CMSC421: Principles of Operating Systems

Nilanjan Banerjee

Assistant Professor, University of Maryland
Baltimore County
nilanb@umbc.edu
http://www.csee.umbc.edu/~nilanb/teaching/421/

Acknowledgments: Some of the slides are adapted from Prof. Mark Corner and Prof. Emery Berger's OS course at Umass Amherst

Announcements

- Homework 3 will be out tomorrow
- Grades will be out for the midterm and project 1 before end of weekend.

Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high
  - Page fault to get page
  - Replace existing frame
  - But quickly need replaced frame back
  - This leads to:
    - Low CPU utilization
    - Operating system thinking that it needs to increase the degree of multiprogramming
    - Another process added to the system
- Thrashing ≡ a process is busy swapping pages in and out

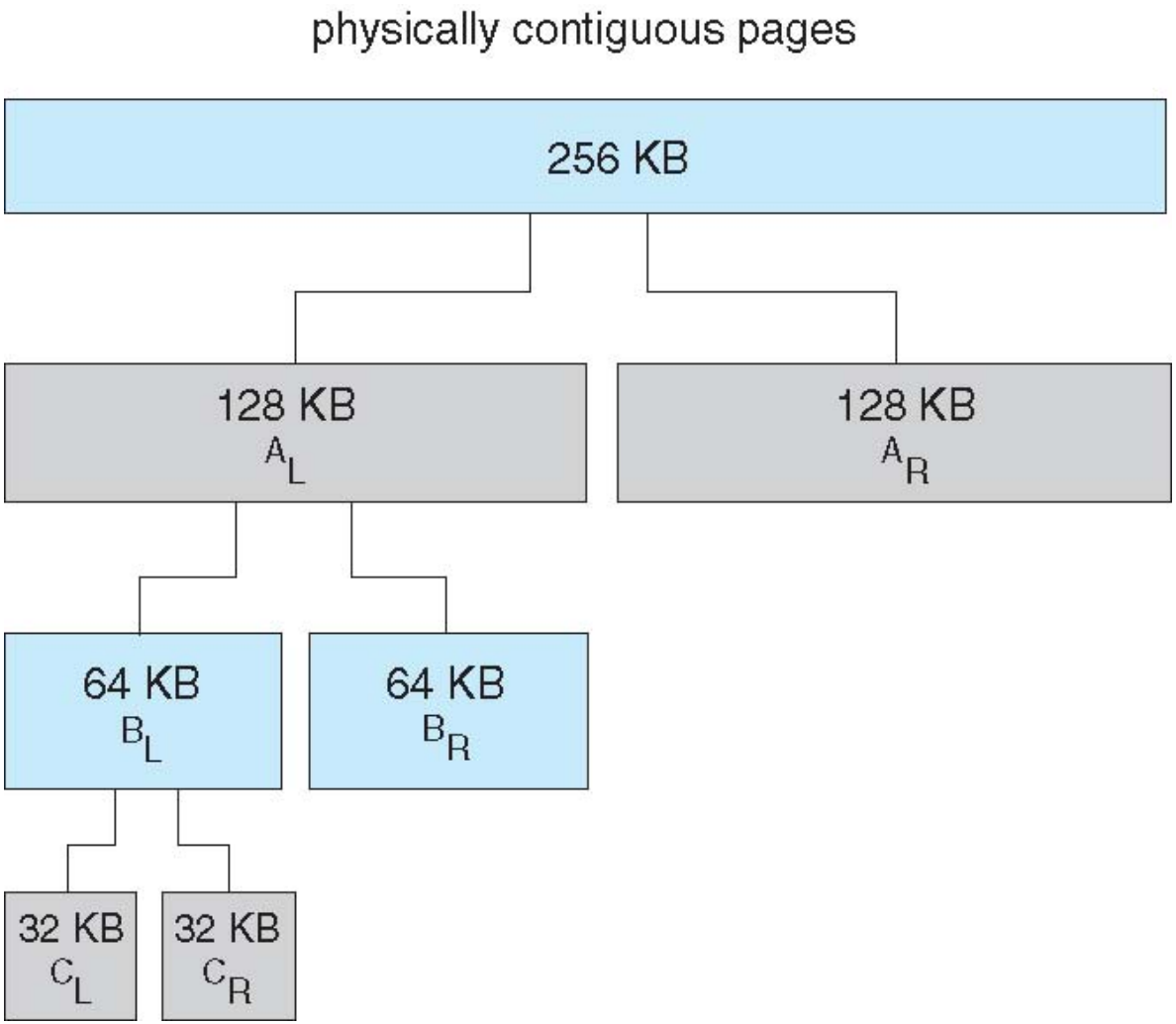
Allocating Kernel Memory

- Treated differently from user memory
- Often allocated from a free-memory pool
  - Kernel requests memory for structures of varying sizes
  - Some kernel memory needs to be contiguous
    - I.e. for device I/O

Buddy System

- Allocates memory from fixed-size segment consisting of physically-contiguous pages
- Memory allocated using **power-of-2 allocator**
  - Satisfies requests in units sized as power of 2
  - Request rounded up to next highest power of 2
  - When smaller allocation needed than is available, current chunk split into two buddies of next-lower power of 2
    - Continue until appropriate sized chunk available
- For example, assume 256KB chunk available, kernel requests 21KB
  - Split into A_L and A_r of 128KB each
    - One further divided into B_L and B_R of 64KB
      - One further into C_L and C_R of 32KB each – one used to satisfy request
- Advantage – quickly coalesce unused chunks into larger chunk
- Disadvantage - fragmentation

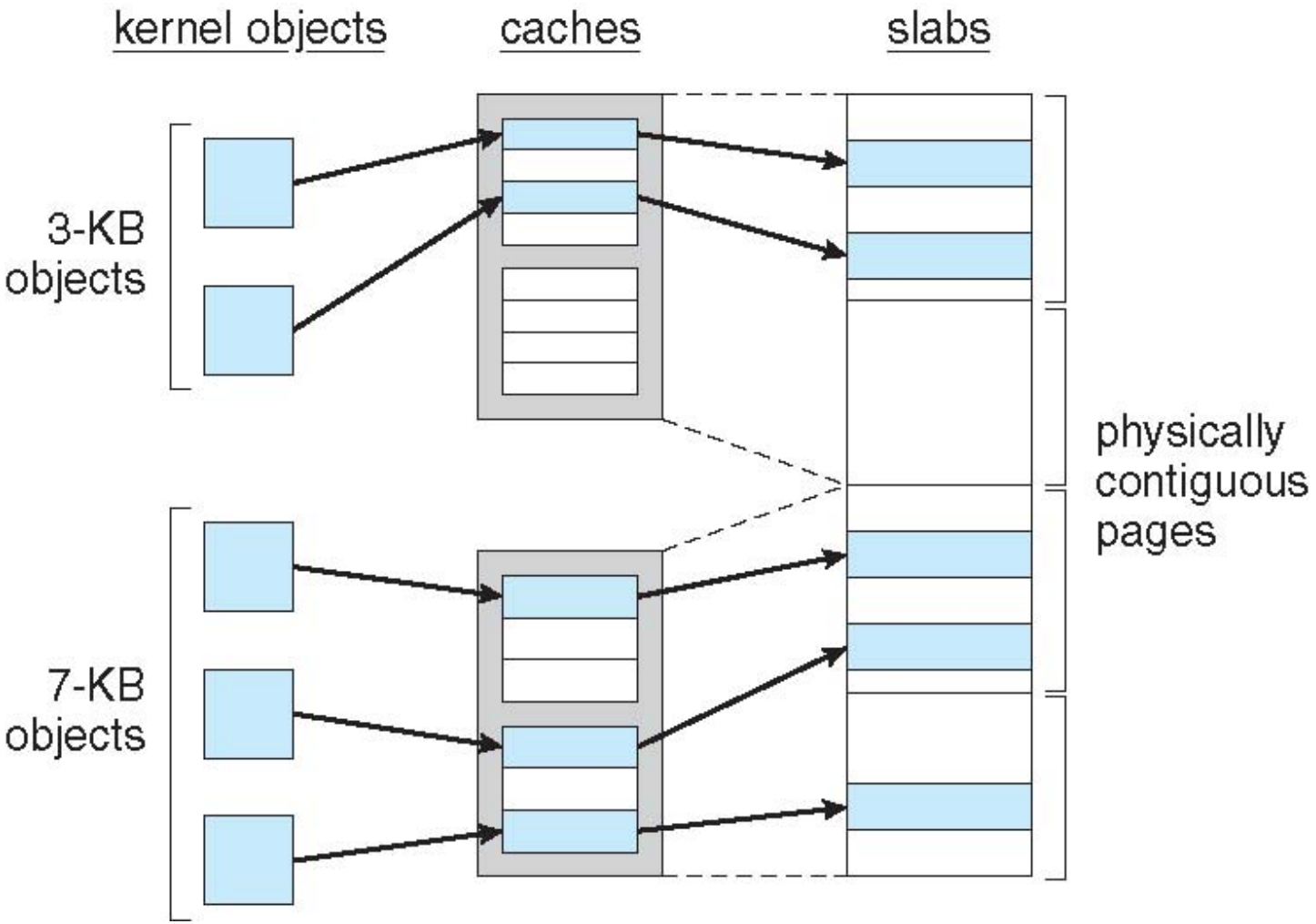
Buddy System Allocator



Slab Allocator

- Alternate strategy
- **Slab** is one or more physically contiguous pages
- **Cache** consists of one or more slabs
- Single cache for each unique kernel data structure
  - Each cache filled with **objects** – instantiations of the data structure
- When cache created, filled with objects marked as **free**
- When structures stored, objects marked as **used**
- If slab is full of used objects, next object allocated from empty slab
  - If no empty slabs, new slab allocated
- Benefits include no fragmentation, fast memory request satisfaction

Slab Allocation

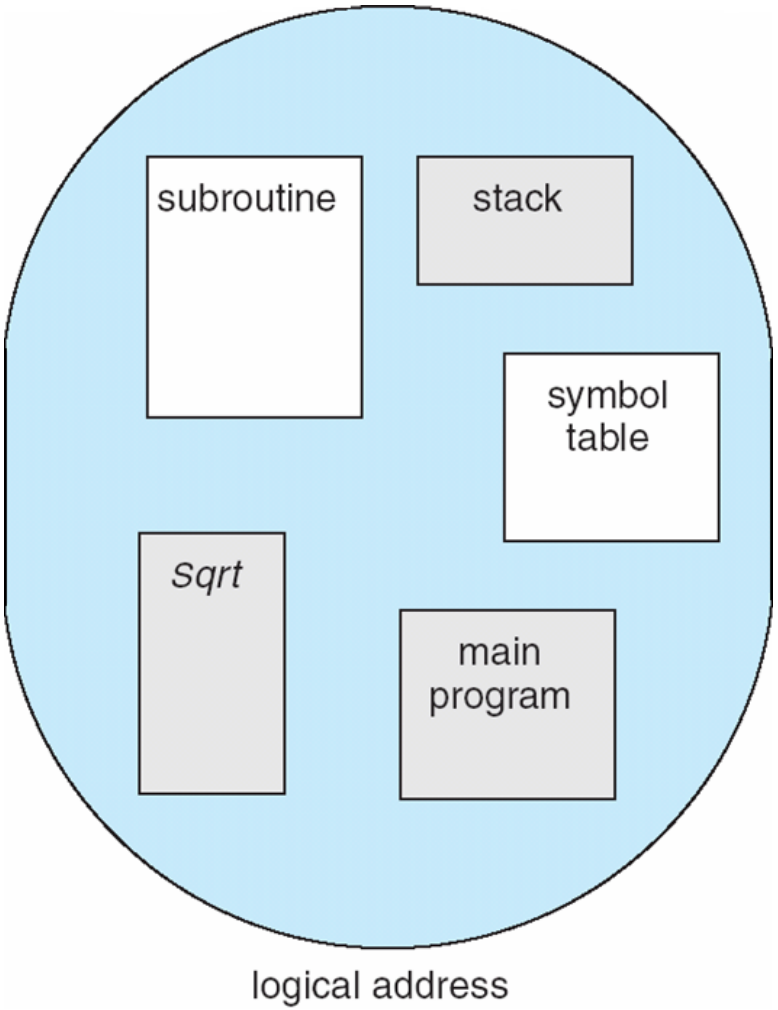


Segmentation (another method of implementing virtual memory)

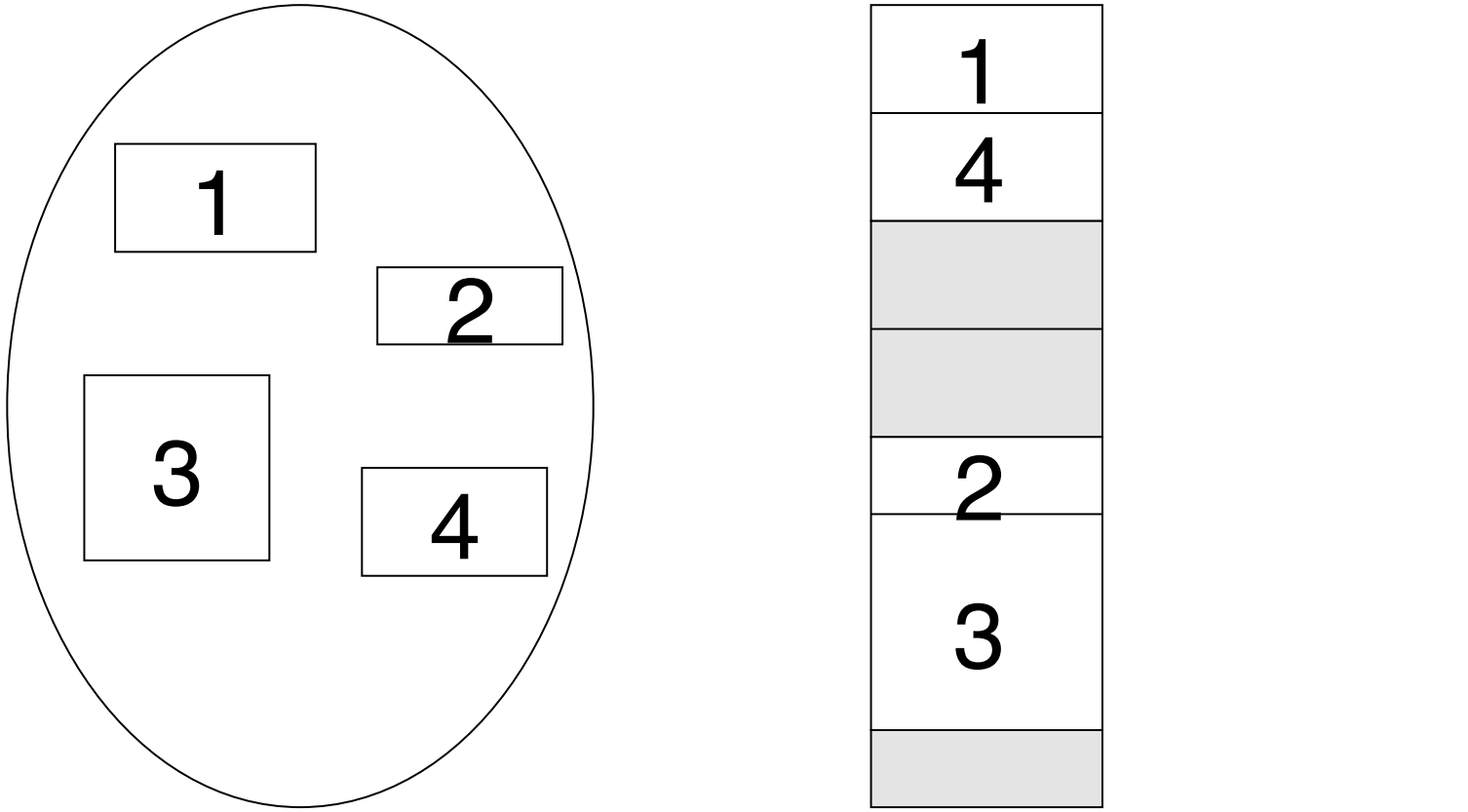
- Memory-management scheme that supports user view of memory
- A program is a collection of segments
  - A segment is a logical unit such as:

    main program
    procedure
    function
    method
    object
    local variables, global variables
    common block
    stack
    symbol table
    arrays

User's View of a Program



Logical View of Segmentation



Segmentation Architecture

- Logical address consists of a two tuple:

  <segment-number, offset>,

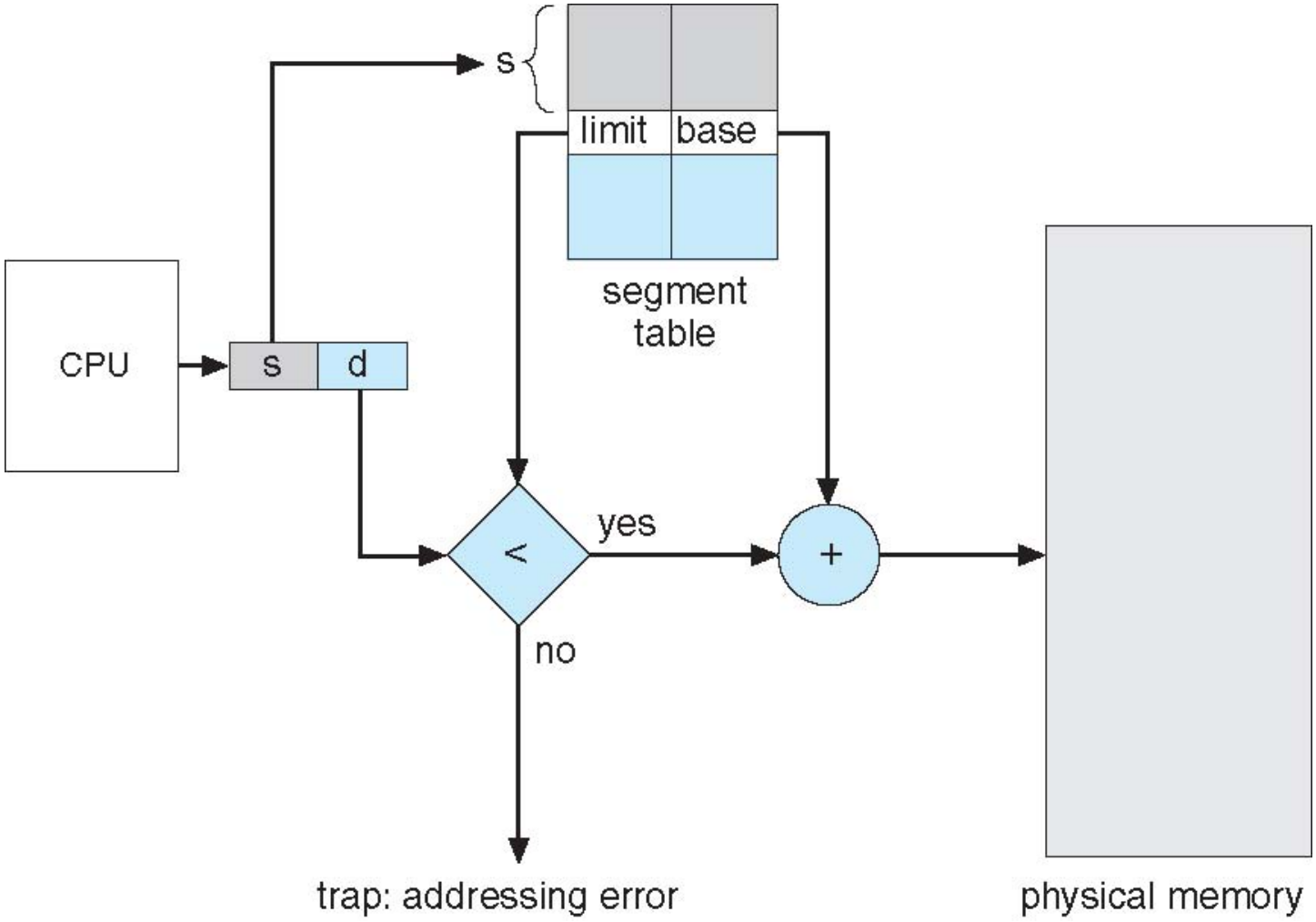
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;

  segment number ***s*** is legal if ***s*** < **STLR**

Segmentation Architecture (Cont.)

- Protection
  - With each entry in segment table associate:
    - validation bit = 0 ⇒ illegal segment
    - read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem
- A segmentation example is shown in the following diagram

Segmentation Hardware



File Systems (Lets start with the disk)

- Disk (hard drive) is a block device
  - You can read and write blocks from the hard drive
  - E.g. give me block number 50, or block number 100
  - Blocks are usually 1KB in size

- You can also create logical block sizes
  - E.g. using the command dd
  - Example of creating files without file system (demo?)

- You can write file systems for block devices (e.g., cdrom, harddrive, flash drives)
- Another type of devices is character devices?
  - Examples?
  - What are the major differences between char and block devices

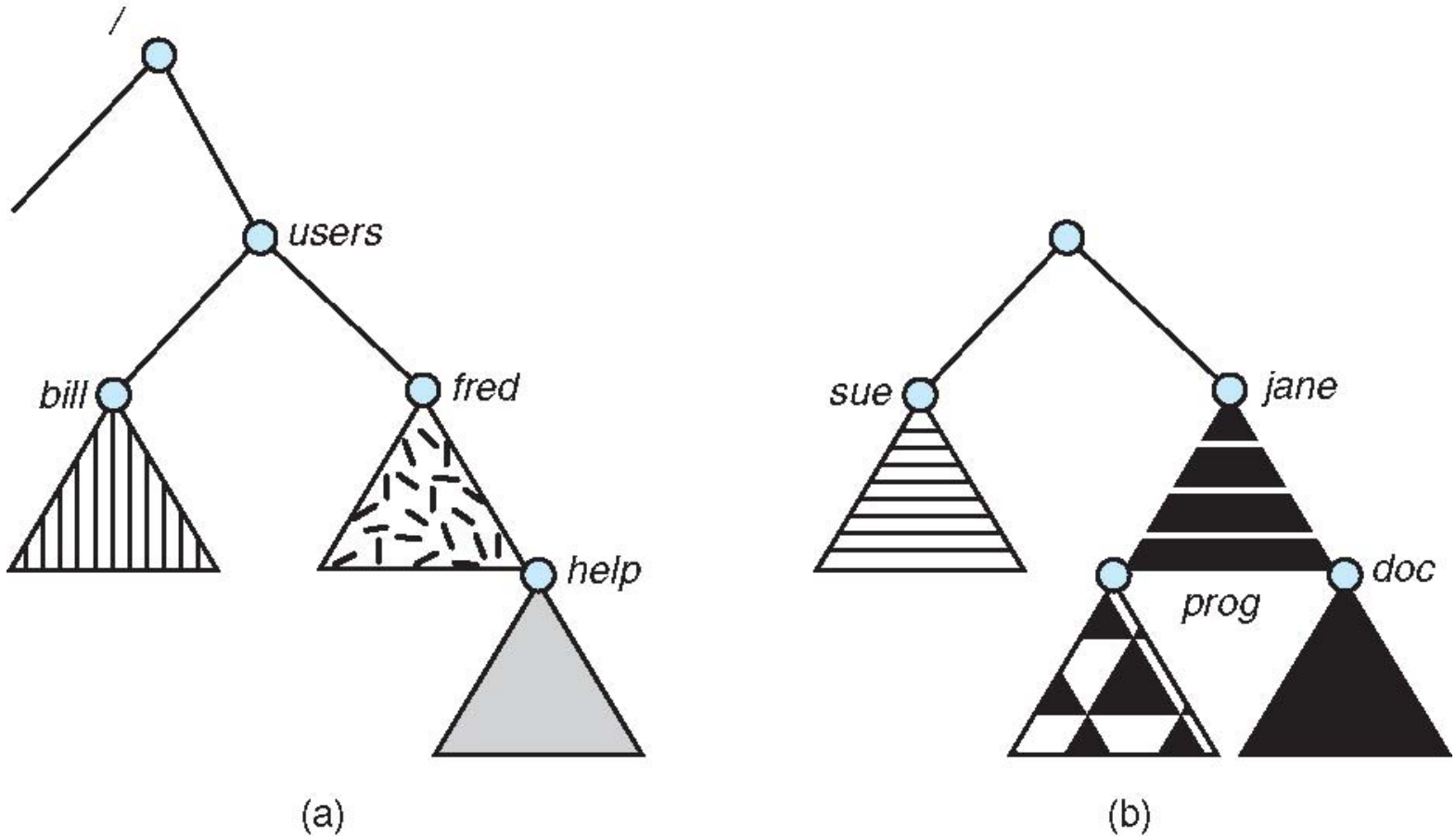
File system structure and file manipulations

- File systems are made of directories
  - In linux the root directory is /
- All directories are children of some directory
  - Directories follow a tree structure
- Directories consist of files
  - We will later talk about how files and directories are represented
- Files are associated with two things
  - Name of the file
  - Pointer to the data stored in the file

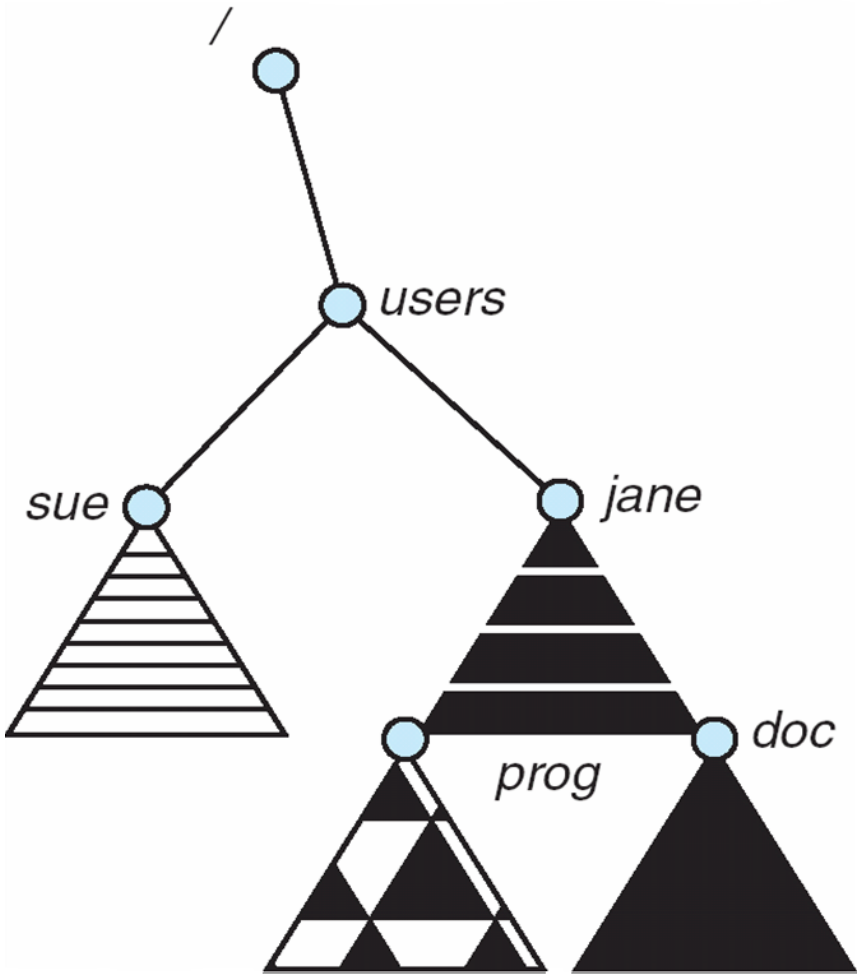
Concept of virtual file systems

- In linux you can use the concept of mounting to add a custom file system to your directory tree
- You can also mount directories on remote machines onto your file system tree
- Lets take a look at a demo

(a) Existing (b) Unmounted Partition



Mount Point



Common file operations (you might be familiar with)

- Creating directories
- Removing directories
- Creating a file
- Open and reading a file
- Deleting a file
- Creating a soft link to a file
- Creating a hard link to a file
- Append a file
- Read last few bytes/characters of a file

Protection

- File owner/creator should be able to control:
  - what can be done
  - by whom

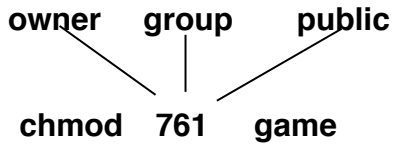
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	RWX 1 1 0
c) **public access**	1	⇒	RWX 0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



In-class Discussion