Chapter Goals

• Describe the two main responsibilities of an operating system

• Define memory and process management

• Explain how timesharing creates the virtual machine illusion

• Explain the relationship between logical and physical addresses

• Compare and contrast memory management techniques
Chapter Goals

• Distinguish between fixed and dynamic partitions
• Define and apply partition selection algorithms
• Explain how demand paging creates the virtual memory illusion
• Explain the stages and transitions of the process life cycle
• Explain the processing of various CPU scheduling algorithms
Software Categories

• **Application software**  Software written to address specific needs—to solve problems in the real world
  
  Word processing programs, games, inventory control systems, automobile diagnostic programs, and missile guidance programs are all application software

• **System software**  Software that manages a computer system at a fundamental level
  
  It provides the tools and an environment in which application software can be created and run
An operating system

- manages computer resources, such as memory and input/output devices
- provides an interface through which a human can interact with the computer
- allows an application program to interact with these other system resources
Operating System

Figure 10.1
An operating system interacts with many aspects of a computer system.
The various roles of an operating system generally revolve around the idea of “sharing nicely”.

An operating system manages resources, and these resources are often shared in one way or another among programs that want to use them.
Resource Management

• **Multiprogramming** The technique of keeping multiple programs in main memory at the same time that compete for access to the CPU so that they can execute.

• **Memory management** The process of keeping track of what programs are in memory and where in memory they reside.
Resource Management

- **Process**  A program in execution
- The operating system performs process management to carefully track the progress of a process and all of its intermediate states
- **CPU scheduling** determines which process in memory is executed by the CPU at any given point
A typical computer in the 1960s and ‘70s was a large machine.

Its processing was managed by a human operator.

The operator would organize various jobs from multiple users into batches.
Batch Processing

Figure 10.2 In early systems, human operators would organize jobs into batches
Timesharing

- **Timesharing system** A system that allows multiple users to interact with a computer at the same time

- **Multiprogramming** A technique that allows multiple processes to be active at once, allowing programmers to interact with the computer system directly, while still sharing its resources

- In a timesharing system, each user has his or her own *virtual machine*, in which all system resources are (in effect) available for use
Other Factors

- **Real-time System** A system in which response time is crucial given the nature of the application.

- **Response time** The time delay between receiving a stimulus and producing a response.

- **Device driver** A small program that “knows” the way a particular device expects to receive and deliver information.
Memory Management

• Operating systems must employ techniques to
  – Track where and how a program resides in memory
  – Convert **logical addresses** into actual **addresses**

• **Logical address** (sometimes called a virtual or relative address) A value that specifies a generic location, relative to the program but not to the reality of main memory

• **Physical address** An actual address in the main memory device
Memory Management

Figure 10.3
Memory is a continuous set of bits referenced by specific addresses
Single Contiguous Memory Management

- There are only two programs in memory
  - The operating system
  - The application program

- This approach is called **single contiguous memory management**

**Figure 10.4**
Main memory divided into two sections
Single Contiguous Memory Management

• A logical address is simply an integer value relative to the starting point of the program

• To produce a physical address, we add a logical address to the starting address of the program in physical main memory
Single Contiguous Memory Management

Figure 10.5
binding a logical address to a physical one
Partition Memory Management

- **Fixed partitions**  Main memory is divided into a particular number of partitions
- **Dynamic partitions**  Partitions are created to fit the needs of the programs
Partition Memory Management

- At any point in time memory is divided into a set of partitions, some empty and some allocated to programs.

- **Base register** A register that holds the beginning address of the current partition.

- **Bounds register** A register that holds the length of the current partition.

Figure 10.6 Address resolution in partition memory management
Partition Selection Algorithms

Which partition should we allocate to a new program?

• **First fit**  Allocate program to the first partition big enough to hold it

• **Best fit**  Allocated program to the smallest partition big enough to hold it

• **Worst fit**  Allocate program to the largest partition big enough to hold it
Paged Memory Management

- **Paged memory technique**: A memory management technique in which processes are divided into fixed-size **pages** and stored in memory **frames** when loaded into memory.
  - **Frame**: A fixed-size portion of main memory that holds a process page.
  - **Page**: A fixed-size portion of a process that is stored into a memory frame.
  - **Page-map table (PMT)**: A table used by the operating system to keep track of page/frame relationships.
Paged Memory Management

- To produce a physical address, you first look up the page in the PMT to find the frame number in which it is stored.
- Then multiply the frame number by the frame size and add the offset to get the physical address.

Figure 10.7
A paged memory management approach
Paged Memory Management

• **Demand paging** An important extension of paged memory management
  – Not all parts of a program actually have to be in memory at the same time
  – In demand paging, the pages are brought into memory on demand

• **Page swap** The act of bringing in a page from secondary memory, which often causes another page to be written back to secondary memory
Paged Memory Management

• The demand paging approach gives rise to the idea of *virtual memory*, the illusion that there are no restrictions on the size of a program.

• Too much page swapping, however, is called *thrashing* and can seriously degrade system performance.
Process Management

- The Process States

Figure 10.8  The process life cycle
The Process Control Block

• The operating system must manage a large amount of data for each active process

• Usually that data is stored in a data structure called a process control block (PCB)

• Each state is represented by a list of PCBs, one for each process in that state
Keep in mind that there is only one CPU and therefore only one set of CPU registers

- These registers contain the values for the currently executing process

Each time a process is moved to the running state:

- Register values for the currently running process are stored into its PCB
- Register values of the new running state are loaded into the CPU
- This exchange of information is called a context switch
CPU Scheduling

- **CPU Scheduling** The act of determining which process in the *ready* state should be moved to the *running* state
  - Many processes may be in the ready state
  - Only one process can be in the running state, making progress at any one time

- *Which one gets to move from ready to running?*
CPU Scheduling

- **Nonpreemptive scheduling**  The currently executing process gives up the CPU voluntarily
- **Preemptive scheduling**  The operating system decides to favor another process, preempting the currently executing process
- **Turnaround time**  The amount of time between when a process arrives in the ready state the first time and when it exits the running state for the last time
CPU Scheduling Algorithms

First-Come, First-Served
– Processes are moved to the CPU in the order in which they arrive in the running state

Shortest Job Next
– Process with shortest estimated running time in the ready state is moved into the running state first

Round Robin
– Each process runs for a specified time slice and moves from the running state to the ready state to await its next turn if not finished
### First-Come, First-Served

<table>
<thead>
<tr>
<th>Process</th>
<th>Service time</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>140</td>
</tr>
<tr>
<td>p2</td>
<td>75</td>
</tr>
<tr>
<td>p3</td>
<td>320</td>
</tr>
<tr>
<td>p4</td>
<td>280</td>
</tr>
<tr>
<td>p5</td>
<td>125</td>
</tr>
</tbody>
</table>

![Time chart for processes](chart.png)
Shortest Job Next

• Looks at all processes in the ready state and dispatches the one with the smallest service time
Round Robin

- Distributes the processing time equitably among all ready processes

- The algorithm establishes a particular **time slice** (or time quantum), which is the amount of time each process receives before being preempted and returned to the ready state to allow another process its turn
Round Robin

- Suppose the time slice was 50
CPU Scheduling Algorithms

Are they preemptive or non-preemptive? Explain

• First-Come, First-Served?

• Shortest Job Next?

• Round Robin?