Chapter 7

Operating Systems

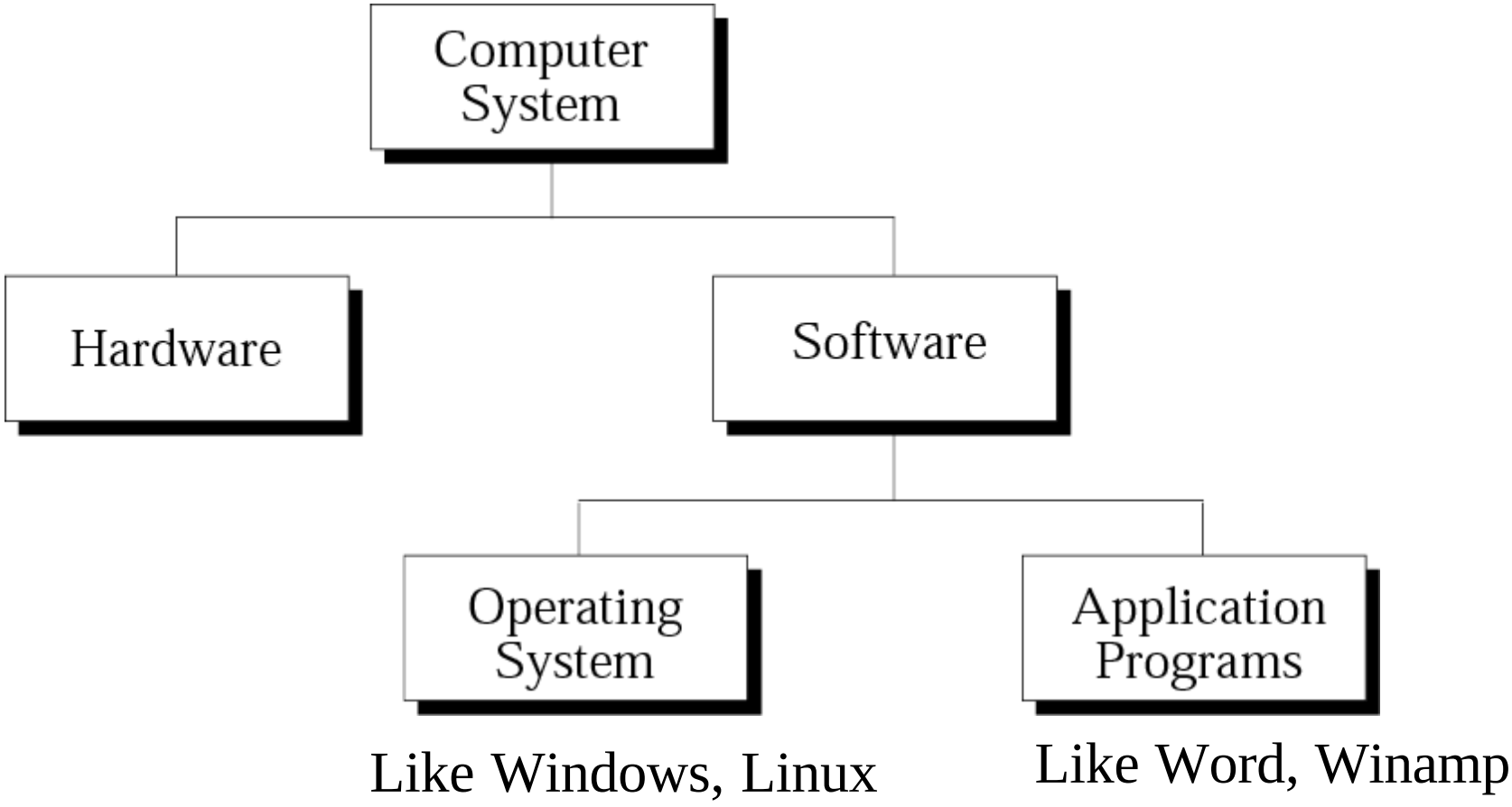
OBJECTIVES

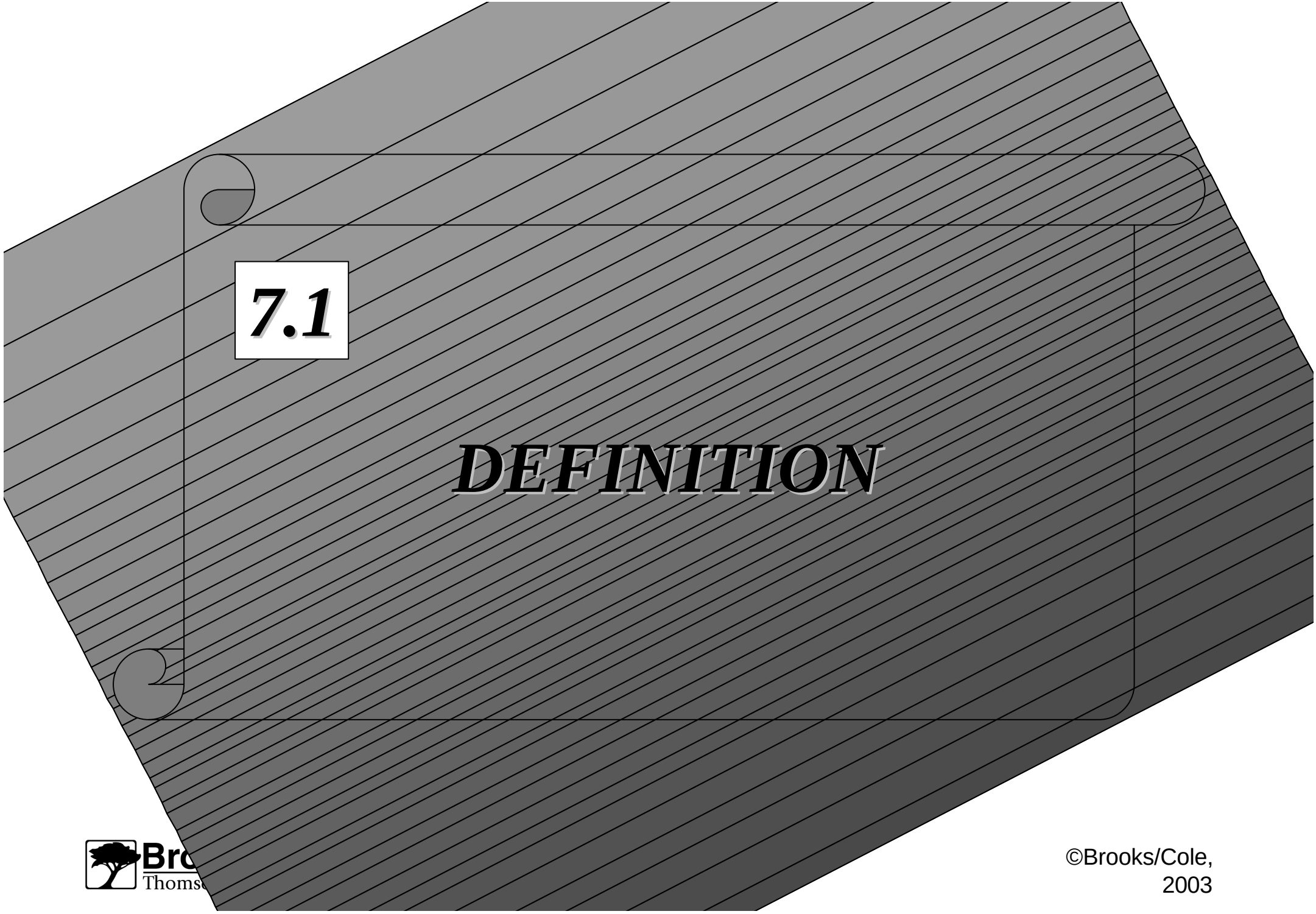
After reading this chapter, the reader should be able to:

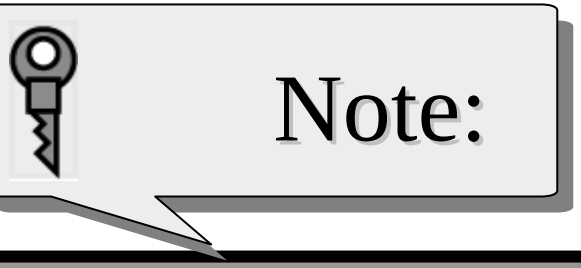
- Define the purpose and functions of an operating system.
- Understand the components of an operating system.
- Understand the concept of virtual memory.
- Understand the concept of deadlock and starvation.
- List some of the characteristics of popular operating systems such as Windows 2000, UNIX, and Linux.

Figure 7-1

Computer System







An operating system is an interface between the hardware of a computer and the user (program or human) that facilitates the execution of the other programs and the access to hardware and software resources.

Major Design Goals of an Operating System

- Efficiency use of hardware
- Easy to use resources

7.2

EVOLUTION:
Batch systems
Time-sharing systems
Personal systems
Parallel systems
Distributed systems

Batch System

- Designed in 1950 for mainframe computers
- Punched cards, line printer, tape
- Each program is called a job.
- Programmer can’t interact with system
- OS simply ensures that all of the resources were transferred from one job to the next.

Time-Sharing System

- Multiprogramming was introduced.
- Process: a program in memory waiting for execution.
- Assign a resource to a process when resource is available.
  - One process=> I/O, another =>CPU
- Time Sharing=> Each process are located a portion of time to use the resource.
  - Computer is faster than human
- Scheduling: Allocating resources to processes.

Personal System

- Introduced when Personal Computer (PC) were introduced.
- Like DOS (Disk Operating System).

Parallel System

- Multiple CPU
- Each CPU can serve a program or part of a program
- Tasks can be accomplished in parallel instead of serially.

Distributed System

- A job was previously done by one computer.
- Networking creating a new dimension of OS.
- A program can be run on different computers via network.
- Should handle controlling security

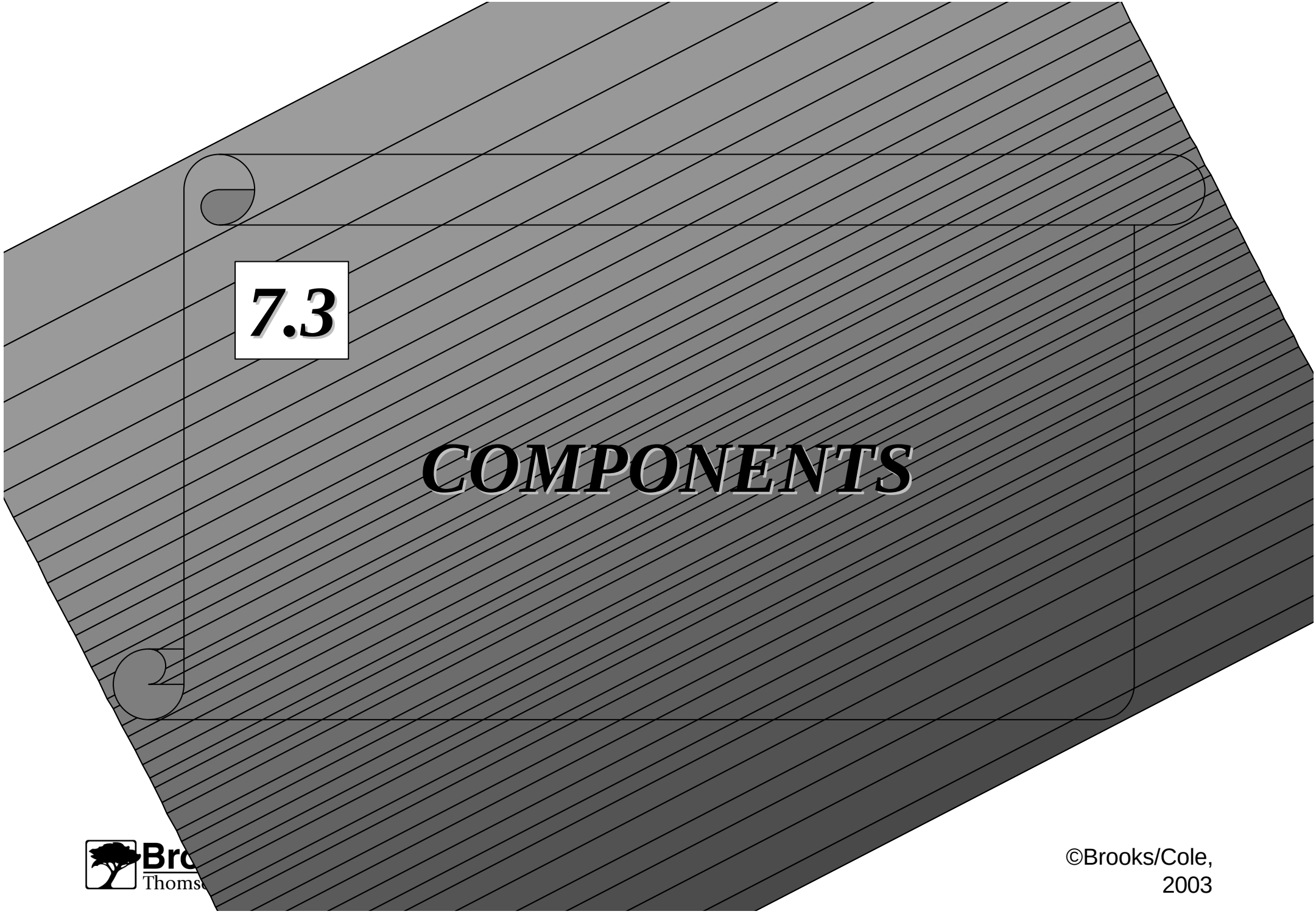
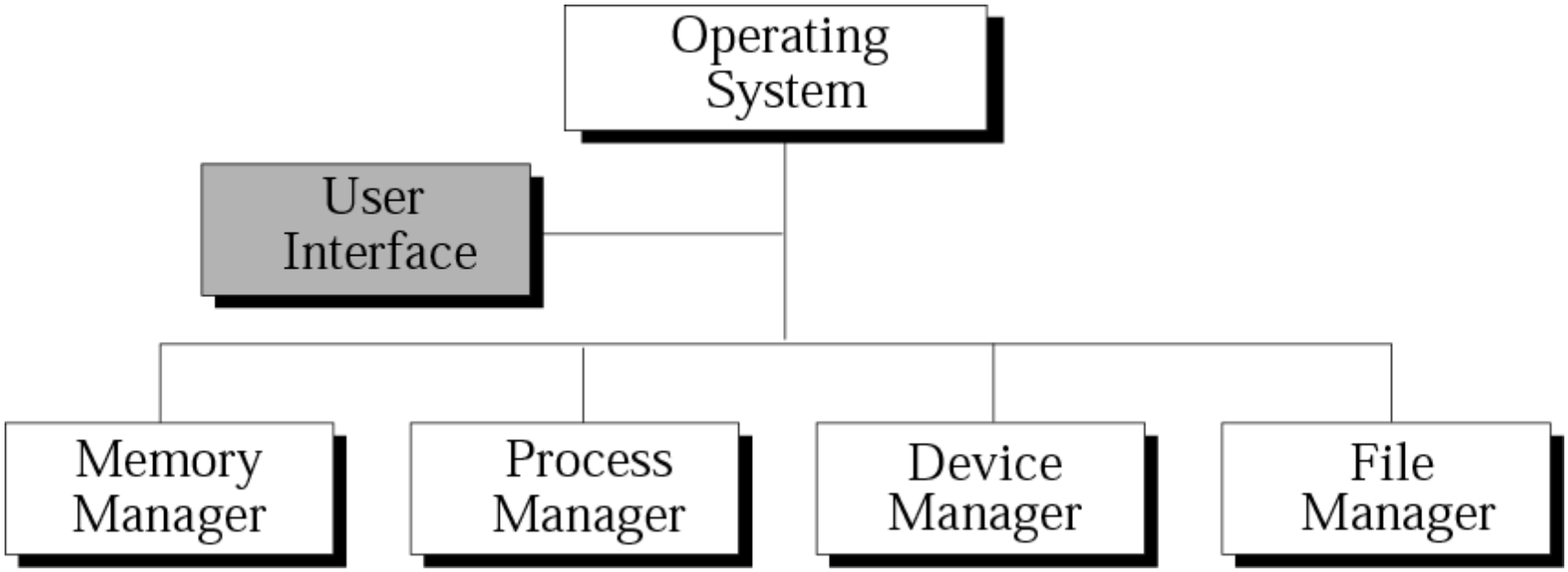


Figure 7-2

Components of an operating system

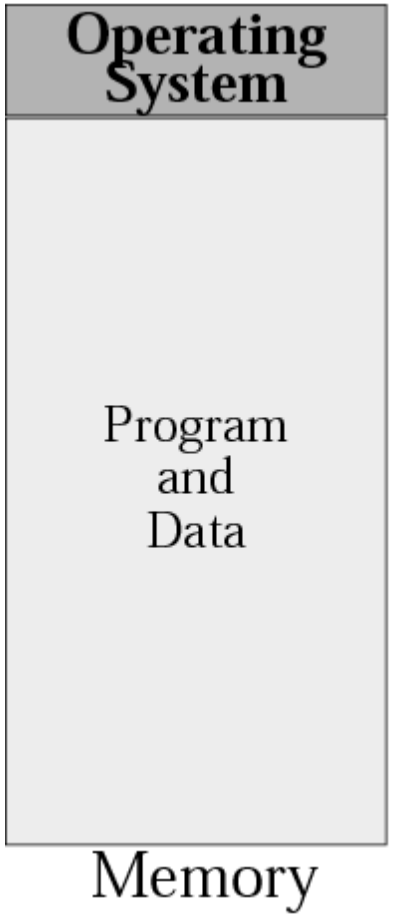


Memory Manager:

- Prevent “running out of the memory” problem.
- mono-programming
- multi-programming

Figure 7-3

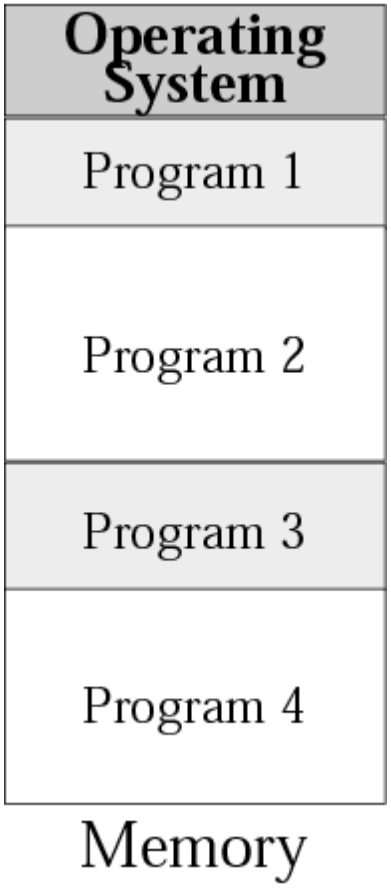
Monoprogramming



- The whole program is in memory for execution.
- When finished, replace by another program.
- Program must fit memory.
- One program is running, no other program can be executed.
  - Not efficient especially for I/O waiting.

Figure 7-4

Multiprogramming



- More programs in memory.
- Executed concurrently.
- Non-swapping:
  - Program remains in memory for the duration of execution.
- Swapping:
  - program can be swapped between memory and disk.

Figure 7-5

Categories of multiprogramming

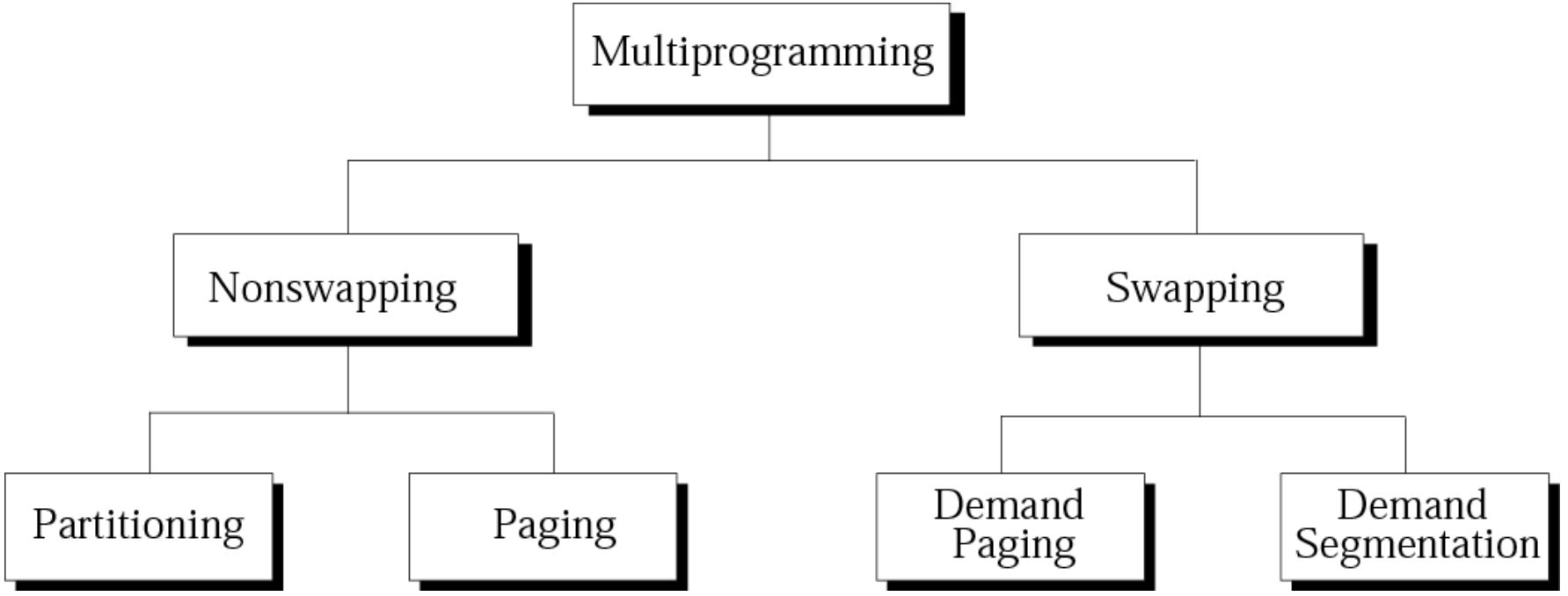
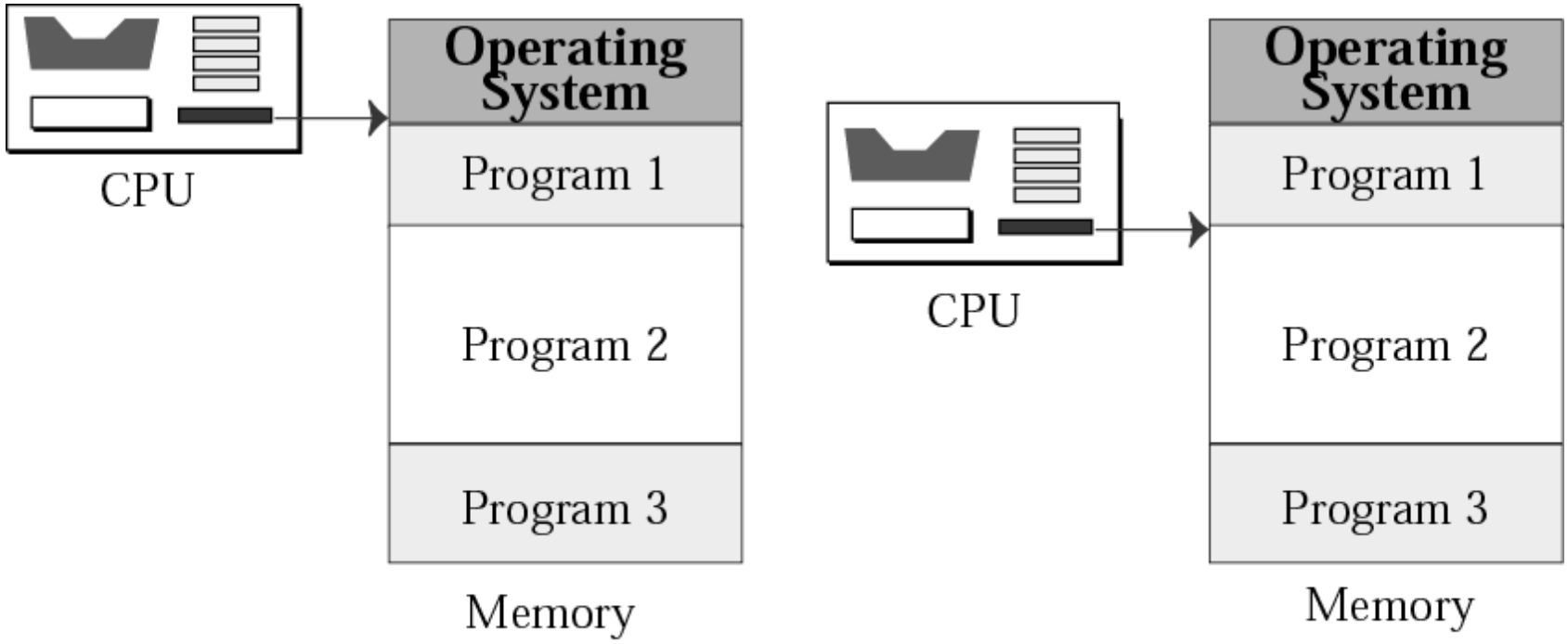


Figure 7-6

Partitioning



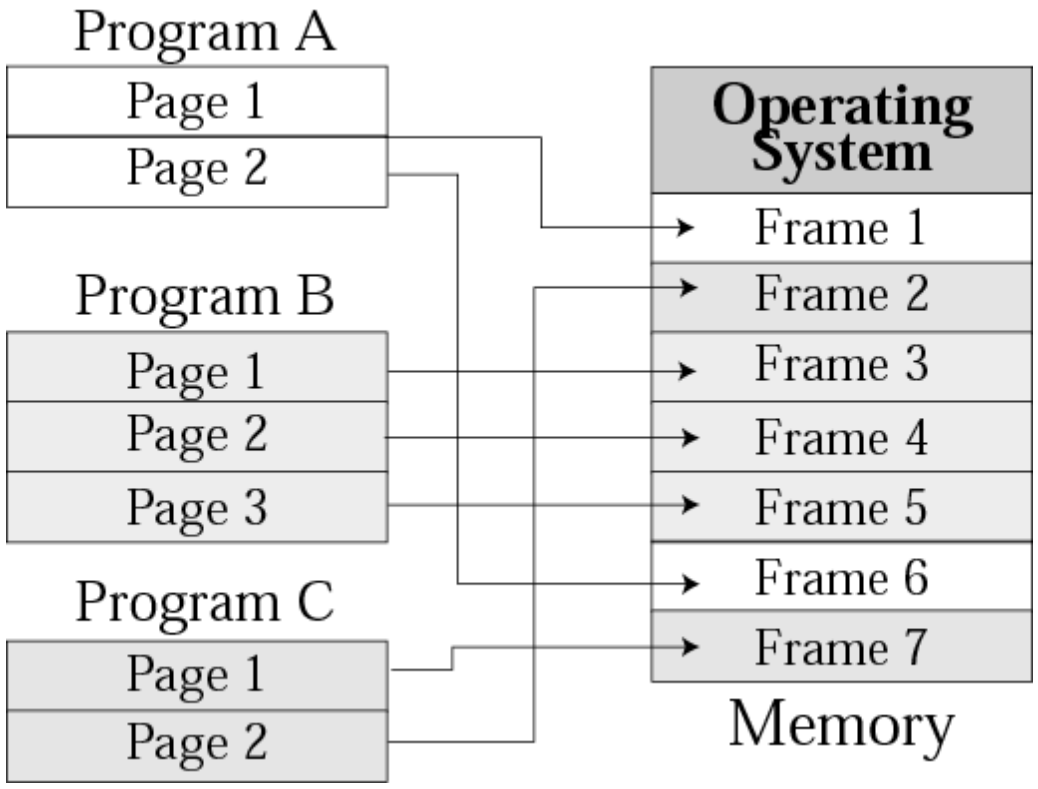
a. CPU starts executing program 1. b. CPU starts executing program 2.

Partitioning

- Memory divide into variable length partitions.
- Each partition holds one program.
- A program occupy contiguous locations.
- CPU switches
  - encounter an I/O operation
  - the execution time has expired.
- Problem:
  - Memory holes appear after programs are replaced.
  - Remove the holes create extra overhead.

Figure 7-7

Paging



Paging

- Improve the efficiency of the partitioning.
- Memory is divided into frames.
- Program is divided into pages.
- Pages can be loaded into non-contiguous frames.
- Problem:
- The program needs to wait when there are not enough non-occupied frames.

Demanded Paging

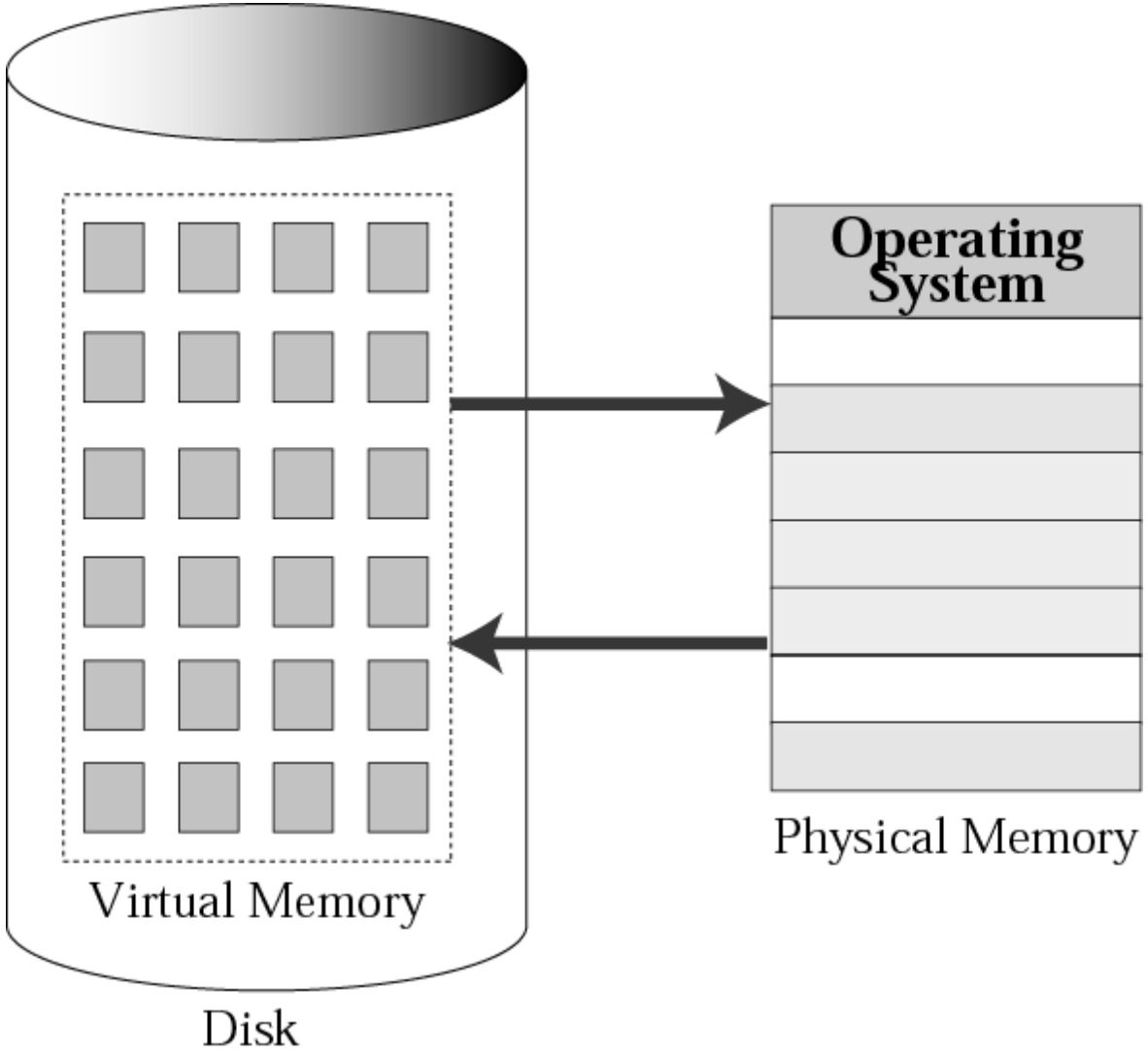
- Does require that entire program be in memory for execution.
- Pages can be loaded, executed one by one.

Demanded Segmentation

- In paging, a program is dividend into pages.
- A program can be divided into modules.
- Demand segmentation: The program is divided into segments .
- Demanding Paging and Segmentation:
  - Divide a segment into pages.

Figure 7-8

Virtual memory
Used in Swapping technique

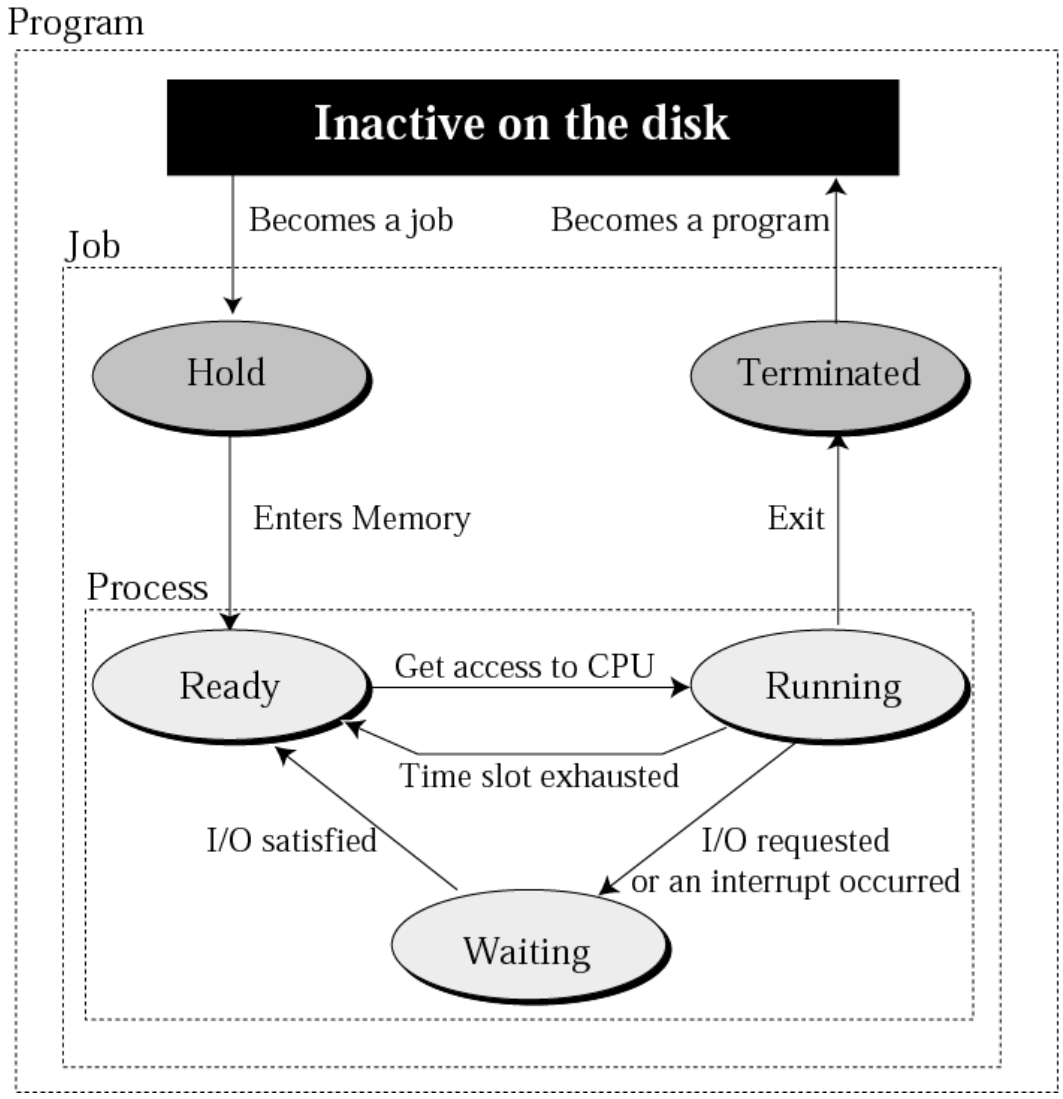


Process Manager

- Program: A set of instructions written by a programmer.
- Job: A program becomes a job at the time it is selected for execution until it finishes.
  - It may be residing on the disk waiting for loading.
  - It may be residing on the memory waiting for execution.
- Process: A program in execution.
  - A process is a job residing in the memory.

Figure 7-9

State diagram with the boundaries between a program, a job, and a process



Swapping and virtual memory is not considered here.

Schedulers

- To move a job (process) from one state to another.
  - Job schedulers
  - Process schedulers

Figure 7-10

Job scheduler

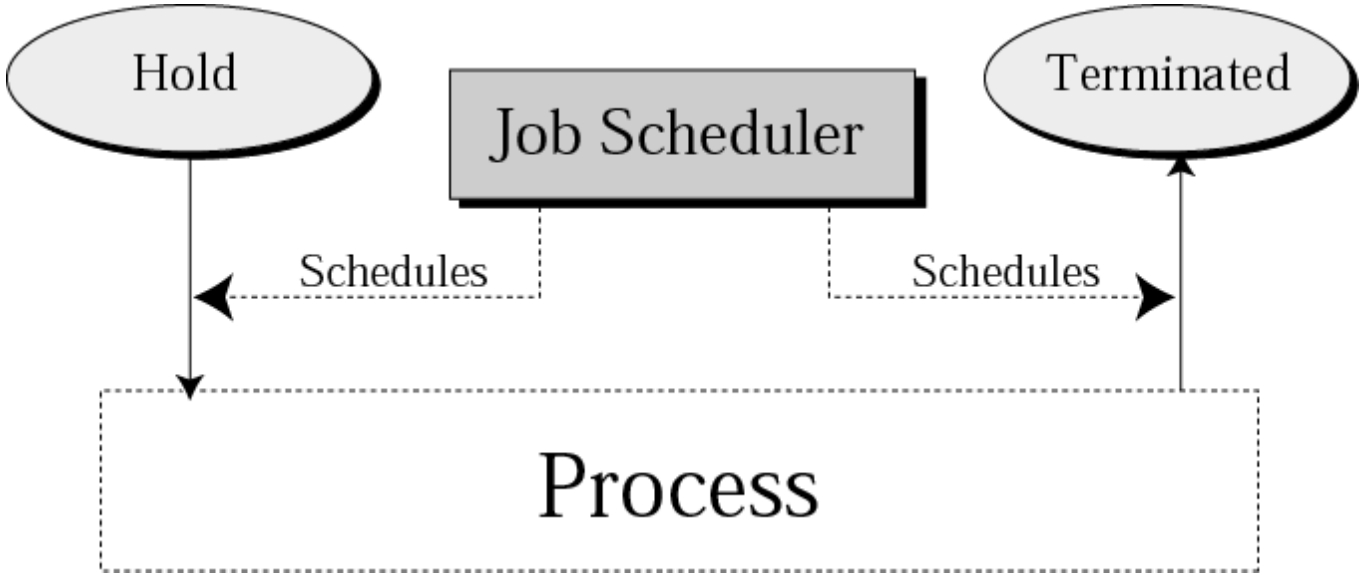


Figure 7-11

Process scheduler

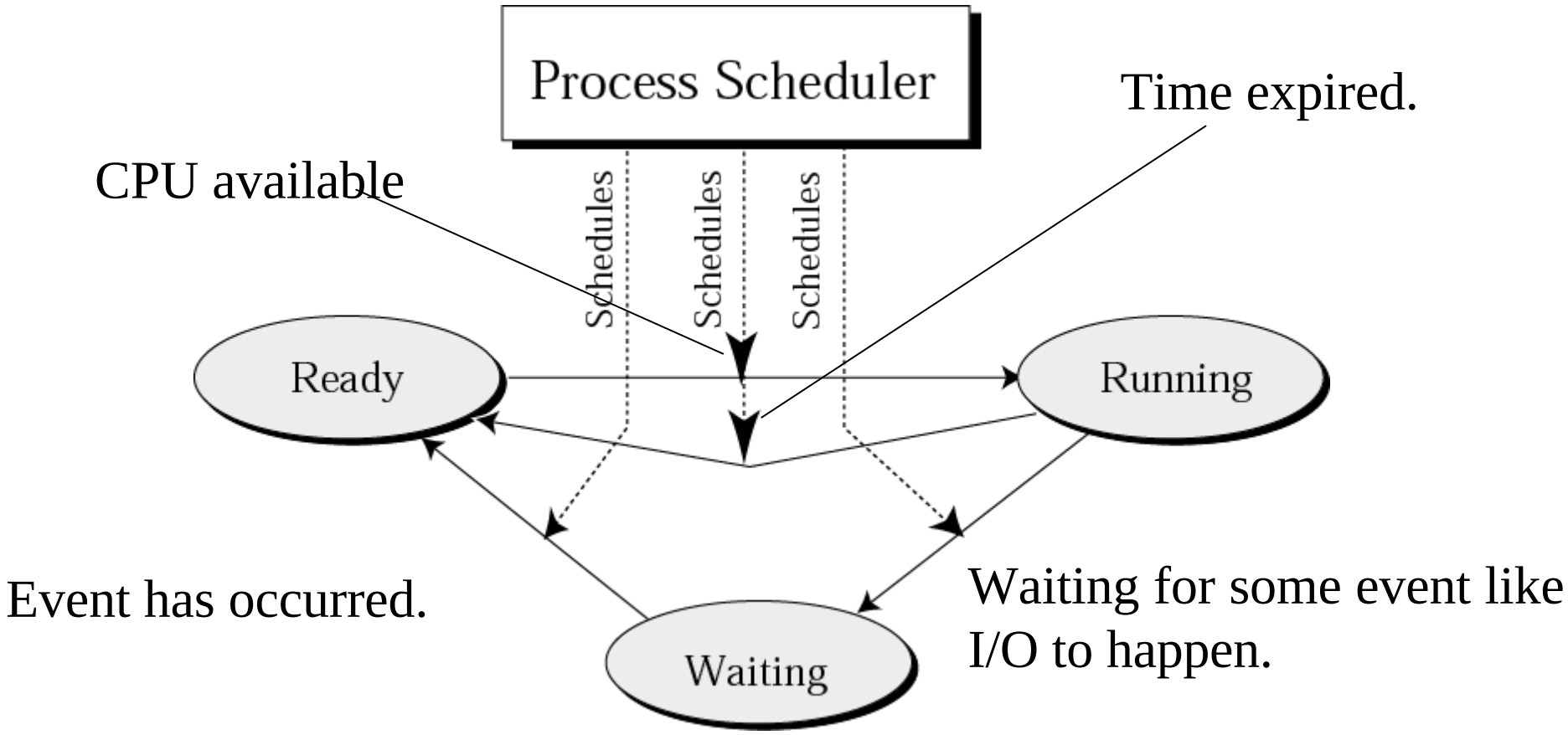
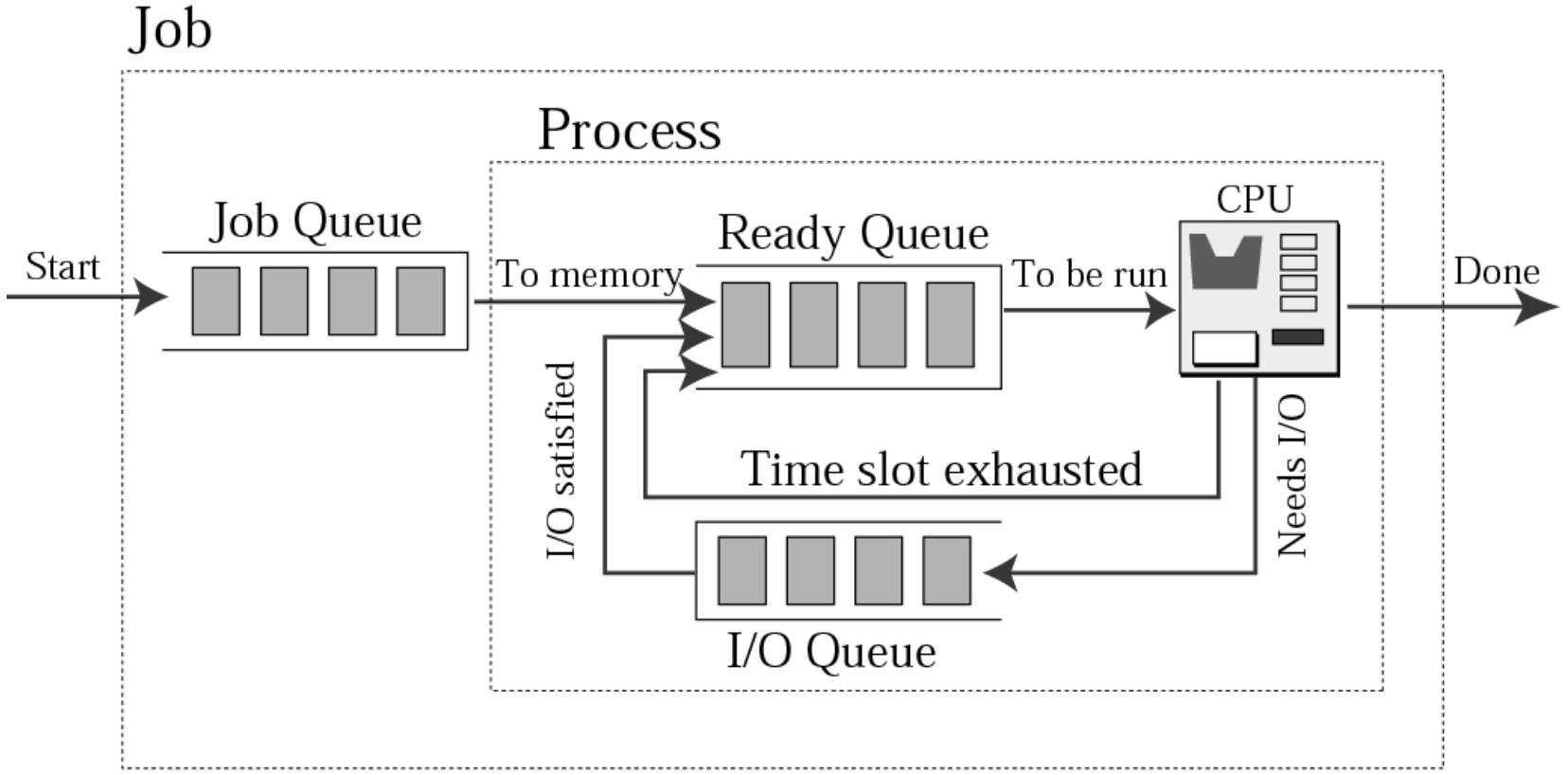


Figure 7-12

Queues for process management

Queue: waiting list (can use different policies).
Job (process) control blocks store information about the job (process).
These blocks are stored in queues.

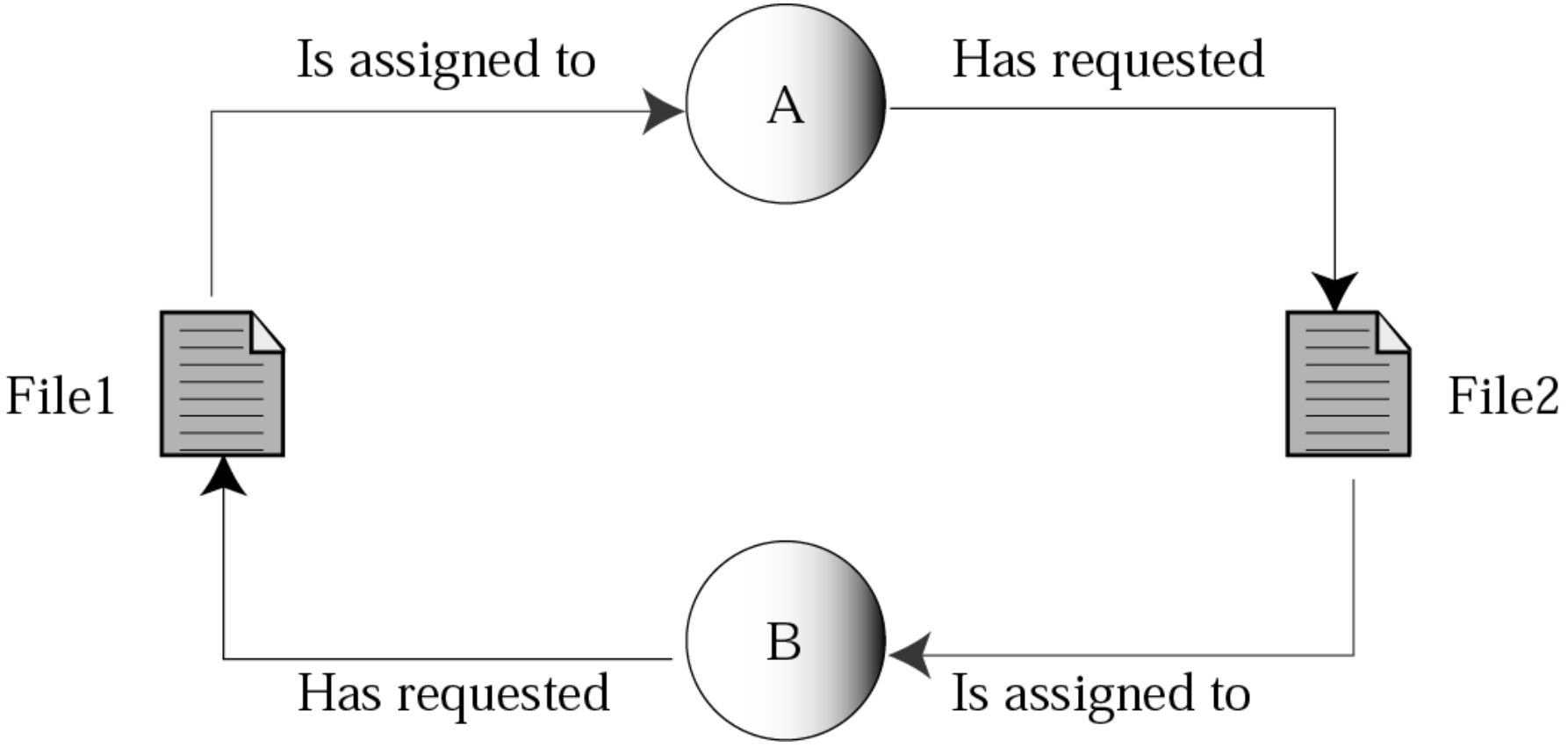


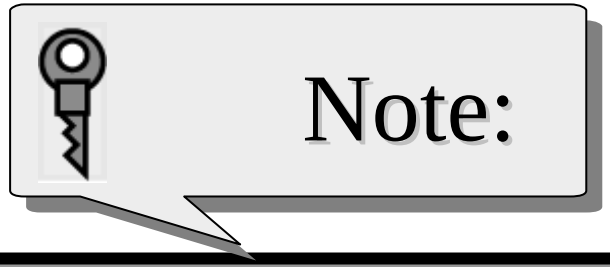
Process Synchronization

- Synchronize different processes with different resources.
- Two problem:
  - Deadlock
  - Starvation

Figure 7-13

An Example of Deadlock





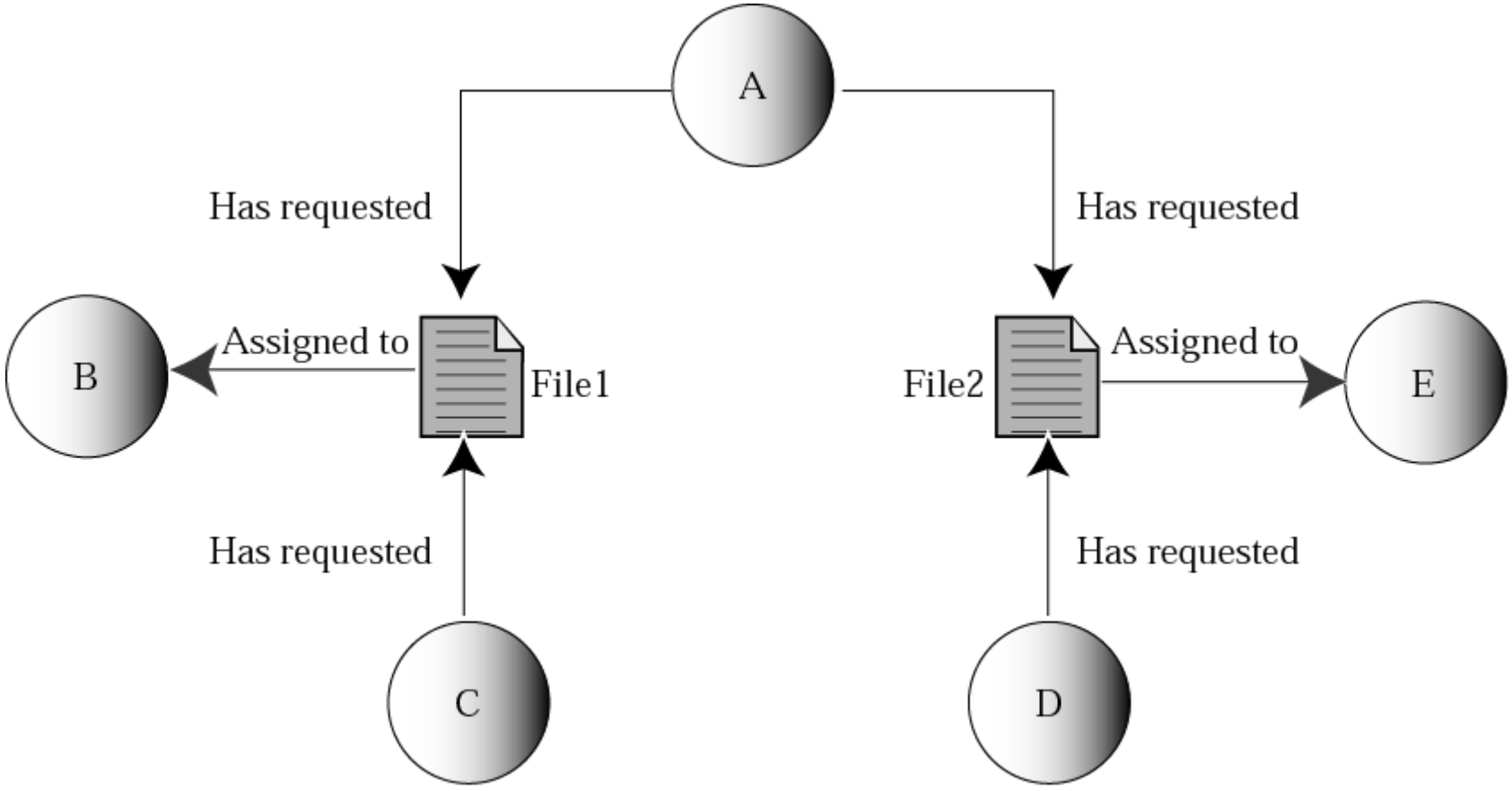
Deadlock occurs when the operating system allows a process to start running without first checking to see if the required resources are ready and allows the process to hold it as long as it wants.

Deadlock

- Necessary conditions:
  - Mutual exclusion (Only one process can hold a resource)
  - Hold a resource even through it can't use it until other resource is available.
  - No preemption (OS can't reallocate the resource)
  - Circular waiting
- Prevent deadlock:
  - Allow a process to run when resources are available (starvation problem).
  - Limit the time a process can hold a resource.

Figure 7-15.a

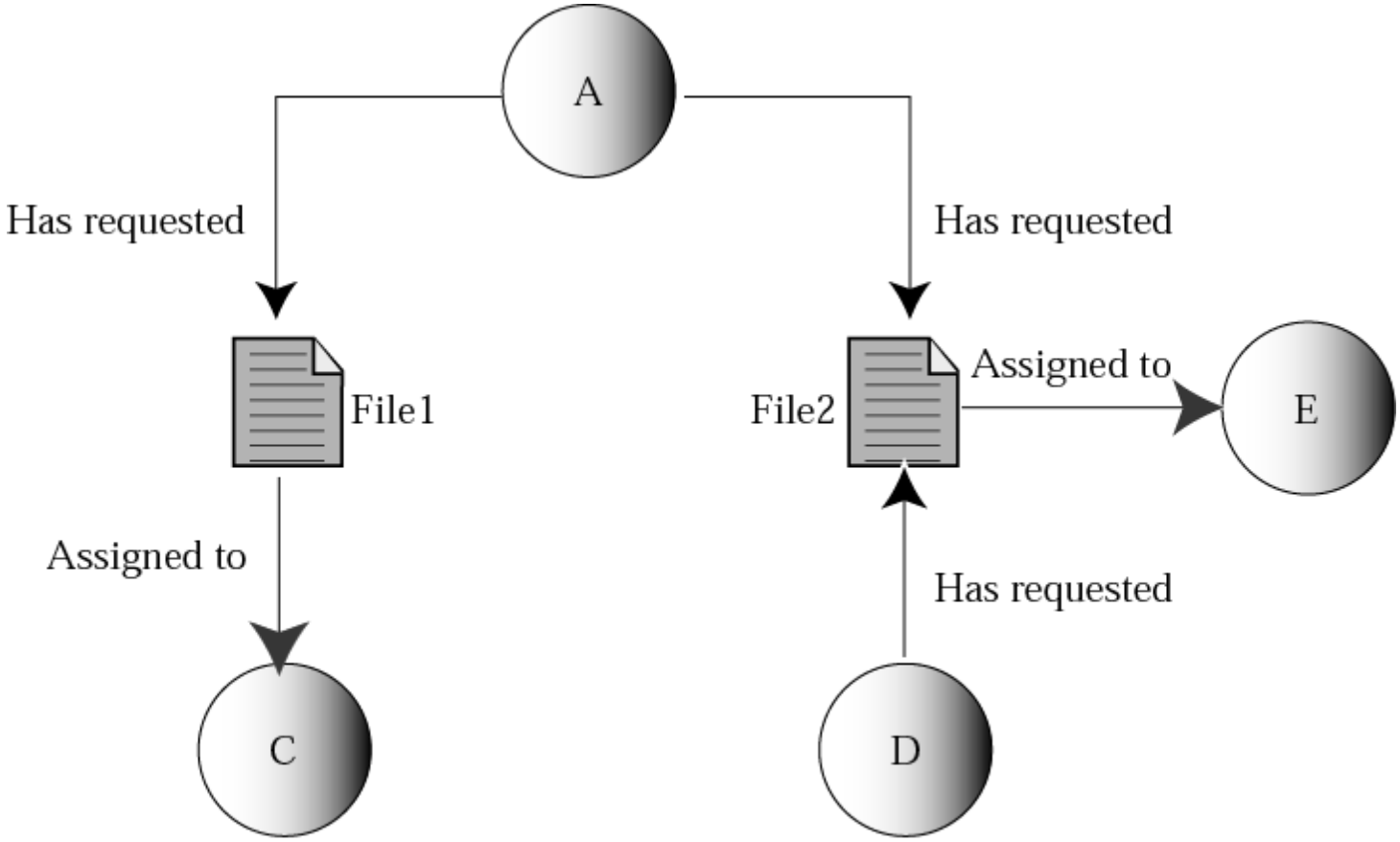
Starvation



a. Process A needs both File1 and File2.

Figure 7-15.b

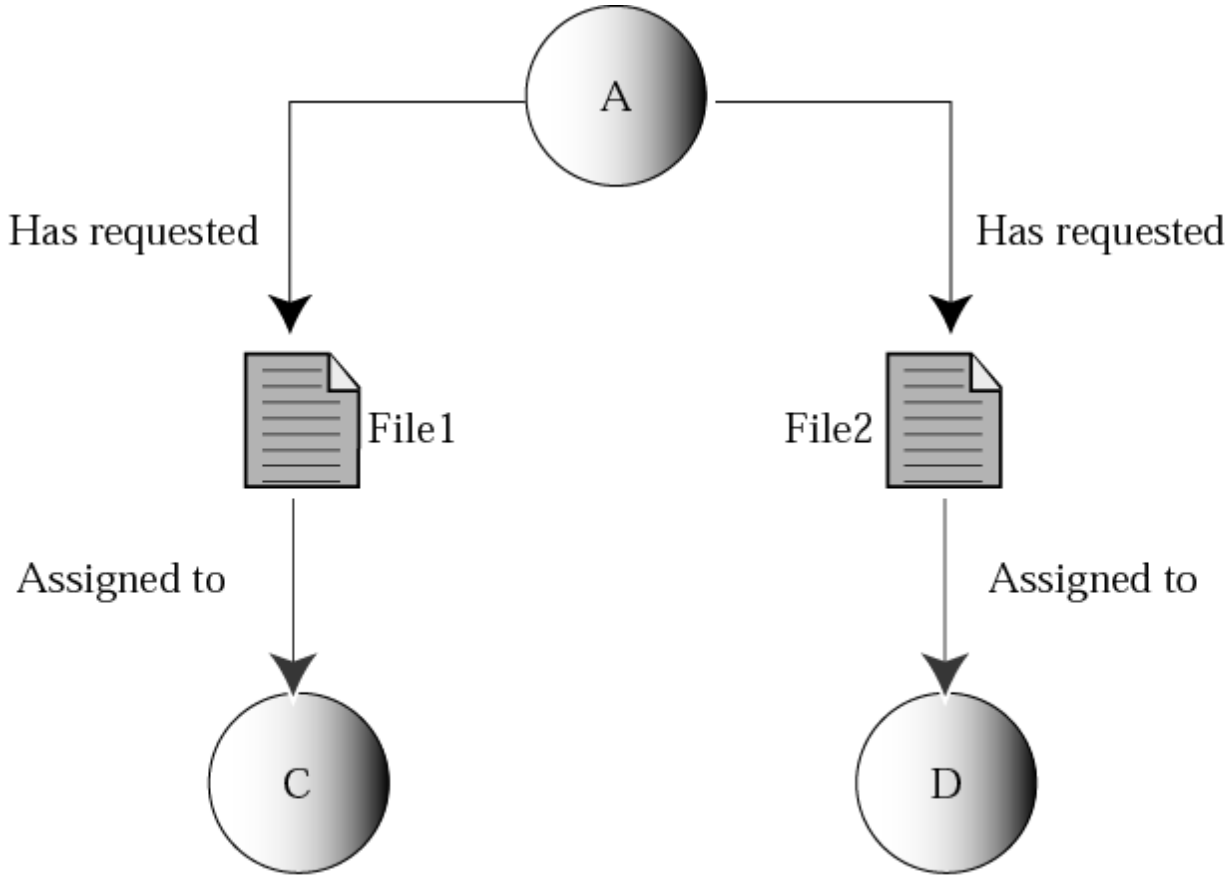
Starvation



b. Process A still needs both File1 and File2.

Figure 7-15.c

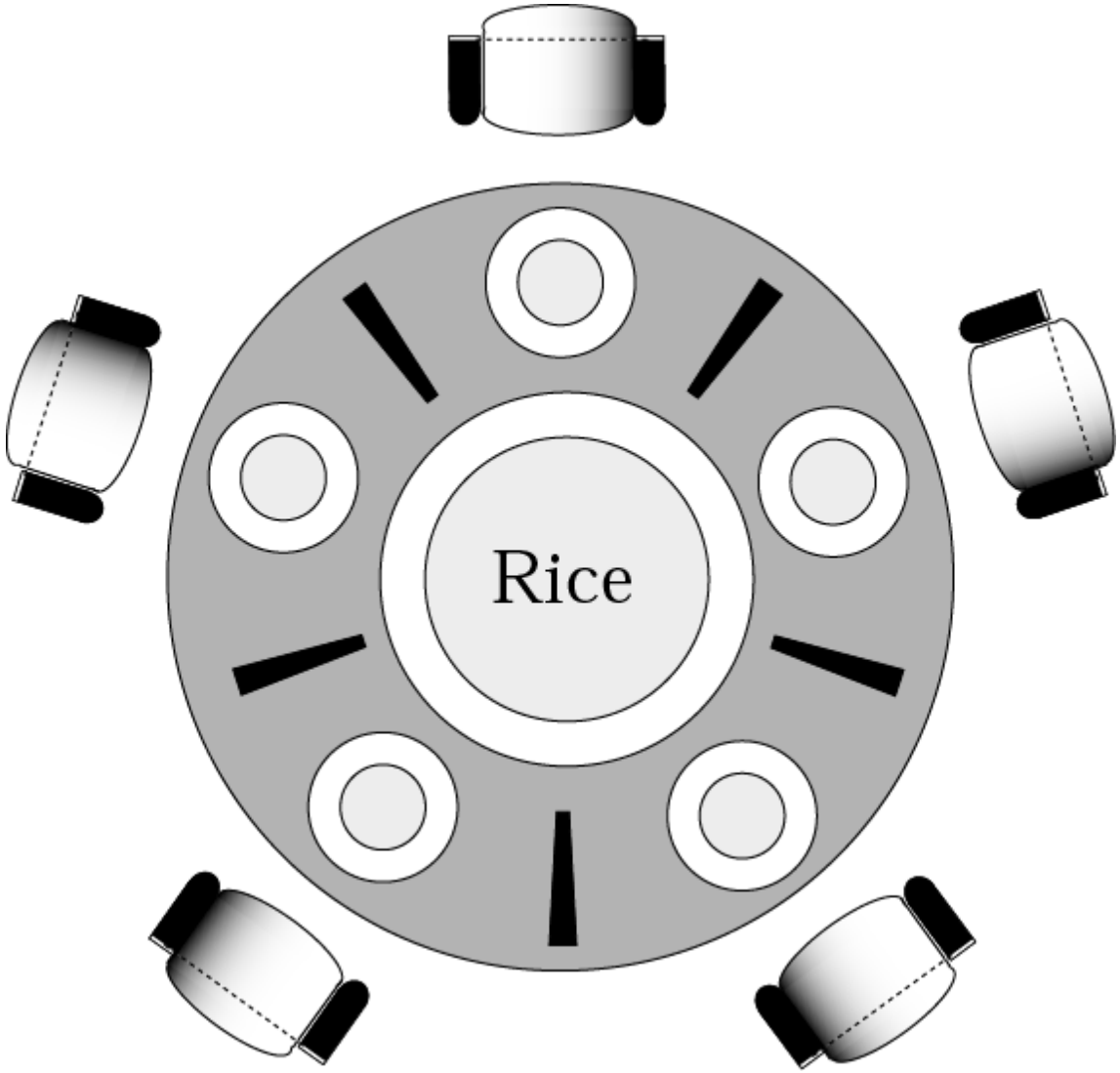
Starvation



c. Process A still needs both File1 and File2 (starving).

Figure 7-16

Dining philosophers



Device Manager

- Input/Output manager
- Limit by the number and the speeds of I/O devices. (slower than CPU and memory)
- Device manager
  - Monitor the devices constantly
  - Scheduling for accessing the I/O devices.

File Manager

- File manager control access to the files.
- Access right
  - like read only, R/W-able , execute, can’t access
- Supervise the creation, deletion, and modification of the files.
- Control the naming of the files.
- Storage of the files.
- Backup.

User Interface

- A program that accepts the requests from users or processes and interpret them for the OS.
- UNIX– shell
- GUI (graphical user interface)– window

7.4

POPULAR OPERATING SYSTEMS:
Unix; Linux; Windows

Windows 2000

- Menu-driven operating system
- GUI
- virtual memory
- multiprogramming

UNIX

- Portable among different platforms
  - written mostly in C program
- A powerful set of utilities that can be combined by scripts.
- OS includes drivers itself.
- multi-programming
- virtual memory

LINUX

- Developed by Linus, Torvalds.
- Also called UNIX clone.
  - Similar to UNIX.
- Can be run on PC.