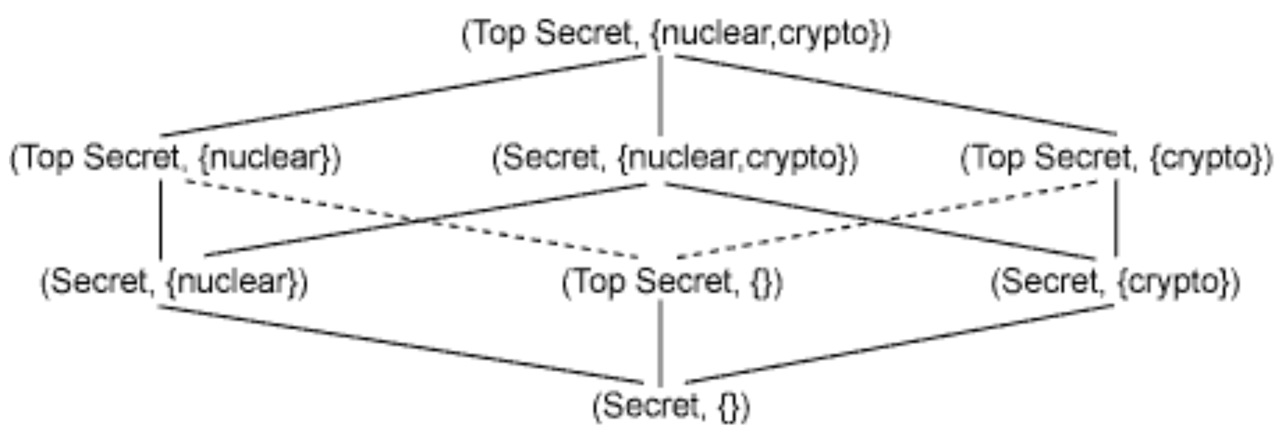
4 stands for "read",
2 stands for "write",
1 stands for "execute", and
0 stands for "no permission."

Problems?

- What might go wrong with DAC or Capabilities?
  - Security is left to the discretion of subjects
  - Impossible to guarantee security of system
  - Security of system changes over time.
- Solution?
  - <u>Mandatory Access Control</u>: Operating system constrains the ability of subjects (even owners) to perform operations on objects according to a system-wide *security policy*.

Bell-LaPadula Model

- A multi-level security model that provides strong confidentiality guarantees.
- Formalizes Classified Information
- State machine (Lattice) specifies permissible actions



SELinux

- Designed by the NSA
- A more flexible solution than MLS
- SELinux Policies are comprised of 3 components:
    - Labeling State defines security contexts for every file (object) and user (subject).
    - Protection State defines the permitted <subject,object,operation> tuples.
    - Transition State permits ways for subjects and objects to move between security contexts.
- Enforcement mechanism designed to satisfy reference monitor concept

LSM Architecture

- Linux Kernel modified in 5 ways:
  - Opaque security fields added to certain kernel data structures
  - Security hook function calls inserted at various points with the kernel code
  - A generic security system call added
  - Function to allow modules to register and unregistered as security modules
  - Move capabilities logic into an optional security module

Reference Monitor

Cool. But how do we implement these models in an operating system?