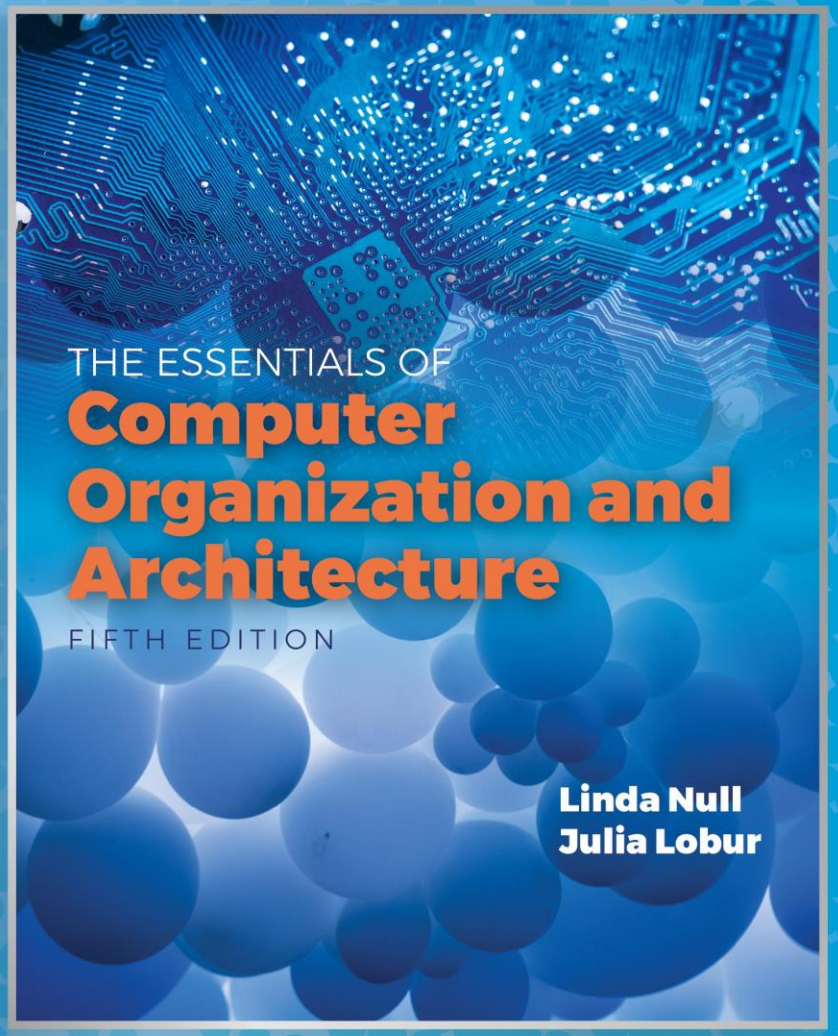
This is the first lecture of Chapter 8

Chapter 8

System Software (A)



Quick review of last lecture

- Magnetic Tape
  - Digital linear tape (DLT) and Serpentine recording
  - Digital audio tape (DAT) and helical scan recording
  - Linear Tape Open (TLO)
    - a linear digital tape format
    - hold up to 1.4TB (Generation 5)
- Redundant Array of Independent Disks (RAID)
  - RAID Levels 0 – 6
  - RAID DP
  - RAID 10
  - RAID 50

Objectives

- Become familiar with the functions provided by operating systems, Protected environments, and programming tools.
- Understand the role played by each software component in maintaining the integrity of a computer system and its data.

8.1 Introduction

- The biggest and fastest computer in the world is of no use if it cannot efficiently provide beneficial services to its users.
- Users see the computer through their application programs. These programs are ultimately executed by computer hardware.
- System software—in the form of operating systems and middleware—is the glue that holds everything together.

8.2 Operating Systems (1 of 12)

8.2.1 Operating Systems History

- The evolution of operating systems has paralleled the evolution of computer hardware.
  - As hardware became more powerful, operating systems allowed people to more easily manage the power of the machine.
- In the days when main memory was measured in kilobytes, and tape drives were the only form of magnetic storage, operating systems were simple *resident monitor* programs.
  - The resident monitor could only load, execute, and terminate programs.

8.2 Operating Systems (2 of 12)

- In the 1960s, hardware has become powerful enough to accommodate *multiprogramming*, the concurrent execution of more than one task.
- Multiprogramming is achieved by allocating each process a given portion of CPU time (a *timeslice*).
- Interactive multiprogramming systems were called *timesharing* systems.
  - When a process is taken from the CPU and replaced by another, we say that a *context switch* has occurred.

8.2 Operating Systems (3 of 12)

- Today, multiprocessor systems have become commonplace.
  - They present an array of challenges to the operating system designer, including the manner in which the processors will be synchronized, and how to keep their activities from interfering with each other.
- *Tightly coupled* multiprocessor systems share a common memory and the same set of I/O devices.
  - *Symmetric multiprocessor systems* are tightly coupled and load balanced.

8.2 Operating Systems (4 of 12)

- Loosely coupled multiprocessor systems have physically separate memory.
  - These are often called *distributed systems*.
  - Another type of distributed system is a networked system, which consists of a collection of interconnected, collaborating workstations.
- Real-time operating systems control computers that respond to their environment.
  - *Hard real-time* systems have tight timing constraints, *soft real-time* systems do not.

8.2 Operating Systems (5 of 12)

- Personal computer operating systems are designed for ease of use rather than high performance.
- The idea that revolutionized small computer operating systems was the BIOS (basic input-output operating system) chip that permitted a single operating system to function on different types of small systems.
  - The BIOS takes care of the details involved in addressing divergent peripheral device designs and protocols.

8.2 Operating Systems (6 of 12)

- Operating systems having graphical user interfaces were first brought to market in the 1980s.
- At one time, these systems were considered appropriate only for desktop publishing and games. Today they are seen as technology enablers for users with little formal computer education.
- Once solely a server operating system, Linux holds the promise of bringing Unix to ordinary desktop systems.

8.2 Operating Systems (7 of 12)

8.2.2 Operating Systems Design

- Two operating system components are crucial: The *kernel* and the system programs.
- As the core of the operating system, the kernel performs scheduling, synchronization, memory management, interrupt handling and it provides security and protection.
- *Microkernel* systems provide minimal functionality, with most services carried out by external programs.
- *Monolithic* systems provide most of their services within a single operating system program.

8.2 Operating Systems (8 of 12)

- Microkernel systems provide better security, easier maintenance, and portability at the expense of execution speed.
  - Examples are MINIX, Mach, and QNX.
  - Symmetric multiprocessor computers are ideal platforms for microkernel operating systems.
- Monolithic systems give faster execution speed, but are difficult to port from one architecture to another.
  - Examples are Linux, MacOS, and DOS.

8.2 Operating Systems (9 of 12)

8.2.3 Operating System Services

- Process management lies at the heart of operating system services.
  - The operating system creates processes, schedules their access to resources, deletes processes, and deallocates resources that were allocated during process execution.
- The operating system monitors the activities of each process to avoid synchronization problems that can occur when processes use shared resources.
- If processes need to communicate with one another, the operating system provides the services.

8.2 Operating Systems (10 of 12)

- The operating system schedules process execution.
- First, the operating system determines which process shall be granted access to the CPU.
  - This is *long-term scheduling*.
- After a number of processes have been admitted, the operating system determines which one will have access to the CPU at any particular moment.
  - This is *short-term scheduling*.
- Context switches occur when a process is taken from the CPU and replaced by another process.
  - Information relating to the state of the process is preserved during a context switch.

8.2 Operating Systems (11 of 12)

- Short-term scheduling can be nonpreemptive or preemptive
- In nonpreemptive scheduling, a process has use of the CPU until either it terminates or must wait for resources that are temporarily unavailable.
- In preemptive scheduling, each process is allocated a time slice. When the time slice expires, a context switch occurs.
- A context switch can also occur when a higher-priority process needs the CPU.

8.2 Operating Systems (12 of 12)

- Four approaches to CPU scheduling are:
  - First-come, first-served where jobs are serviced in arrival sequence and run to completion if they have all of the resources they need.
  - Shortest job first where the smallest jobs get scheduled first. (The trouble is in knowing which jobs are shortest!)
  - Round robin scheduling where each job is allotted a certain amount of CPU time. A context switch occurs when the time expires.
  - Priority scheduling preempts a job with a lower priority when a higher-priority job needs the CPU.

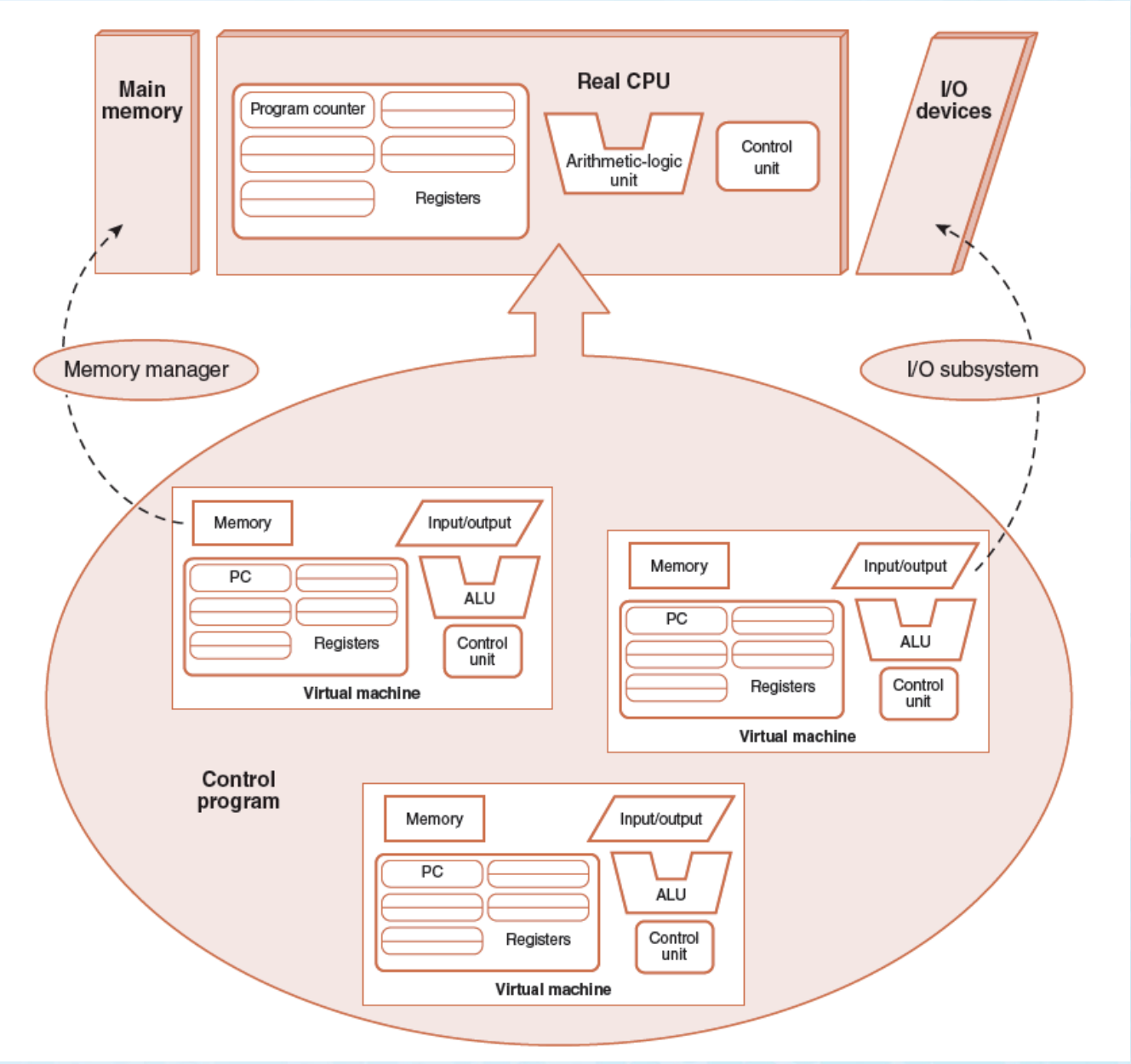
8.3 Protected Environments (1 of 7)

- In their role as resource managers and protectors, many operating systems provide protected environments that isolate processes, or groups of processes from each other.
- Three common approaches to establishing protected environments are virtual machines, subsystems, and partitions.
- These environments simplify system management and control, and can provide emulated machines to enable execution of programs that the system would otherwise be unable to run.

8.3 Protected Environments (2 of 7)

8.3.1 Virtual Machines

- Virtual machines are a protected environment that presents an image of itself—or the image of a totally different architecture—to the processes that run within the environment.
- A virtual machine is exactly that: an imaginary computer.
- The underlying real machine is under the control of the kernel. The kernel receives and manages all resource requests that emit from processes running in the virtual environment.
- The next slide provides an illustration.

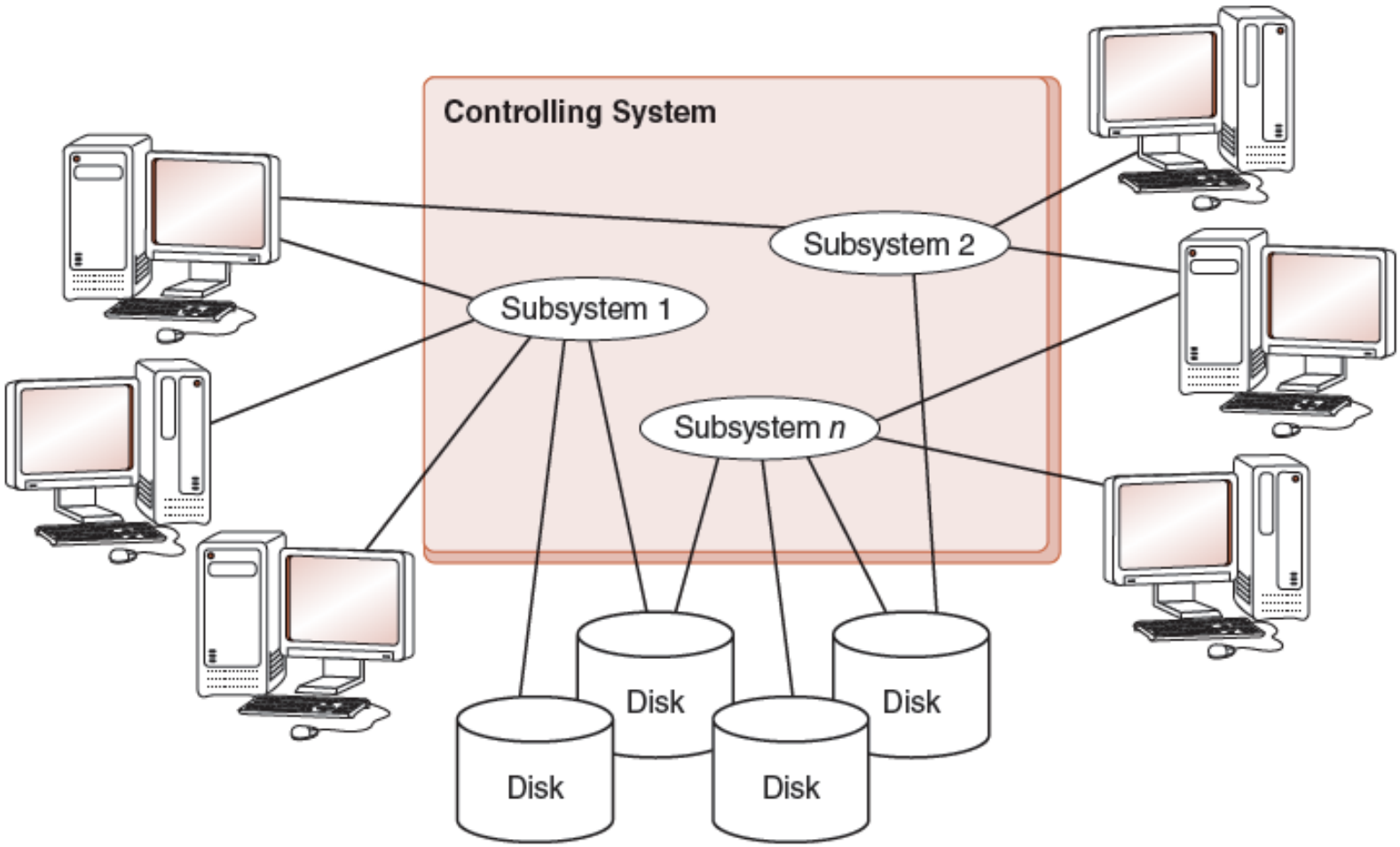


8.3 Protected Environments (4 of 7)

8.3.2 Subsystems and Partitions

- Subsystems are another type of protected environment.
- They provide logically distinct environments that can be individually controlled and managed. They can be stopped and started independent of each other.
  - Subsystems can have special purposes, such as controlling I/O or virtual machines. Others partition large application systems to make them more manageable.
  - In many cases, resources must be made visible to the subsystem before they can be accessed by the processes running within it.

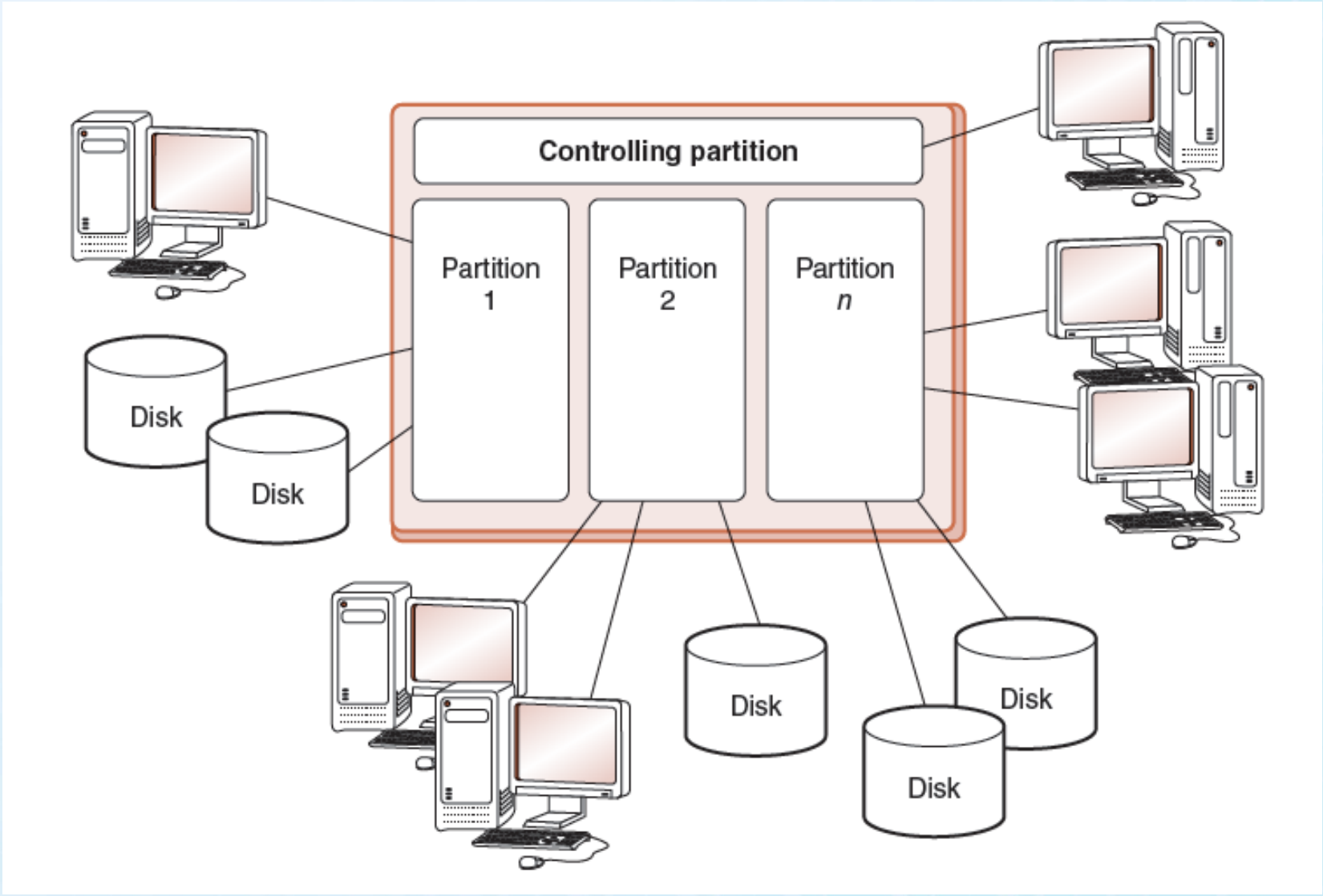
The next slide provides an illustration.



8.3 Protected Environments (6 of 7)

- In very large computers, subsystems do not go far enough to establish a protected environment.
- Logical partitions (LPARs) provide much higher barriers: Processes running within a logical partition have no access to processes running in another partition unless a connection between them (e.g., FTP) is explicitly established.
- LPARs are an enabling technology for the recent trend of consolidating hundreds of small servers within the confines of a single large system.

The next slide provides an illustration.



8.4 Programming Tools (1 of 13)

8.4.1 Assembler and Assembly

- Programming tools carry out the mechanics of software creation within the confines of the operating system and hardware environment.
- Assemblers are the simplest of all programming tools. They translate mnemonic instructions to machine code.
- Most assemblers carry out this translation in two passes over the source code.
  - The first pass partially assembles the code and builds the symbol table.
  - The second pass completes the instructions by supplying values stored in the symbol table.