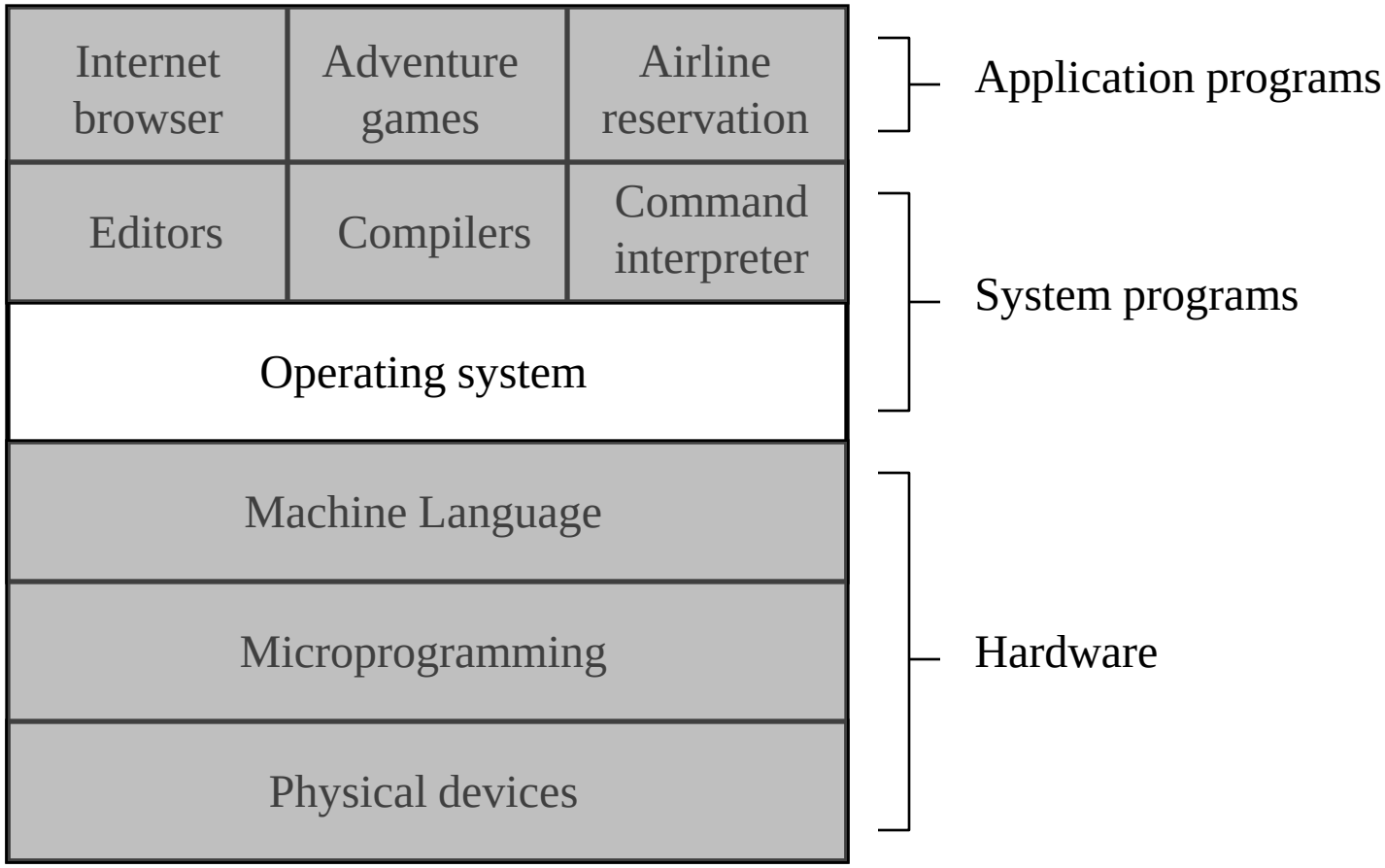
CHAPTER 3

Operating Systems (and Networks)

- Using hardware directly is highly complicated
  - even at the machine language level
- Especially so when multiple users want to perform multiple tasks - all at the same time
  - abstraction layer: *Operating System*

Prepared by Dr. Syed Khaldoon Khurshid

3.X: Computer System Overview

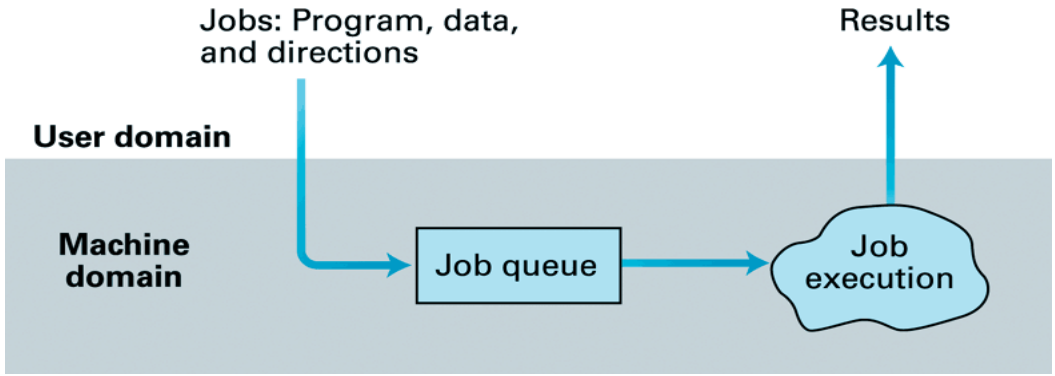


3.X: What is an Operating System?

- Top-down view: operating system is there to present the user with the equivalent of a '*virtual machine*'
  - hardware is difficult to program
  - user should not be annoyed with low level details
  - OS =>high-level abstractions (files, device access,.)
- Bottom-up view: operating system is there to manage all the pieces of a complex system
  - orderly, controlled management of *multiple programs* running at the same time
  - if needed: orderly, controlled management of *multiple users* at the same time

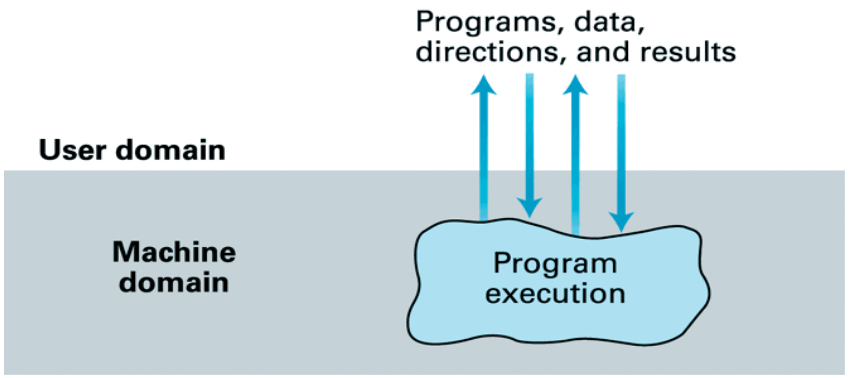
3.1: The Evolution of Operating Systems (1)

- 1945-1955:
  - User/programmer was ‘operating system’
- 1955-1965:
  - Human operator was ‘operating system’
  - Advent of *‘batch processing’*:



3.1: The Evolution of Operating Systems (2)

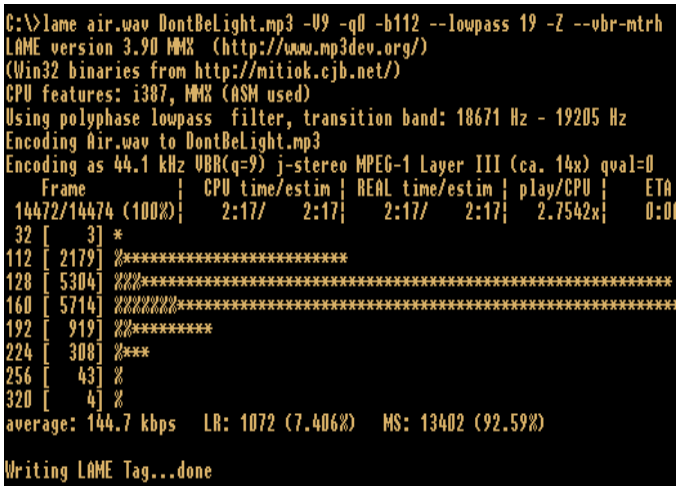
- 1965-1980:
  - Advent of *'interactive processing'*:



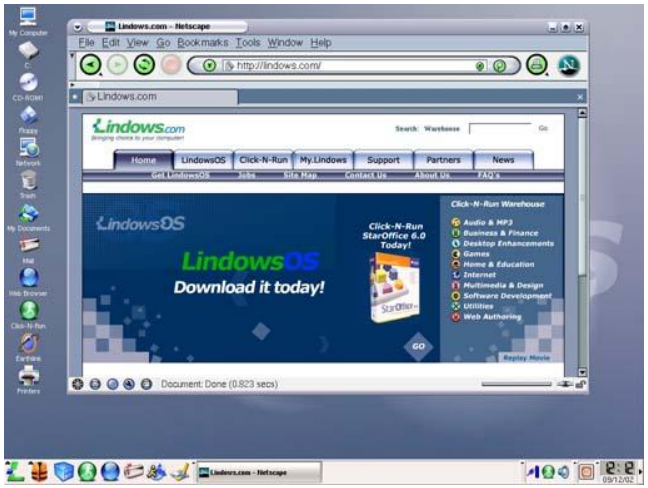
  - Provide services in a timely manner
    - *'real-time processing'*
  - Multitasking (single-user) & time-sharing (multi-user)

3.1: The Evolution of Operating Systems (3)

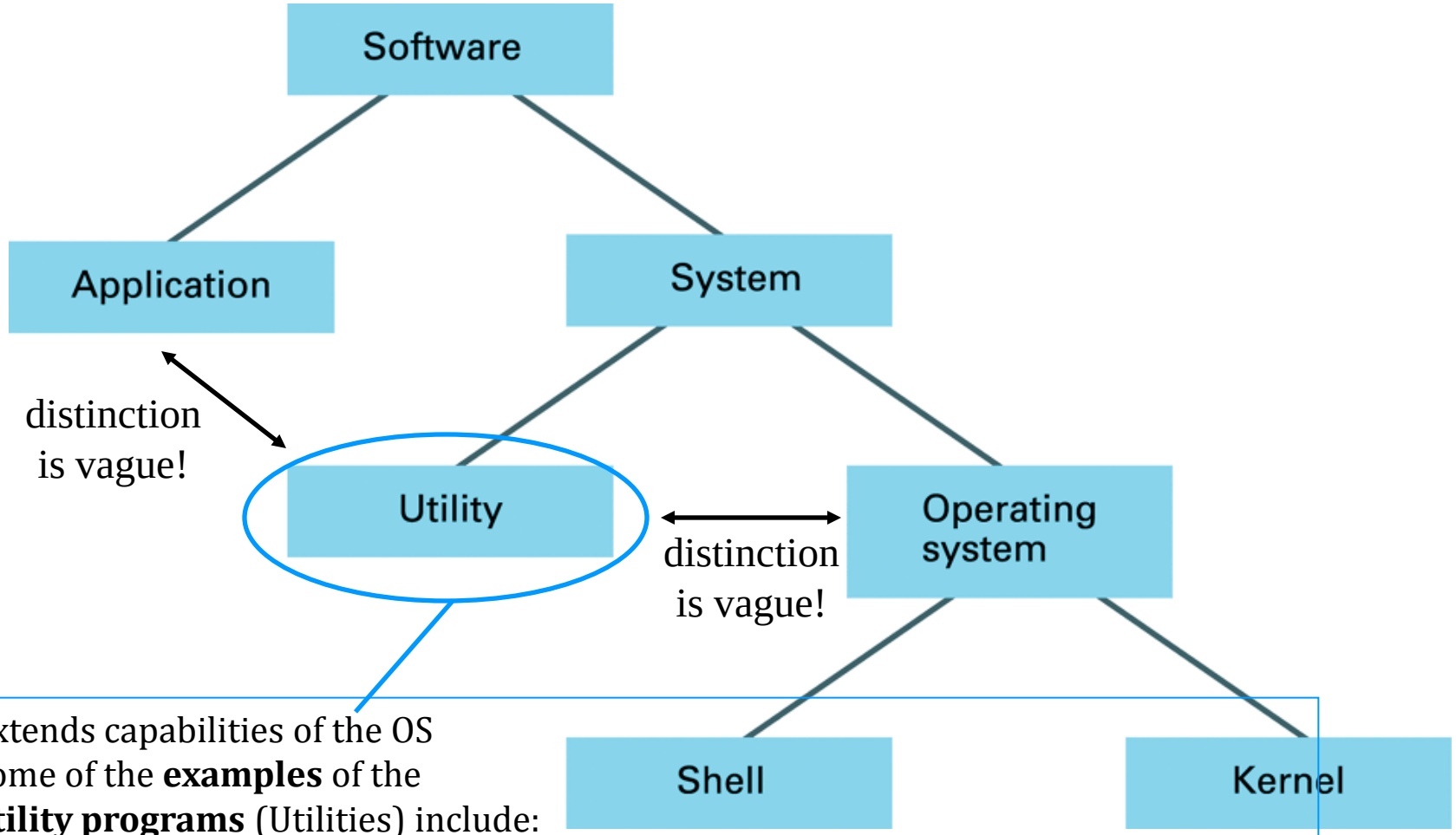
- 1980-now:
  - Operating systems for multi-processor architectures
    - includes: *load balancing*
  - Focus on user-friendliness:
    - especially: Graphical User Interface (GUI)



=>



3.2: Software classification



Extends capabilities of the OS
Some of the **examples** of the **utility programs** (Utilities) include:
Disk defragmenters, System Profilers, Network Managers, Application Launchers, Antivirus **software**, Backup **software**, Disk repair, Disk Cleaners, Registry Cleaners, Disk Space analyzer, file manager, File Compression, Data Security and many more

3.2: Components of an Operating System

- Interface between the OS and users:
  - shell (command-line, or GUI incl. window manager)
- Internal part of OS:
  - kernel :
    - file manager (coordinates the use of mass storage)
      - {Directory, folder, path and file descriptor}
    - memory manager (coordinates the use of main memory, especially in single-user or multi-user environments)
      - {Virtual Memory: Pages created and stored on Mass storage}
    - device drivers (for communication with external device controllers)
    - scheduler (coordinates the execution of multiple activities)
    - dispatcher (controls the allocation of time slices to activities)

3.3: The Concept of a Process (1)

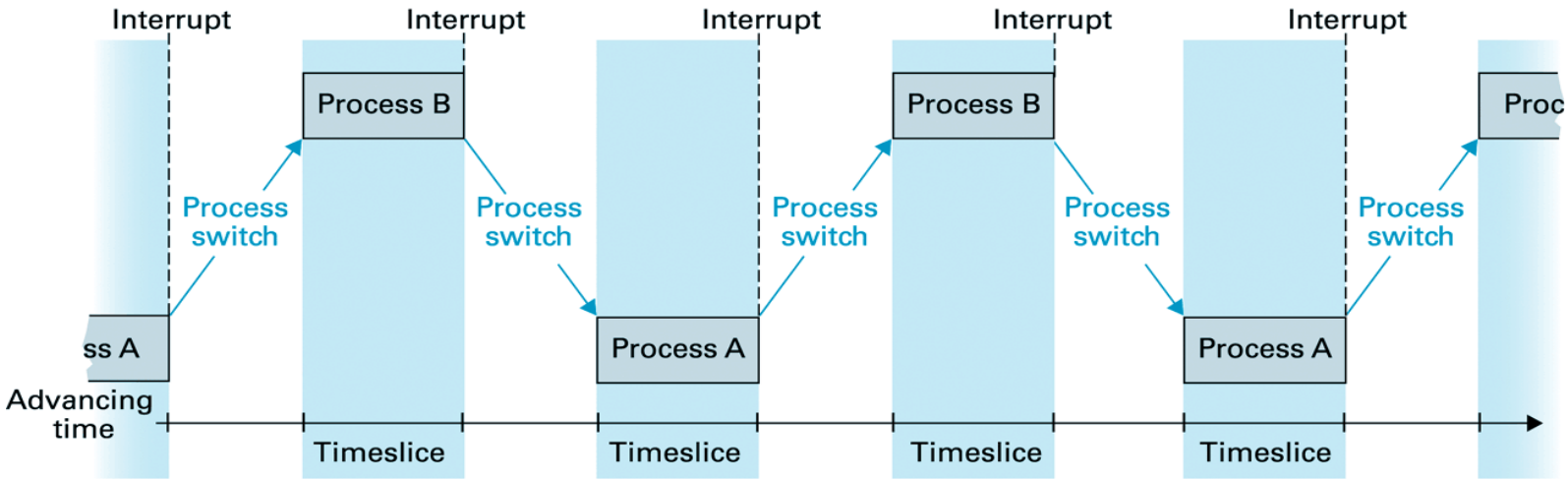
- Important is distinction between a ‘program’ and the ‘activity of executing a program’!
  - Program is a *static* set of directions
  - Activity is *dynamic*, and its properties may change over time **=>** ‘*process*’
- Process has state, including:
  - current position in program (value of the program counter)
  - values in general-purpose registers & memory cells
- So: state is snapshot of machine at certain time

3.3: The Concept of a Process (2)

- A single program may run multiple processes
  - Example: multi-user word-processing program
    - two users edit separate documents at same time
    - both use the same copy of the program in main memory, but each with its specific set of data & process states
- In a computer system many processes compete for *time slices*
- Coordination / administration of these is one of most important tasks of time-sharing OS…

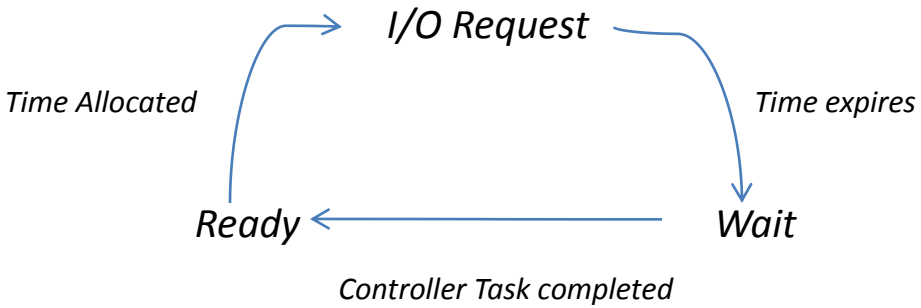
3.3: Process Administration & Time-sharing

- Process administration handled by
  - (1) scheduler
    - keeps track of all processes by maintaining a *process table*
      - *Entries like memory area and Priority (wait or ready)*
  - (2) dispatcher
    - ensures that scheduled processes are executed by dividing time into slices/ quantum , and switching CPU's attention among the processes
      - *Interrupt handler*



Process Administration

- *Coordination of Dispatcher and Scheduler : Adjusting to Priorities*
- *Example:*



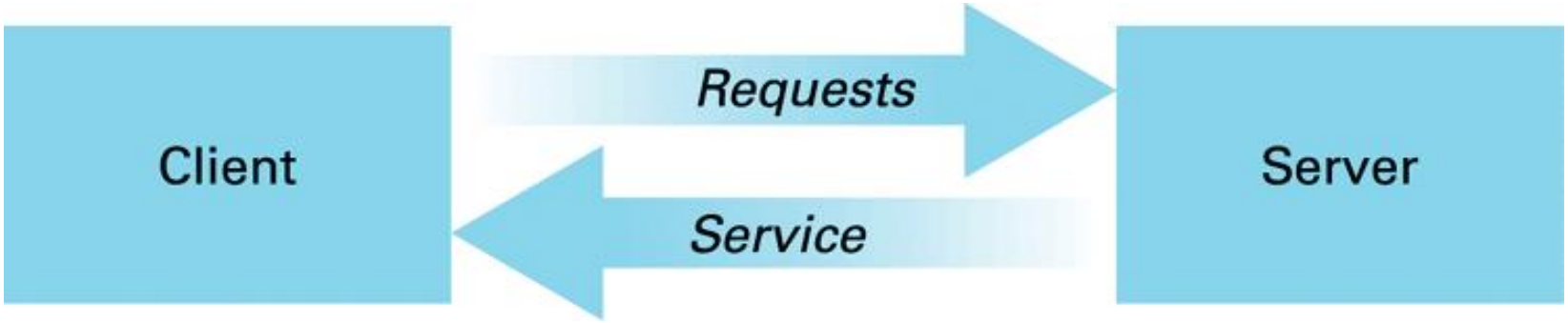
- *Restore at the point of interrupt*

- *Immediate Environment prior to interrupt-> Process States*

3.3: Interprocess Communication

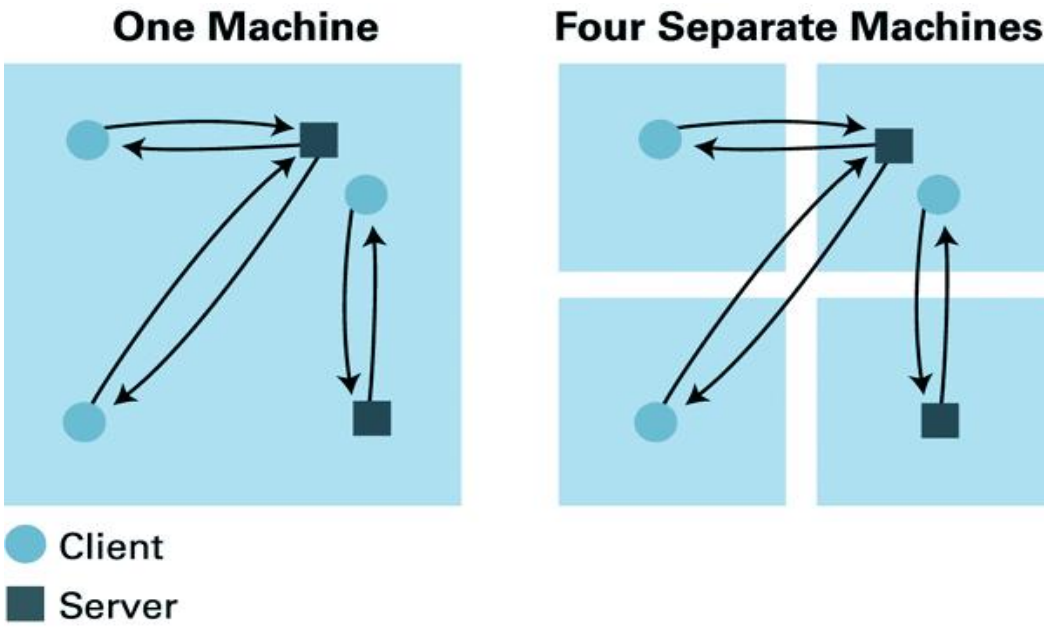
- Various units within an OS also execute as processes
- To coordinate their activities, these processes must communicate with each other
  - Interprocess communication

- One form of Interprocess communication:
  - *client/server model* (also used in computer networks)
  - example: file-server providing access to files on request

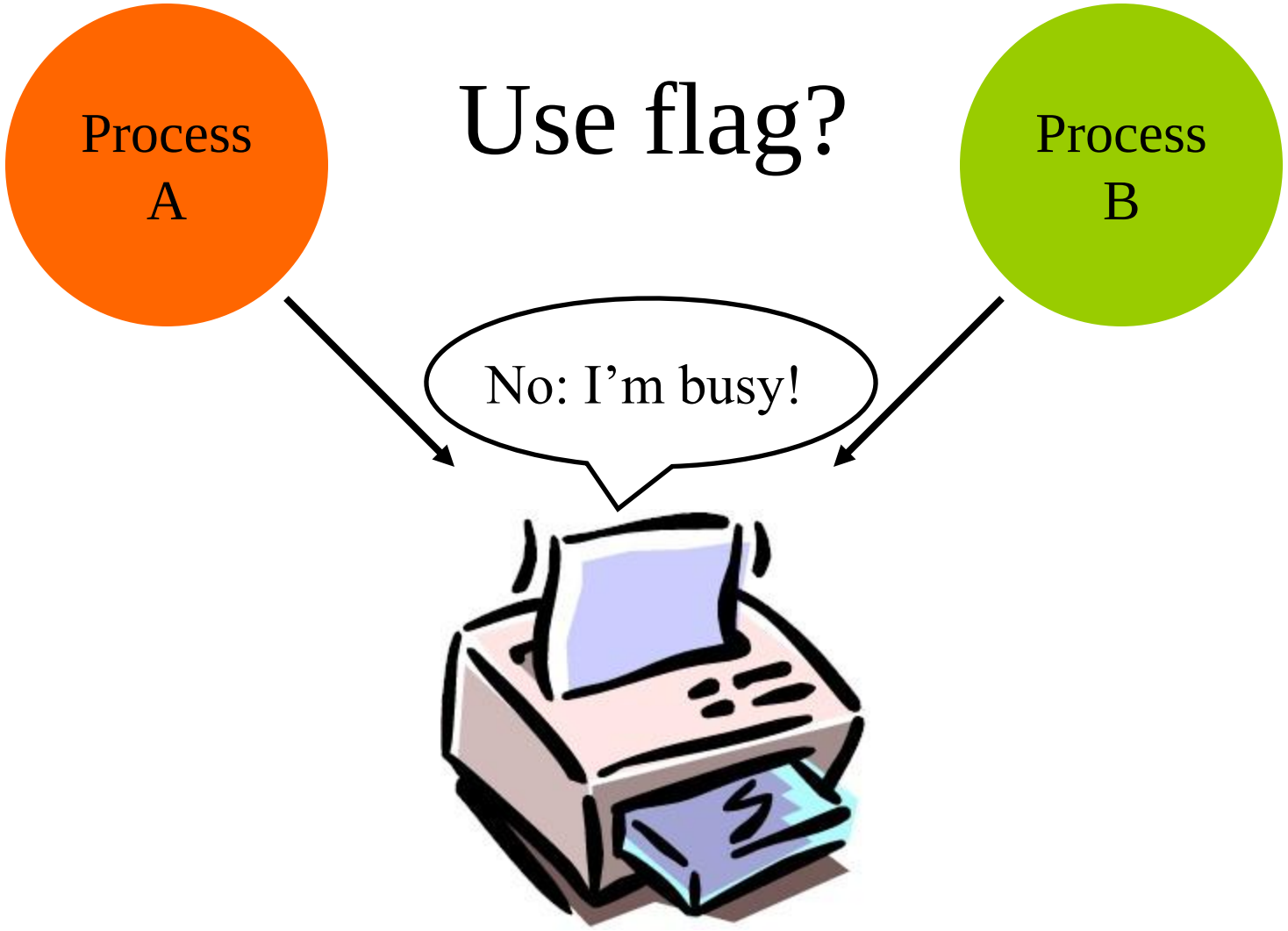


3.3: The Client/Server Model

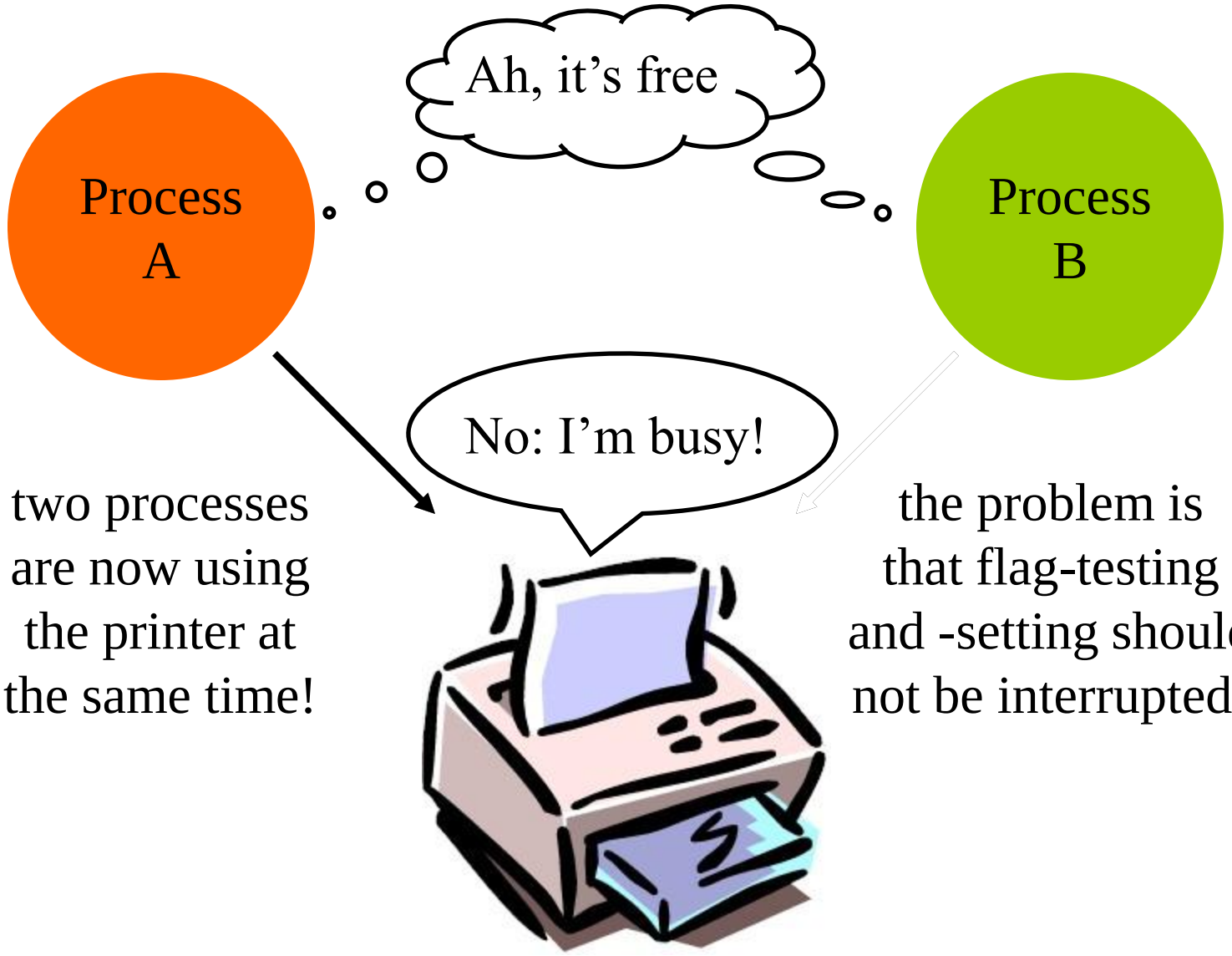
- Client and Servers are softwares within a machine or among different machines
- The role of the server is the same whether the client resides on the same machine or on a distant machine!
- Difference only in the communication software, not in the clients and servers
- CORBA(Common Object Request Broker Architecture): standard for network wide communication between softwares units known as objects (such as clients and servers)



3.4: Handling Competition Among Processes



3.4: Problems…!



two processes are now using the printer at the same time!

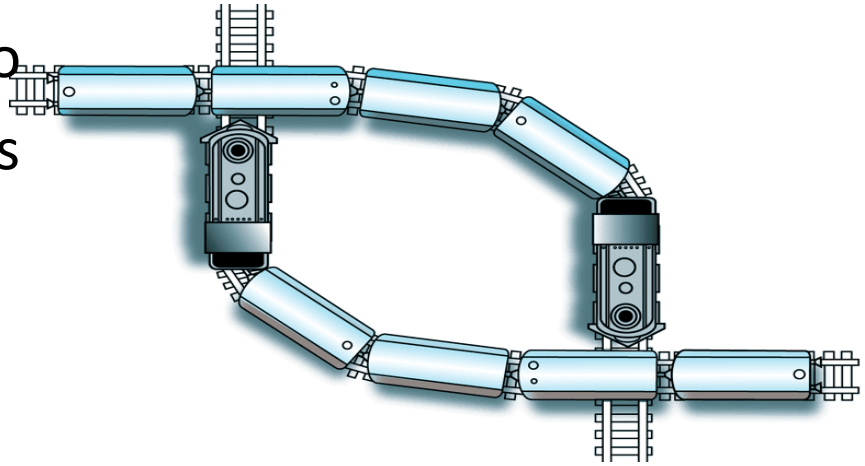
the problem is that flag-testing and -setting should not be interrupted!

3.4: Solutions...

- One possibility is to use *interrupt enable* and *interrupt disable* instructions
  - disadvantage: process may remove the possibility of being interrupted altogether
- Other approach is to use single *test-and-set* instruction
  - always completed before an interrupt can be handled
  - flag implemented this way is a.k.a.: *semaphore*
  - Critical Region
  - A sequence of instructions executed by only one process at a time
  - Mutual Exclusion: A requirement

3.4: Another problem: Deadlock

- Two or more processes are blocked because each is waiting for access to resources allocated to another
  - task 1: printer yes, disk drive no
  - task 2: printer no, disk drive yes



Three must conditions for deadlock:

1) **There is competition for non shareable resources.**
2) **The resources are requested on partial basis; process returning back for more resources.**
3) **Once a resource has been allocated, it cannot be forcibly retrieved.**

Solutions:

- **Deadlock detection and correction**
  - by forcibly retrieving some allocated resources
  - Example: Process Kill
- **Deadlock avoidance**
  - spooling (make the resource appear as if it can be shared by multiple processes at same time)
  - Example: Printer Spooling
- **Other Problems**
  - File access: read and write

Chapter 3 - Operating Systems: Conclusions

- Operating System - ‘glue’ between hardware and applications
- Manages and controls multiple applications running at same time
- May also service multiple users at same time
- Multi-tasking / time-sharing based on processes
- Difficulties arise due to competition among multiple processes and deadlocks