UNIVERSITY OF OREGON

# CIS 415: Operating Systems

## Final Review

Spring 2012

Prof. Kevin Butler

# Final Exam

- Wednesday, June 13 at 8 AM (section 1) and 1 PM (section 2)
- 2 hours
- Structure: similar to previous exam in terms of question layout, may be similar length or possibly a bit longer
- Questions:
  - some technical details
  - some conceptual questions
- Material: slides, text, class discussion, homeworks

# What we've covered

UNIVERSITY OF OREGON

- Everything from the first part of the course is fair game
  - but probably de-emphasized a bit - bit twiddling probably not required but have a strong idea of the concepts
- Chapter 6 -- Synchronization
- Chapter 7 -- Deadlocks & Chapter 18 -- Distributed Coordination
- Chapter 8 -- Main Memory (Physical)
- Chapter 9 -- Virtual Memory
- Chapter 10 -- File System Interface & Chapter 14 -- Protection
- Chapter 11 -- File System Implementation
- Chapter 12 -- Storage
- Chapter 13 -- I/O
- Chapter 21 - Linux

# Synchronization



- Problems
- Synchronization Requirements
- Disabling Interrupts
- Busy-wait/Spinlock solutions
  - Related to properties
- Hardware-enabled Solutions
- OS-supported

# Requirements for Solution

- 1. Mutual Exclusion - If process Pi is executing in its critical section, then no other processes can be executing in their critical sections
- 2. Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely
- 3. Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted
  - Assume that each process executes at a nonzero speed
  - No assumption concerning relative speed of the N processes

# Synchronization

UNIVERSITY OF OREGON

- Hardware Enabled Solutions
- OS-supported Solutions
  - Mutex
  - Semaphores
  - Condition Variables
- Apply these to code
- Classic Synchronization Problems

# Semaphores

UNIVERSITY OF OREGON

- You are given a data-type Semaphore_t.
- On a variable of this type, you are allowed
  - P(Semaphore_t) -- wait
  - V(Semaphore_t) – signal
- Intuitive Functionality:
  - Logically one could visualize the semaphore as having a counter initially set to 0.
  - When you do a P(), you decrement the count, and need to block if the count becomes negative.
  - When you do a V(), you increment the count and you wake up 1 process from its blocked queue if not null.

# Deadlocks

UNIVERSITY OF OREGON

- Necessary Conditions
- Safe States
- Resource Allocation Graph
- Deadlock Prevention
  - Safe States
- Deadlock Detection
  - Detection Algorithm
  - Recovery

# Necessary Conditions for a Deadlock

- Mutual exclusion: The requesting process is delayed until the resource held by another is released.
- Hold and wait: A process must be holding at least 1 resource and must be waiting for 1 or more resources held by others.
- No preemption: Resources cannot be preempted from one and given to another.
- Circular wait: A set (P0,P1,…Pn) of waiting processes must exist such that P0 is waiting for a resource held by P1, P1 is waiting for …. by P2, … Pn is waiting for … held by P0.

# Deadlock Prevention Example



5 processes, 3 resource types A (10 instances), B (5 instances), C (7 instances)

**MaxNeeds**

| | A | B | C |
|---|---|---|---|
| P0 | 7 | 5 | 3 |
| P1 | 3 | 2 | 2 |
| P2 | 9 | 0 | 2 |
| P3 | 2 | 2 | 2 |
| P4 | 4 | 3 | 3 |

**Allocated**

| | A | B | C |
|---|---|---|---|
| P0 | 0 | 1 | 0 |
| P1 | 2 | 0 | 0 |
| P2 | 3 | 0 | 2 |
| P3 | 2 | 1 | 1 |
| P4 | 0 | 0 | 2 |

**StillNeeds**

| | A | B | C |
|---|---|---|---|
| P0 | 7 | 4 | 3 |
| P1 | 1 | 2 | 2 |
| P2 | 6 | 0 | 0 |
| P3 | 0 | 1 | 1 |
| P4 | 4 | 3 | 1 |

**Free**

| A | B | C |
|---|---|---|
| 3 | 3 | 2 |

**This state is safe, because there is a reduction sequence <P1, P3, P4, P2, P0> that can satisfy all the requests.**

**Exercise: Formally go through each of the steps that update these matrices for the reduction sequence.**

# Deadlock Detection Example

5 processes, 3 resource types A (7 instances), B (2 instances), C (6 instances)

**Allocated**

| | A | B | C |
|---|---|---|---|
| P0 | 0 | 1 | 0 |
| P1 | 2 | 0 | 0 |
| P2 | 3 | 0 | 3 |
| P3 | 2 | 1 | 1 |
| P4 | 0 | 0 | 2 |

**Request**

| | A | B | C |
|---|---|---|---|
| P0 | 0 | 0 | 0 |
| P1 | 2 | 0 | 2 |
| P2 | 0 | 0 | 0 |
| P3 | 1 | 0 | 0 |
| P4 | 0 | 0 | 2 |

**Free**

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |

This state is NOT deadlocked.

By applying algorithm, the sequence <P0, P2, P3, P1, P4> will result in Done[i] being TRUE for all processes.

# Distributed Coordination

UNIVERSITY OF OREGON

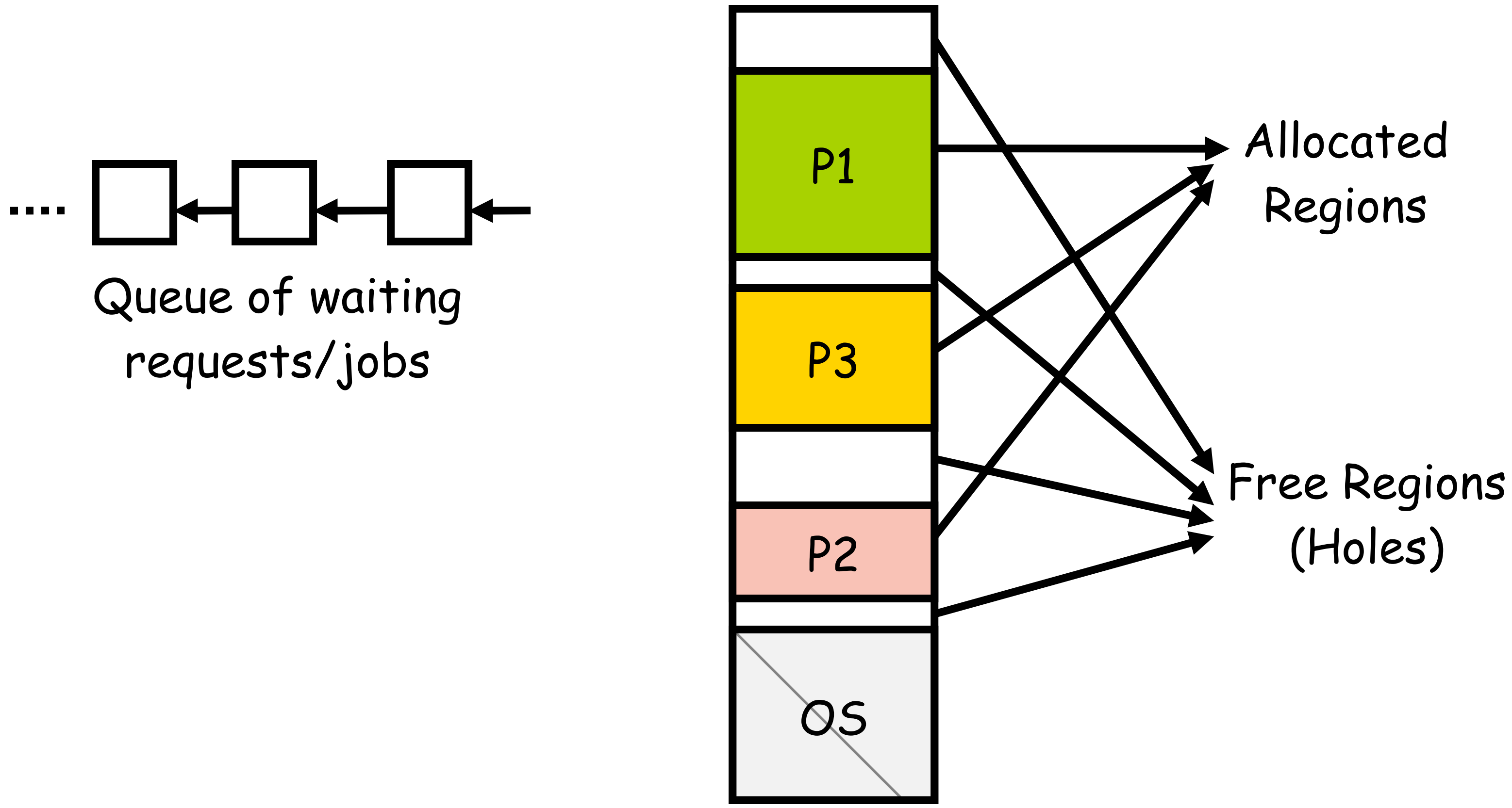
- Event ordering: happened-before relationship
- If events not related by happened-before then they can execute concurrently
- Lamport clock: counter incremented between any two successive events executed within a process
- Distributed Mutex: If $P_i$ is executing in its critical section, then no other process $P_j$ is executing in its critical section.
  - Centralized and Distributed approaches
- Two-phase commit protocol
- Generating unique timestamps: GUIDs
- Wait-die vs wound-ait schemes

# Main Memory

UNIVERSITY OF OREGON

- Swapping
- Allocation
  - Contiguous, Non-contiguous (paging)
  - Algorithms
- Fragmentation
  - Internal, External
- Page-tables, TLBs
  - virtual-physical translation
  - Page table structure, entries

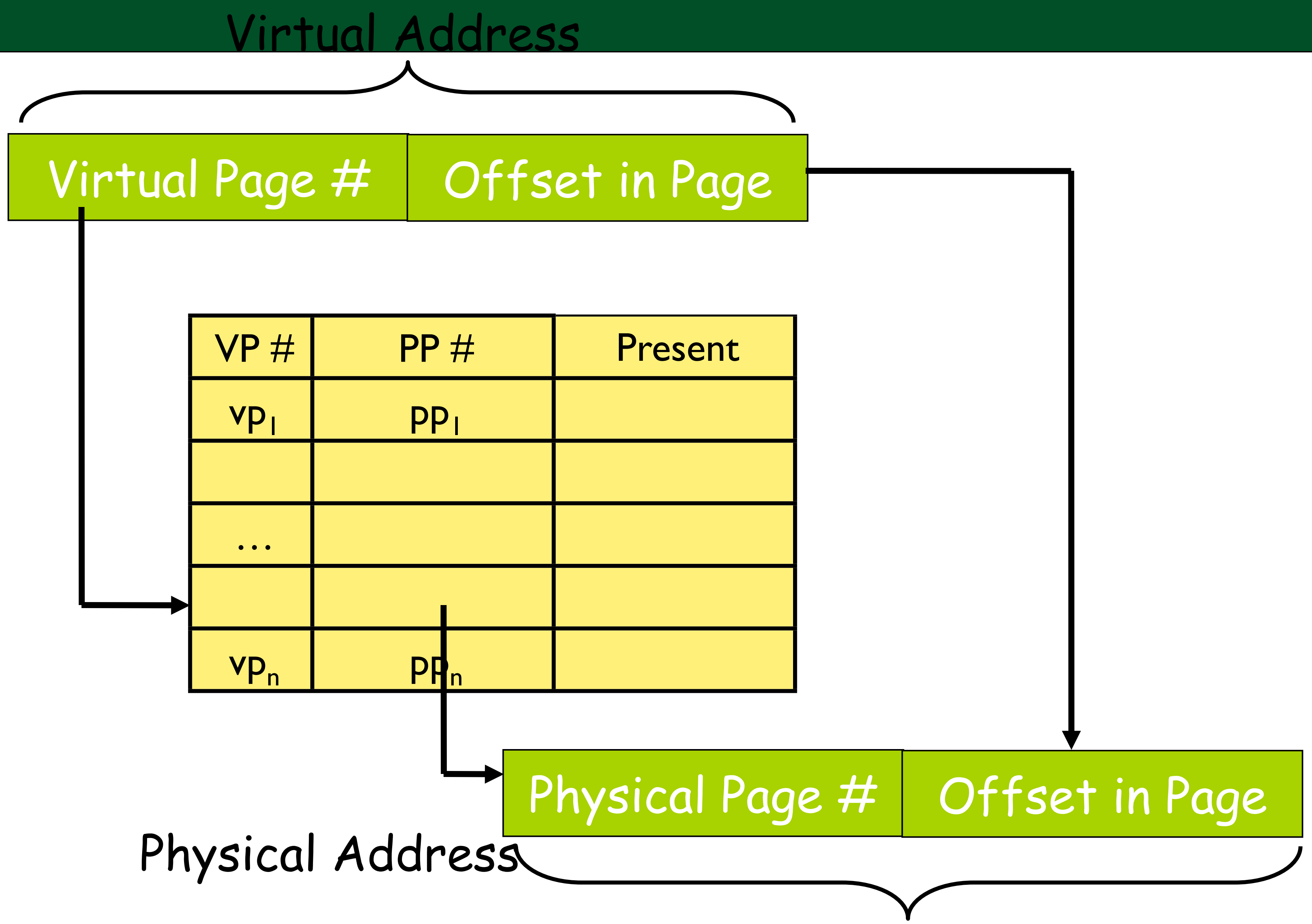
# Memory Allocation




**Question: How do we perform this allocation?**

- Programs are provided with a virtual address space (say 1 MB).
- Role of the OS to fetch data from either physical memory or disk.
  - Done by a mechanism called (demand) paging.
- Divide the virtual address space into units called "virtual pages" each of which is of a fixed size (usually 4K or 8K).
  - For example, 1M virtual address space has 256 4K pages.
- Divide the physical address space into "physical pages"

# Page Tables



Virtual Address
Virtual Page #
Offset in Page
VP #
PP #
Present
vp1
PP1
...
vpn
PPn
Physical Page #
Offset in Page
Physical Address

# Virtual Memory

UNIVERSITY OF OREGON

- Page Fault Handling
  - Performance Estimations
- Memory Initialization
- Page Replacement
  - Algorithms
  - Belady's Anomaly
- Uses of Virtual Memory
  - COW, Shared Pages, Memory-mapped Files
- Thrashing

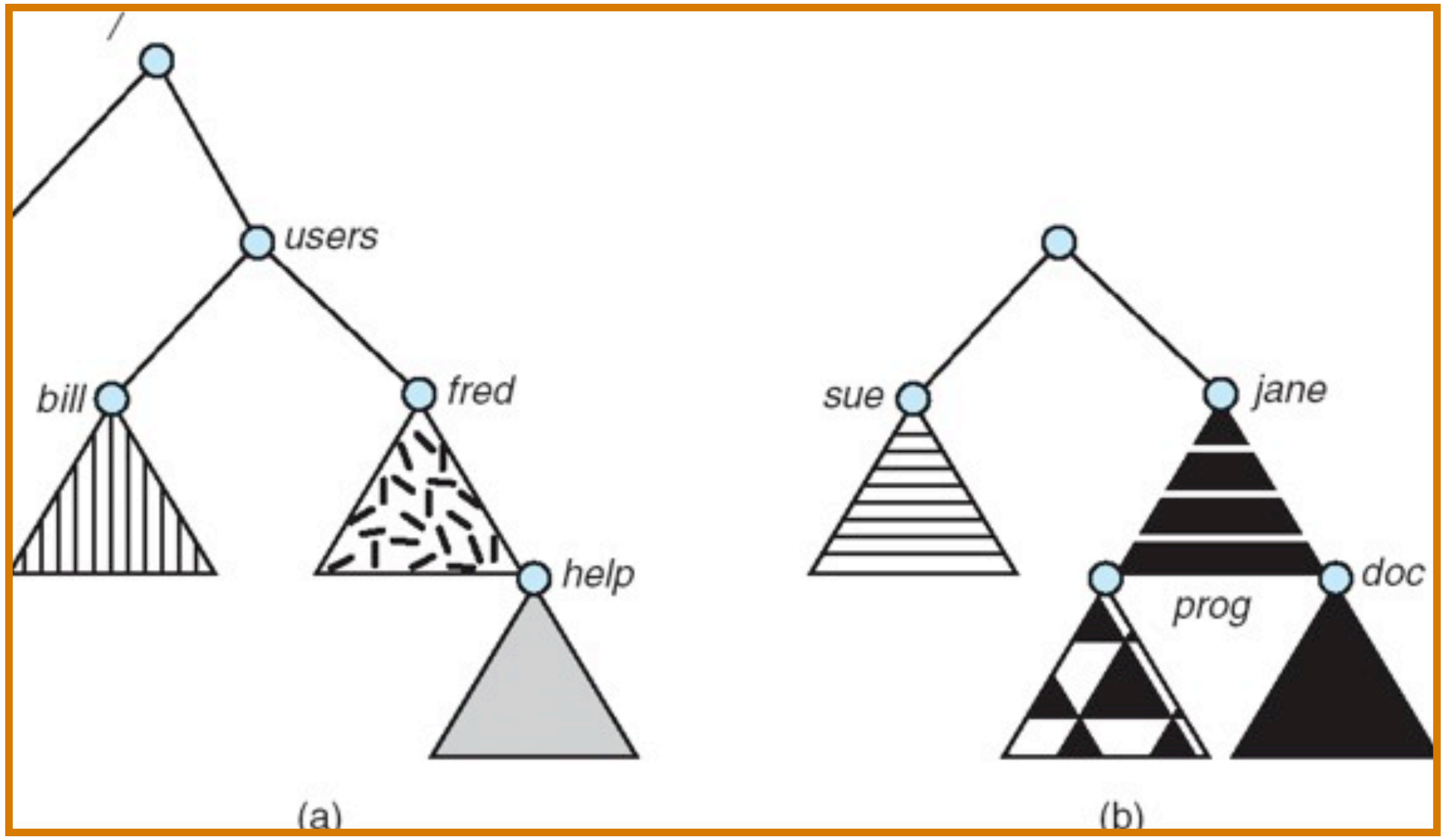
# Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system:
  - page fault
- Operating system looks at another table to decide:
  - Invalid reference -- abort
  - Just not in memory
- Get empty frame
- Swap page into frame
- Reset tables
- Set validation bit = v
- Restart the instruction that caused the page fault

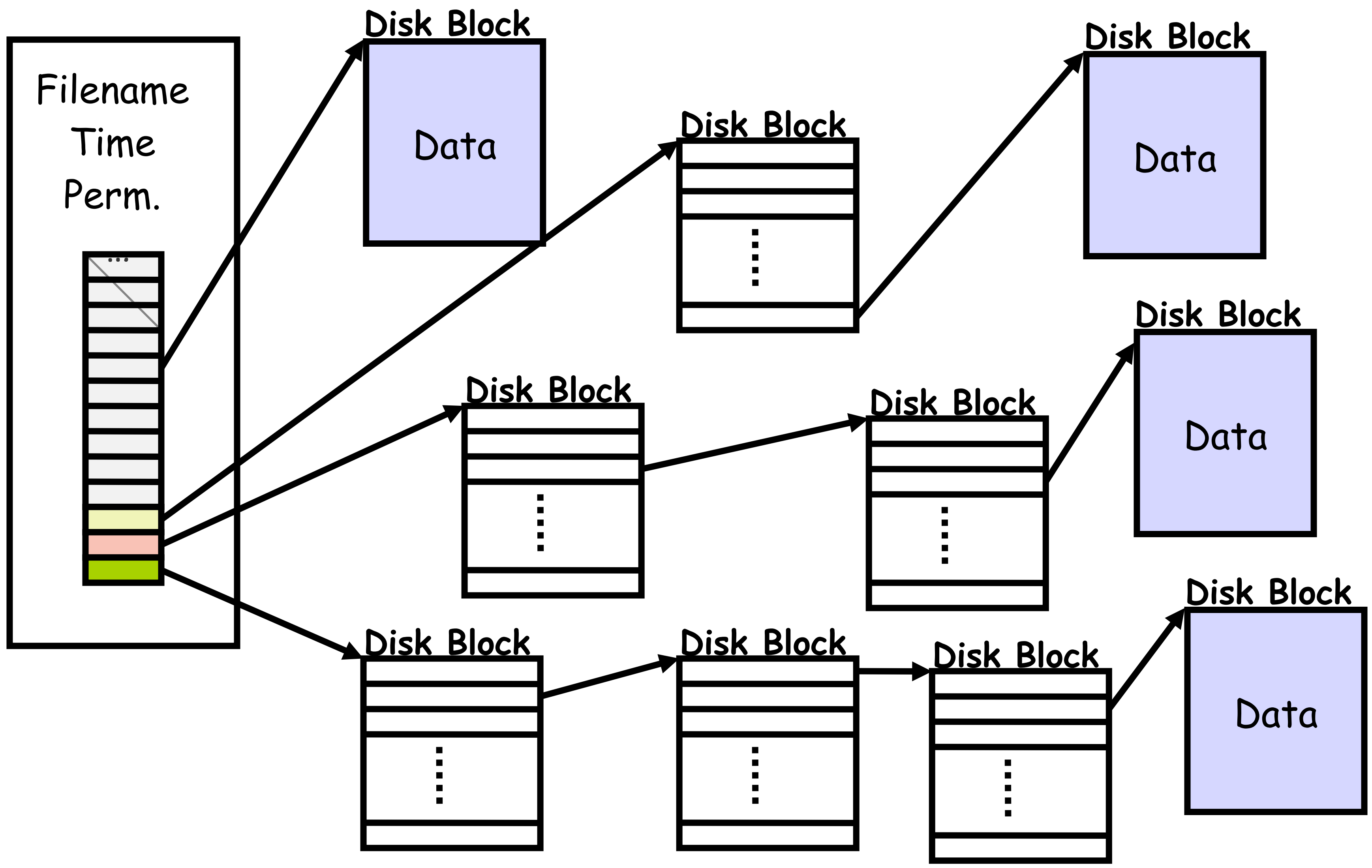
# File Systems

- File System Concepts
  - Files, Directories, File Systems
  - Operations and Usage
  - Remote File Systems
- File System Implementation
  - What's on the disk?  How's it formatted?
  - What's in memory?  How's it represented?
- File System Usage
  - Get a file
  - Caching
  - Free Space
  - Recovery

# File System Mounting



# i-node


Filename
Time
Perm.
...
Disk Block
Data
Disk Block
Disk Block
Data
Disk Block
Disk Block
Disk Block
Data
Disk Block
Disk Block
Disk Block
Disk Block
Data

# Access Control: Mode Bits

UNIVERSITY OF OREGON

- Three classes of users: public, group, owner
- Three types of access permissions:
  - read, write, execute

- Example:

| | Owner | Group | Public |
|---|---|---|---|
| rwx= | 111 | 101 | 101 |
| Octal | 7 | 5 | 5 |

What if no exec access and only owner can read/write?

# Access Control/Authorization

- An ***access control*** system determines what *rights* a particular *entity* has for a set of *objects*
- It answers the question
  - E.g., do you have the right to read /etc/passwd
  - Does Alice have the right to view the CIS website?
  - Do students have the right to share project data?
  - Does Prof. Butler have the right to change your grades?
- An *Access Control Policy* answers these questions

# Mass Storage and I/O



- Disk scheduling algorithms
- Access and transfer time (and their components)
- Real schedulers and their differences
- Buses
- DMA

# Looking forward...

UNIVERSITY OF OREGON

- What is the future of operating system design and research?
  - Mobile phones (Android is a big research platform right now)
  - Clouds (Amazon AWS and other services)
  - Both cases: distributed operation is becoming ever more critical
    - IPC within system and to other components that comprise logical systems
    - ubiquitous computing and prevalent network connectivity
- SECURITY

# Planning your UO career

UNIVERSITY OF OREGON

- If you are interested in what you've learned in this class and want to consider learning more about systems concepts, you may also think about the following courses:
  - CIS 432: Networking
  - CIS 433: Computer and Network Security
- Also: seminars and reading groups
- Think about your future and what you want
  - Take advantage of resources you have at your disposal while you're a student

# Thanks to...

UNIVERSITY OF OREGON

- Colleagues here at UO, at the Pennsylvania State University, the University of Pennsylvania, and Columbia University
  - provided basis for many course materials and course ideas
  - special thanks: Trent Jaeger and Adam Aviv
- Getty Ritter and Zhiyi Woo: GTF for this course
- Joe Pletcher and Hannah Pruse: course oracles
- You! Hope you learned something from it all

- Have a great summer! Good luck Wednesday!