Review Lecture

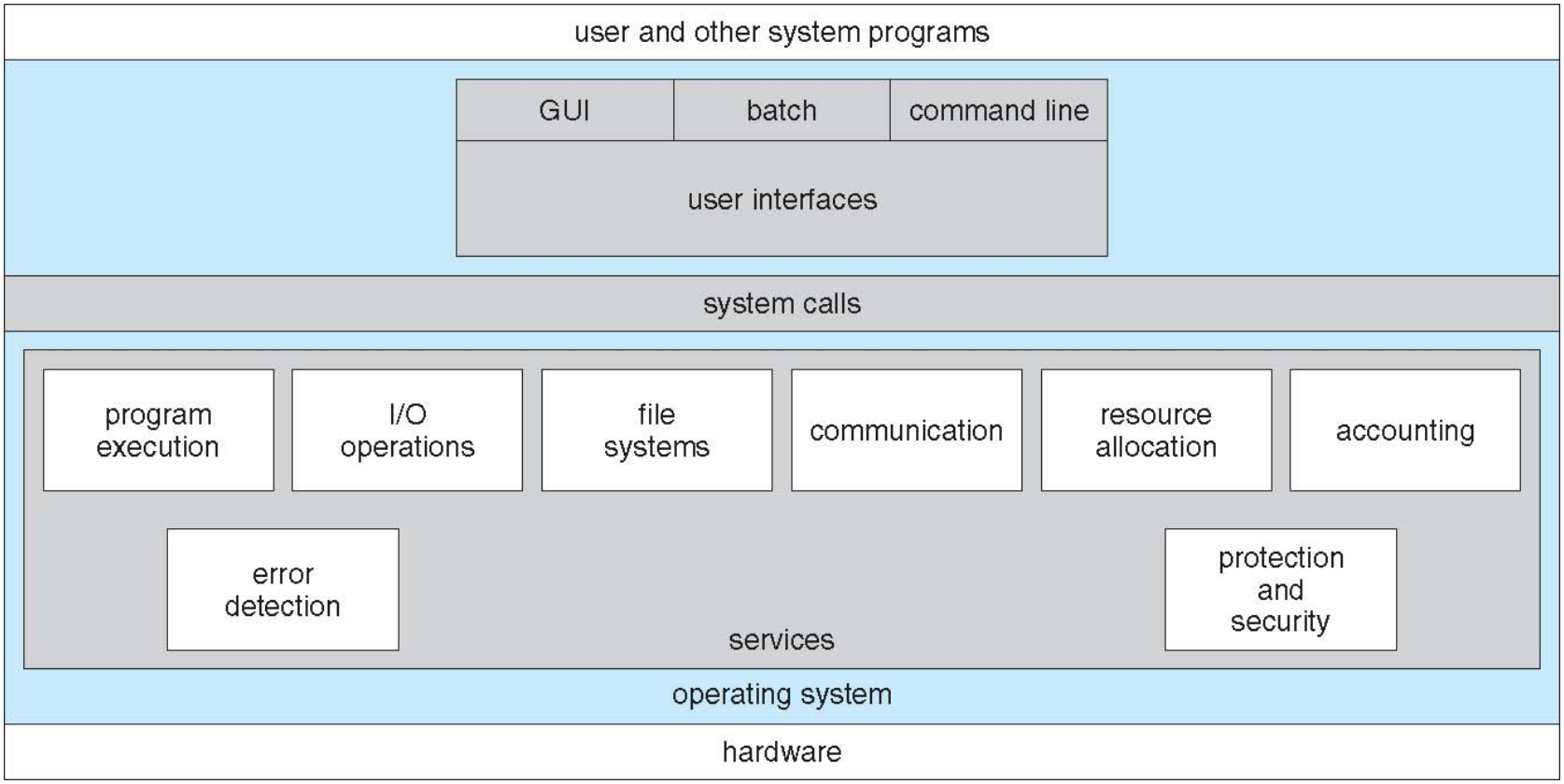
CE2005/CZ2005 and CPE205/CSC205 Operating Systems Course Revision

Overview (Part 1)

- Understand the evolution of OS
  - Batch, multi-programmed, time-sharing systems
- Understand the purpose and goals of OS
  - Resource manager, control program, kernel
  - Efficiency and convenience
- Understand the Components of OS
  - Process, memory, file, I/O, secondary storage…
- Basic Concepts
  - Interrupts, traps, system calls

Overview (Part 1)

- OS Components



Overview (Part 1)

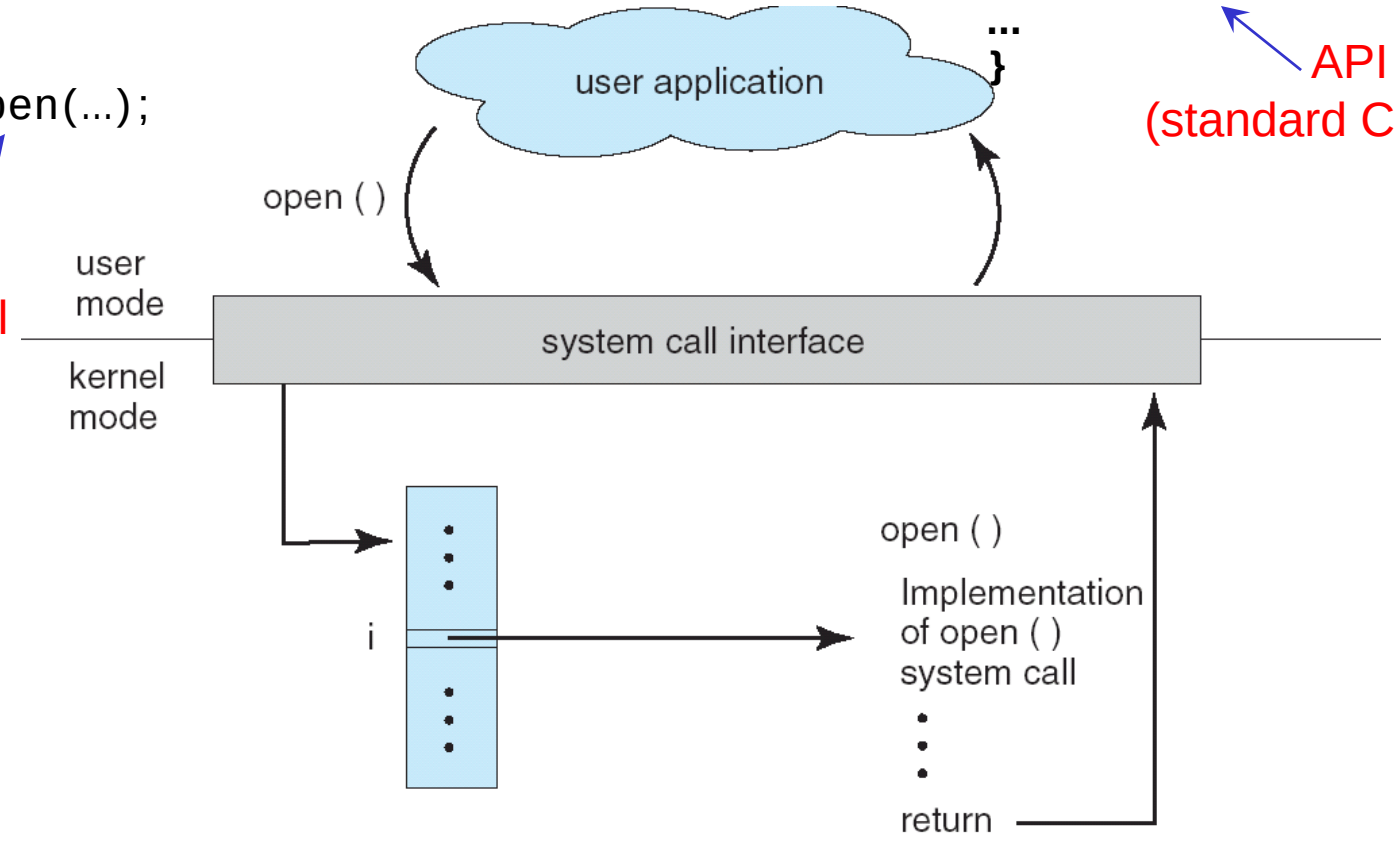
- System Call

```
FILE *fopen(...)
{
Int fd;
...
fd = open(...);
...
}
```

System Call

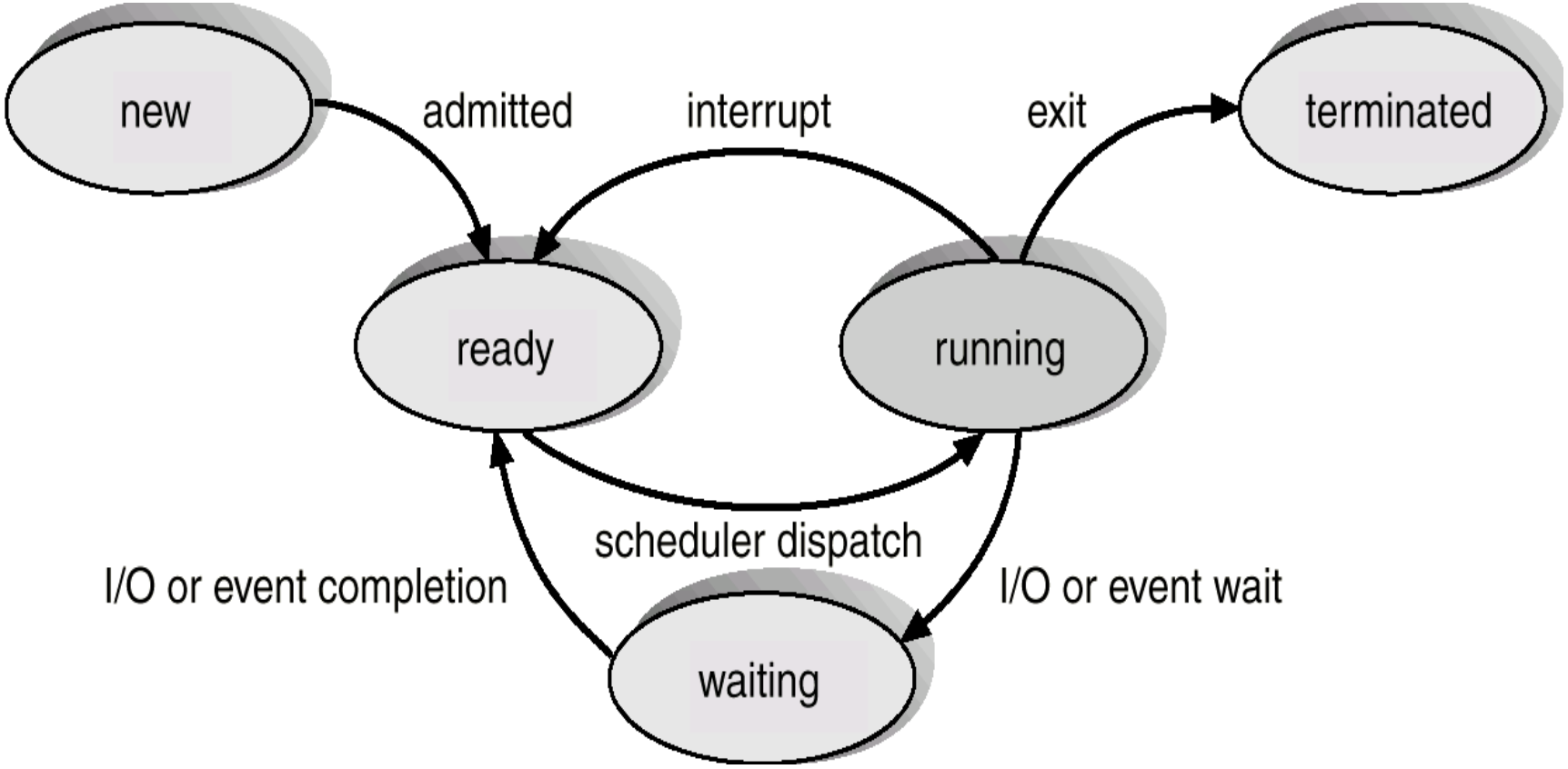
```
#include <stdio.h>
main()
{
FILE *fp;
...
fp = fopen(...);
...
}
```

API
(standard C library)



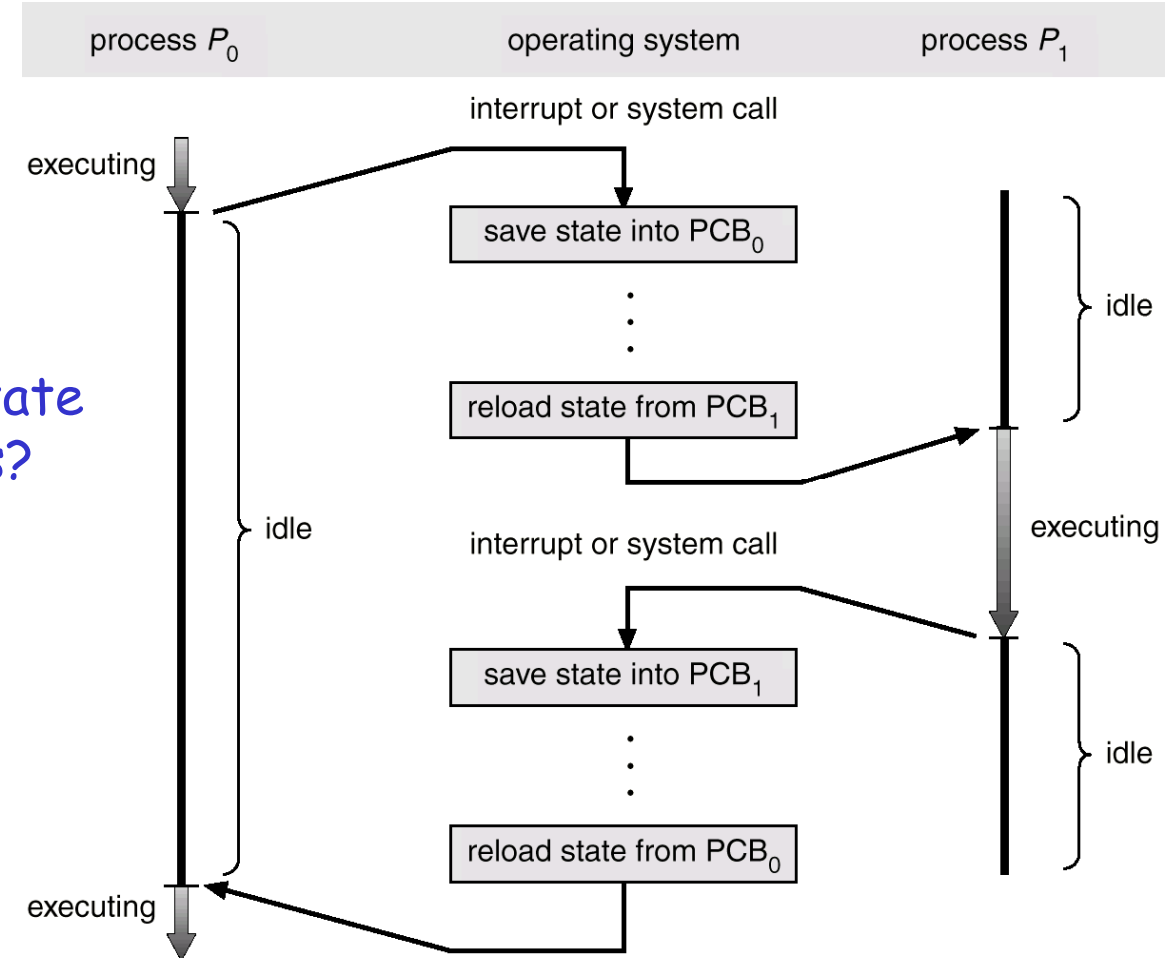
Part 2: Processes

- Process State Transition Diagram



Part 2: Processes

- Process Context Switch



Possible state transitions?

Part 2: Processes

- Level of Scheduling
  - Long-term, medium-term, short-term scheduling

Definitions and objectives?

- Inter-Process Communication
  - Direct, indirect communication
  - Message buffering

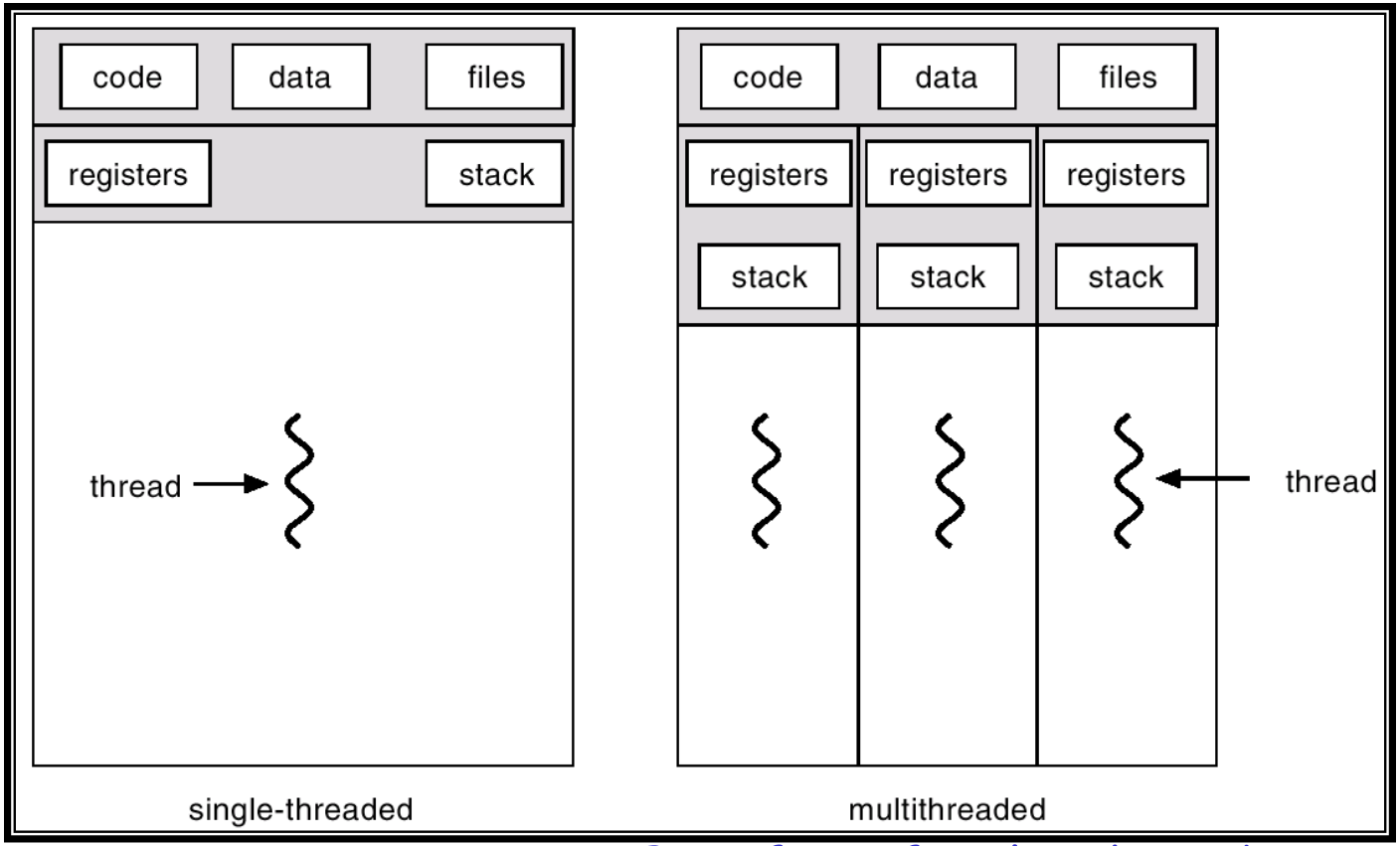
How can communication link be established?

- Basic Concepts
  - Process address space and execution context
  - I/O bound, CPU bound processes
  - Independent, cooperative processes

What are in PCB?

Part 2:Threads

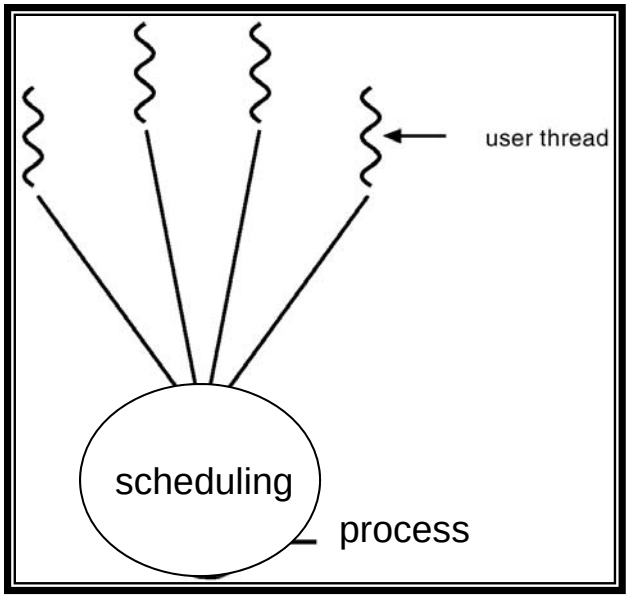
- Thread vs. Process

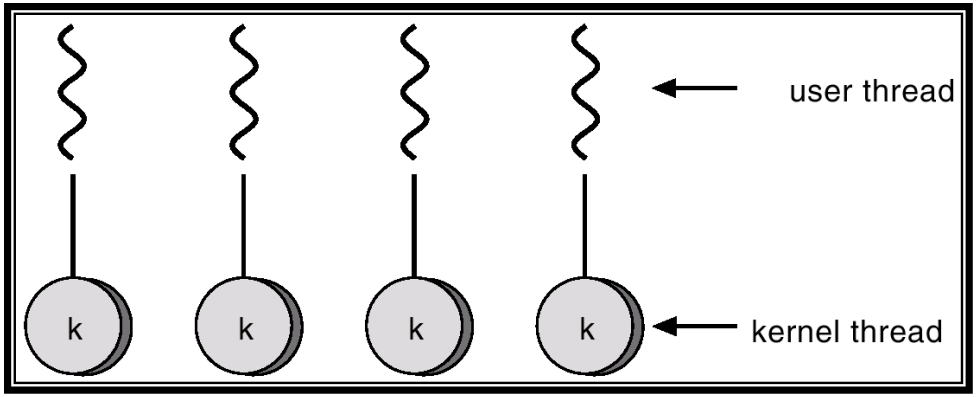


Benefits of multi-threading?
Why?

Part 2:Threads

- User Thread vs. Kernel Threads





Part 3: CPU Scheduling

- Basic Concepts
  - CPU-I/O burse cycle
  - Non-preemptive, preemptive scheduling
- Scheduling algorithms
  - FCFS, SJF, SRTF, Non-Preemptive Priority, Preemptive Priority, RR

Sample question: given a sequence of processes with arrival times and cpu burst times, calculate average waiting time.

Part 4: Process Synchronization

- Critical Section Problem
  - Condition for a correct solution
  - Correct software solution for two processes
- Solution using TestAndSet

```
while (1) {
        while(TestAndSet(lock));
                critical section
        lock = false;
                remainder section
}
```

Solution using variation of TestAndSet() instruction (e.g., Swap())

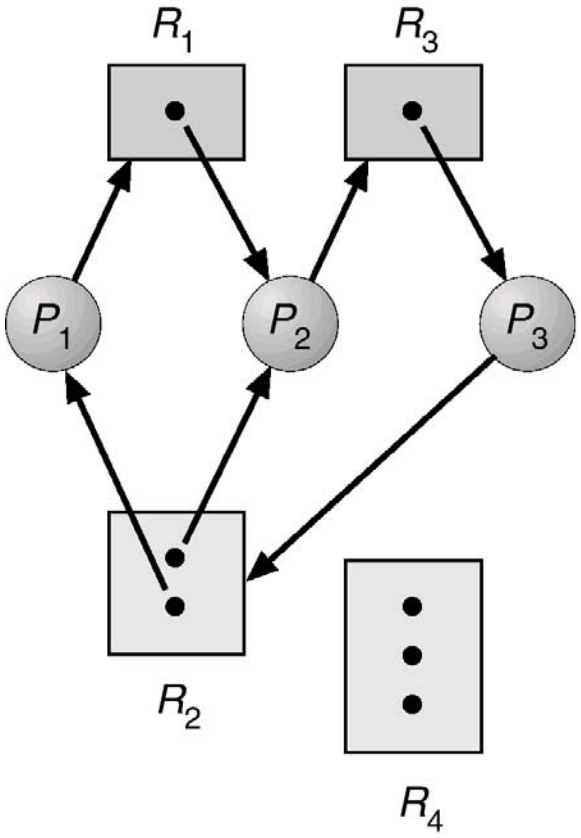
Part 4: Process Synchronization

- Semaphores
  - wait() and signal ()
  - Binary vs. counting semaphore
  - Using TestAndSet to implement semaphore

Definitions and usage

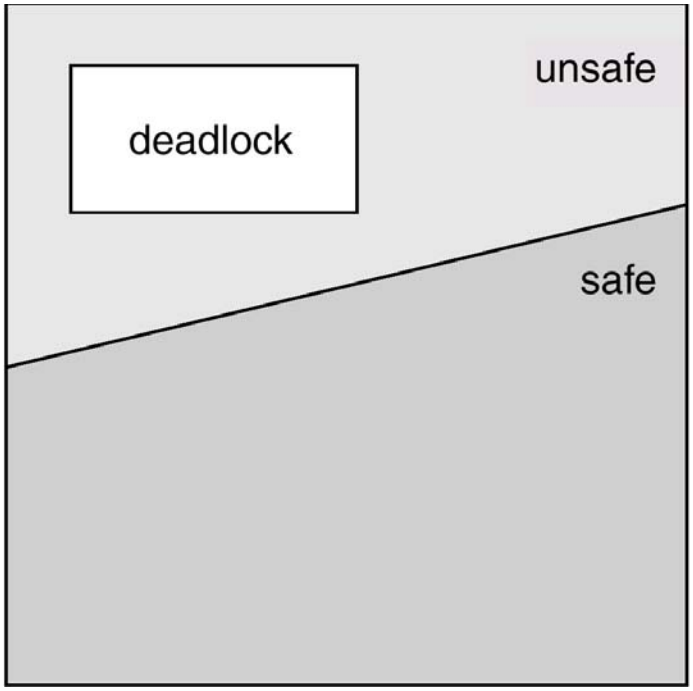
Part 5: Deadlocks

- Four necessary conditions
- Resource Allocation Graph



Part 5: Deadlocks

- Methods for handling deadlocks
  - Deadlock prevention
  - Deadlock avoidance
  - Deadlock detection and recovery (basic concept)
- Deadlock avoidance
  - Resource allocation state
  - Safe state
  - Algorithm to determine safe state
  - Banker's algorithm



Part 5: Deadlocks

- Deadlock detection & recovery
  - Deadlock detection
    - For single instance of each resource
    - For multiple instances of each resource
  - Deadlock recovery
    - Resource preemption vs. process termination
    - Selection of process

Deadlock detection algorithm

Part 6: Memory Organization

- Binding of code and data to memory addresses
  - Compile time
  - Load time
  - Execution time
- Logical vs. physical address space
  - Identical if binding is done at compile-time or load-time
  - Different if binding is done at execution-time
    - ❄ The hardware device (MMU) to map logical to physical address

Part 6: Memory Organization (Cont.)

- Memory allocation schemes
  - Contiguous allocation
    - ❄ Fixed partitioning (internal fragmentation)
    - ❄ Dynamic partitioning (External fragmentation)
      - Dynamic storage allocation strategies: first-fit, best-fit, and worst-fit
      - Compaction to reduce external fragmentation

Sample question: given a memory map and a sequence of memory requests, how to satisfy these requests by using first-fit, best-fit, or worst-fit? If compaction is needed, how to shuffle partitions?

Part 6: Memory Organization (Cont.)

- Memory allocation schemes (cont.)
  - Non-contiguous allocation (paging)
    - External fragmentation eliminated
    - Internal fragmentation possible
    - Logical address format (p, d): How to derive it given the logical address space size and the page size?
    - Address translation from logical to physical address by using page table: given a page table, how to translate a logical address to the physical address?

Part 6: Memory Organization (Cont.)

- Memory allocation schemes (cont.)
  - Non-contiguous allocation (paging) (cont.)
    - Implementation of page table
      - TLB: given TLB lookup time, memory access time, and hit ratio, how to compute EAT?
    - Two-level page table
      - Why is it introduced?
      - Logical address format (p1, p2, d): how to derive it given the logical address space, page size, and entry size? Given the logical address format, how to translate a logical into physical address by using two-level page table?

Part 6: Memory Organization (Cont.)

- Memory allocation schemes (cont.)
  - Non-contiguous allocation (paging) (cont.)
    - Inverted page table
      - How is it different from a normal page table?
      - Its pro and con
    - Page sharing
  - Non-contiguous allocation (segmentation)
    - Logical address format: (segment-no, offset)
      number of bits for offset is determined by the maximum segment size
    - Address translation: base + offset
  - Swapping

Part 7: Virtual Memory

- The motivation of virtual memory
- Virtual memory is implemented by demand paging
- A page fault triggers demand paging: the whole process of demand paging?
- Performance of demand paging: given memory access time, page fault time, and page fault rate, how to compute EAT?

Part 7: Virtual Memory (cont.)

- Page replacement is essential to virtual memory
- Page replacement algorithms: FIFO, Optimal, LRU, Second-chance (Clock)
  - Given a reference string, how it works
  - The performance in terms of page fault rate
  - Belady's Anomaly problem

Sample question: given a reference string and the number of frames, how does each of them work? Draw the memory map after each page reference and indicate when page faults occur.

Part 7: Virtual Memory (cont.)

- Allocation of frames
  - Fixed vs. variable allocation
  - Local vs. global replacement
- Thrashing
  - What is thrashing and why it happens?
  - How to combat it: Working-set model and page fault frequency

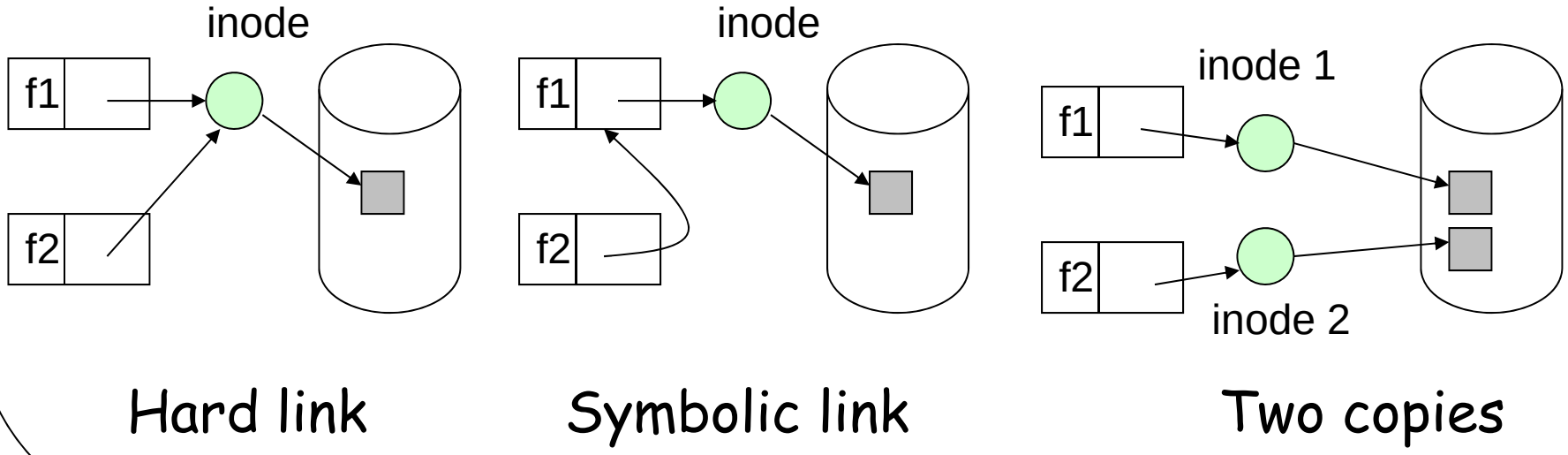
Sample question: given the size of a physical address space and page size, decide whether a program fits in the memory space. What happens if the page size changes?

Part 8: File Systems

- File structure
  - Structured vs. unstructured files
- File access methods
  - Sequential access
  - Random access
- Directory organization
  - Required features: efficiency, naming, and grouping
  - One-level, two-level, tree-structured, acyclic graph

Part 8: File Systems (cont.)

- Directory organization (cont.)
  - Only acyclic graph structure supports *file sharing*
    - A shared file is different from two copies of the same file
    - Hard link, symbolic link, and two copies



Part 8: File Systems (cont.)

- File Protection
  - UNIX
    - Three classes of users: owner, group, public
    - Access rights for each class of users: (r, w, x)

Sample question 1: how to set access rights if the owner can read, write and execute; the group can read and execute; everybody else can only read? -rwxr-xr--

Part 8: File Systems (cont.)

- Three disk allocation schemes: contiguous, linked, and indexed allocation
  - Pros and cons of each scheme
  - In terms of file size and file access method, which scheme is particularly good for what kind of file?
  - Logical to physical address translation

Sample question 1: given the block size, pointer size (for linked allocation), file size and entry size (for indexed allocation), how to translate a logical address into the physical disk address? To access that address, how many disk blocks must be read for each scheme?

Whether the directory is in the memory affects the number of disk blocks read

Part 8: File Systems (cont.)

– Logical to physical address translation (cont.)

Sample question 2: for each scheme, if a block is added into or removed from a particular position within a file, how many disk I/O operations (read and write) are needed?

Whether the directory is in the memory affects the number of disk blocks read

For contiguous allocation, free blocks will be allotted to the end of a file so that the file can grow.

– FAT: a variant of linked allocation
  - ❄ Incorporate free-space management
  - ❄ Random access is improved (reading only the FAT, we can find the location of any block, without actually going through real disk blocks)

Part 8: File Systems (cont.)

- inode: an implementation of indexed allocation
  - Each file has an inode, which is stored in a fixed location on disk
  - inode structure (12, 1, 1, 1)

Sample question 1: given the file pointer size and block size, what is the maximum file size that can be supported by an inode?

Sample question 2: given a file path (absolute or relative), how to translate the path to inode? How many disk references are required to open this file (assuming only one disk block is needed for directory and file respectively)?

Part 9: I/O & Disk

- Efficiency and Generality
- Device-driver layer hides differences among I/O controllers from kernel
- Services provided by kernel I/O subsystem: I/O scheduling, buffering, caching, spooling, and error-handling.
- Major overheads in I/O

Part 9: I/O & Disk (cont.)

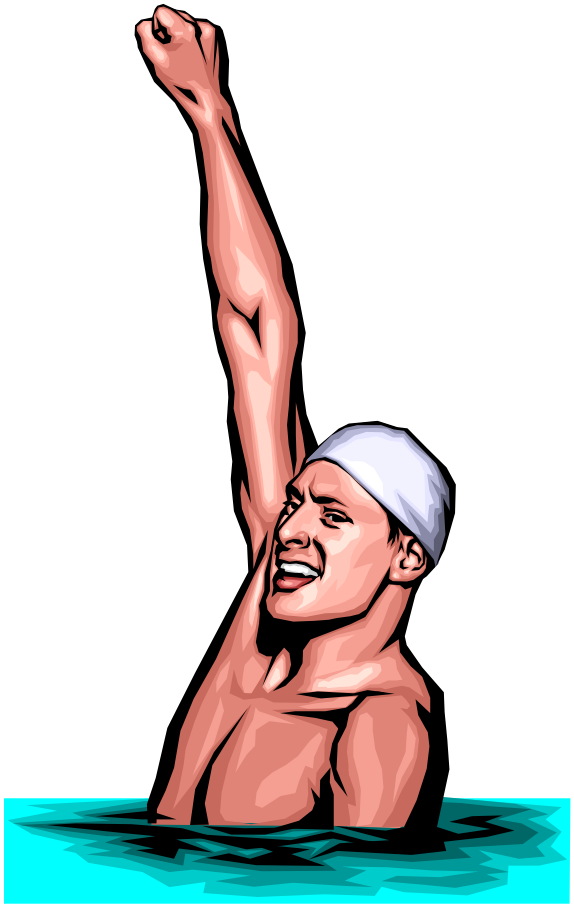
- Disk structure (cylinder, track, sector): array of blocks is mapped into sectors
- Disk access time: seek time + rotational latency + transmission time
- Disk scheduling is to reduce seek time and improve disk bandwidth

Part 9: I/O & Disk (cont.)

- Disk scheduling algorithms: FCFS, SSTF, SCAN, C-SCAN, C-LOOK
  - Pros and cons of each algorithm
  - Starvation problem
  - Considerations in choosing an algorithm

Sample question : given a disk with numbered cylinders and the head moving direction, for a request queue, how does each algorithm work and how many head movements are required for each algorithm?

Best wishes…



You

Consultation Hours:
22 Apr (Fri): 2:00pm – 5:00am
26 Apr (Tue): 2:00pm – 5:00pm
27 Apr (Wed): 9:00pm – 11:30am

Office:
PDCC (N4-02b-62)

Me