

Unit - I

Operating System

Overview



Operating System

- A program that controls the execution of application programs
- A program that controls the execution of users programs
- An interface between applications and hardware

I. Operating System Objectives

- Convenience

- Makes the computer more convenient to use

- Efficiency

- Allows computer system resources to be used in an efficient manner

- Ability to evolve

- Permit effective development, testing, and introduction of new system functions without interfering with service

2. Layers of Computer System

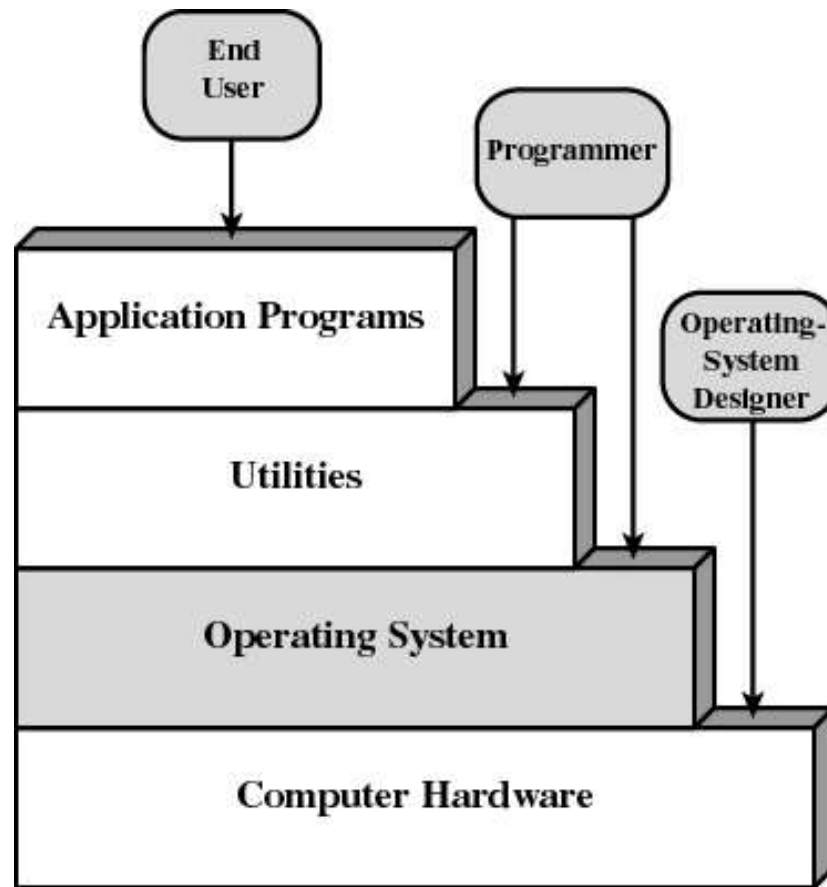


Figure 2.1 Layers and Views of a Computer System

Kernel

- Heart (vital portion) of the operating system
- Next to Hardware
- It always resides in main memory
- Contains most-frequently used functions
- Also called the nucleus

3. Services Provided by the Operating System

- Program development
 - Editors, compilers, linker, loader and debuggers
- Program execution
- Access to I/O devices
- Controlled access to files
- Access to other plug and play devices
- Security to users programs and applications

Services Provided by the Operating System

- Error detection and response
 - internal and external hardware errors
 - memory error
 - device failure
 - software errors
 - arithmetic overflow
 - access forbidden memory locations



Services Provided by the Operating System

- Accounting
 - collect statistics
 - monitor performance

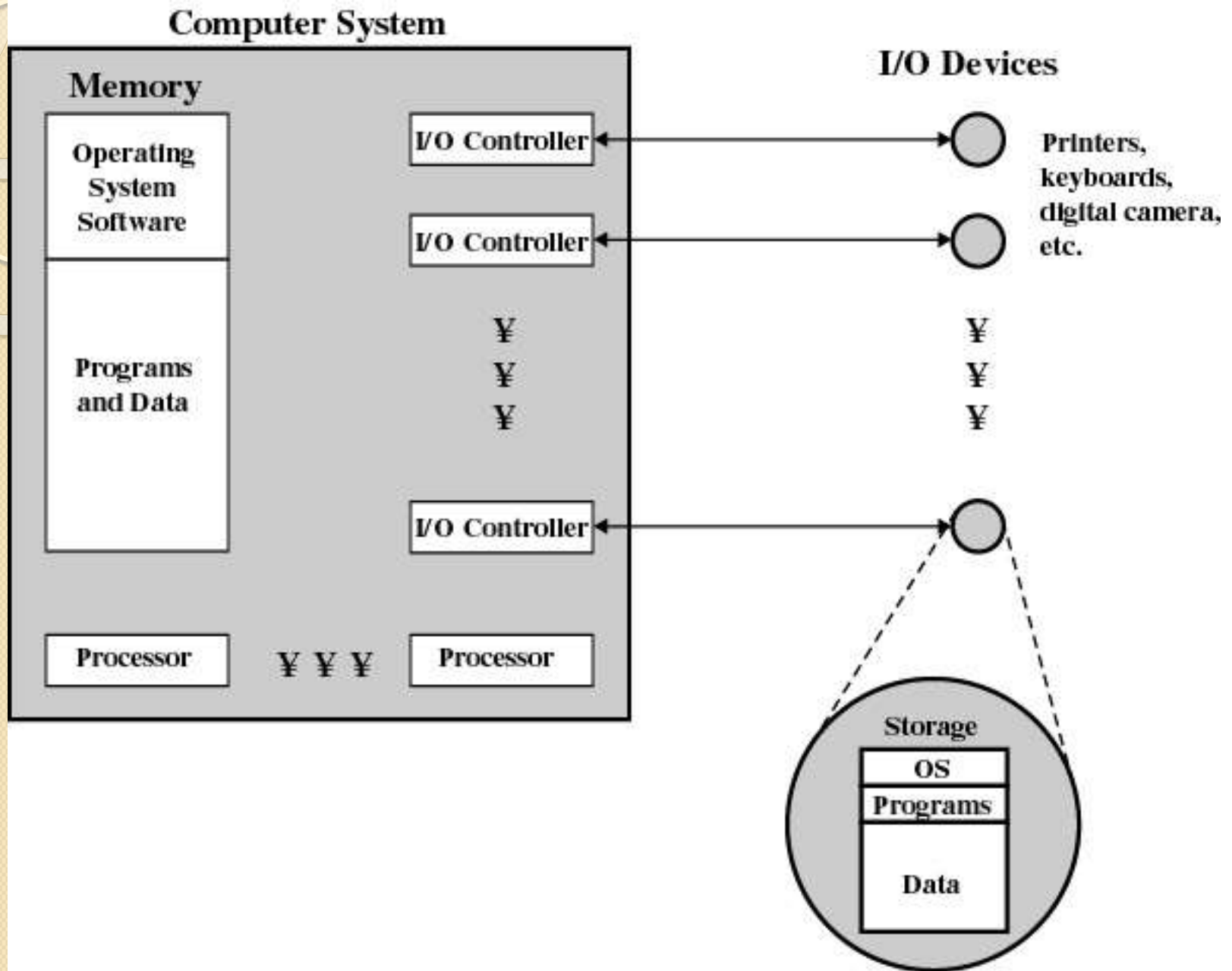


Figure 2.2 The Operating System as Resource Manager

4. Evolution of Operating Systems

4.1 Serial Processing

- No operating system
- Machines run from a console with display lights and toggle switches, input device, and printer
- Schedule time
- Setup included loading the compiler, source program, saving compiled program, and loading and linking

Evolution of Operating Systems

4.2 Simple Batch Systems

- Monitors

- Software that controls the running programs
- Batch jobs together
- Program branches back to monitor when finished
- Resident monitor is in main memory and available for execution

Job Control Language (JCL)

- Special type of programming language
- Provides instruction to the monitor
 - what compiler to use
 - what data to use

Hardware Features

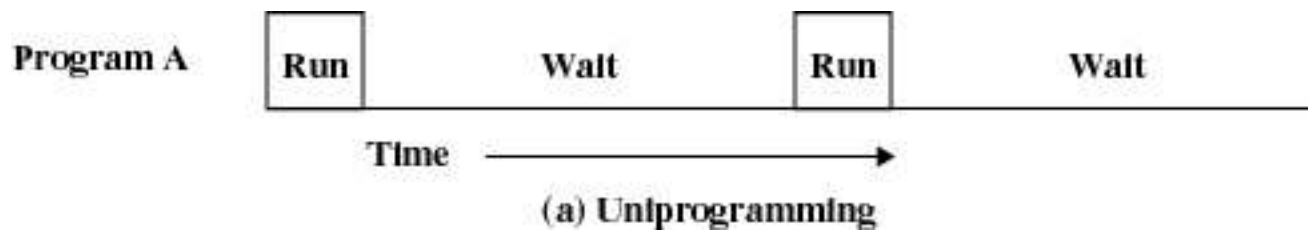
- Memory protection
 - do not allow the memory area containing the monitor to be altered
- Timer
 - prevents a job from monopolizing the system

Hardware Features

- Memory protection
 - do not allow the memory area containing the monitor to be altered
- Timer
 - prevents a job from monopolizing the system

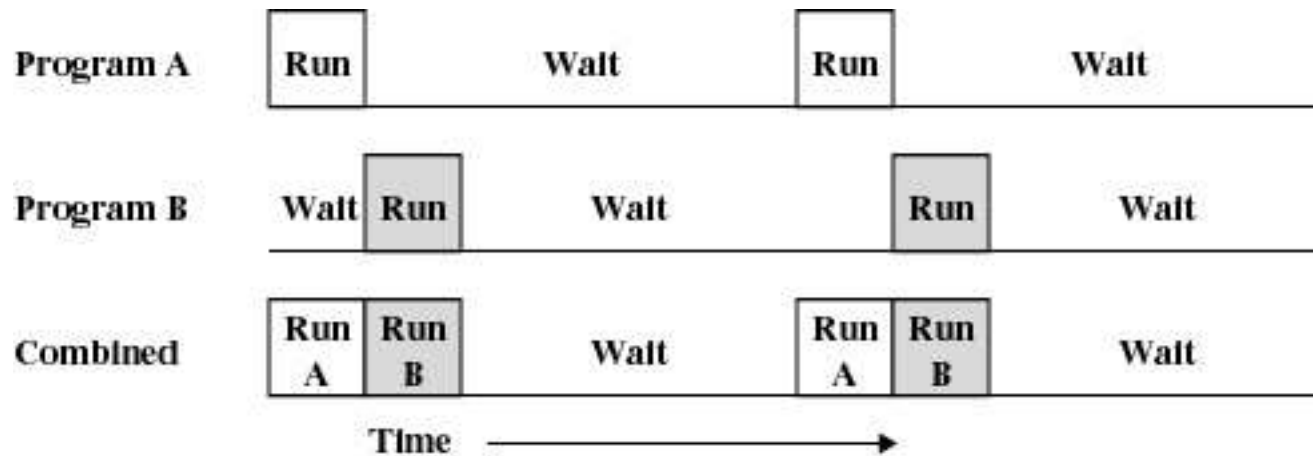
4.3 Uniprogramming

- Processor must wait for I/O instruction to complete before proceeding



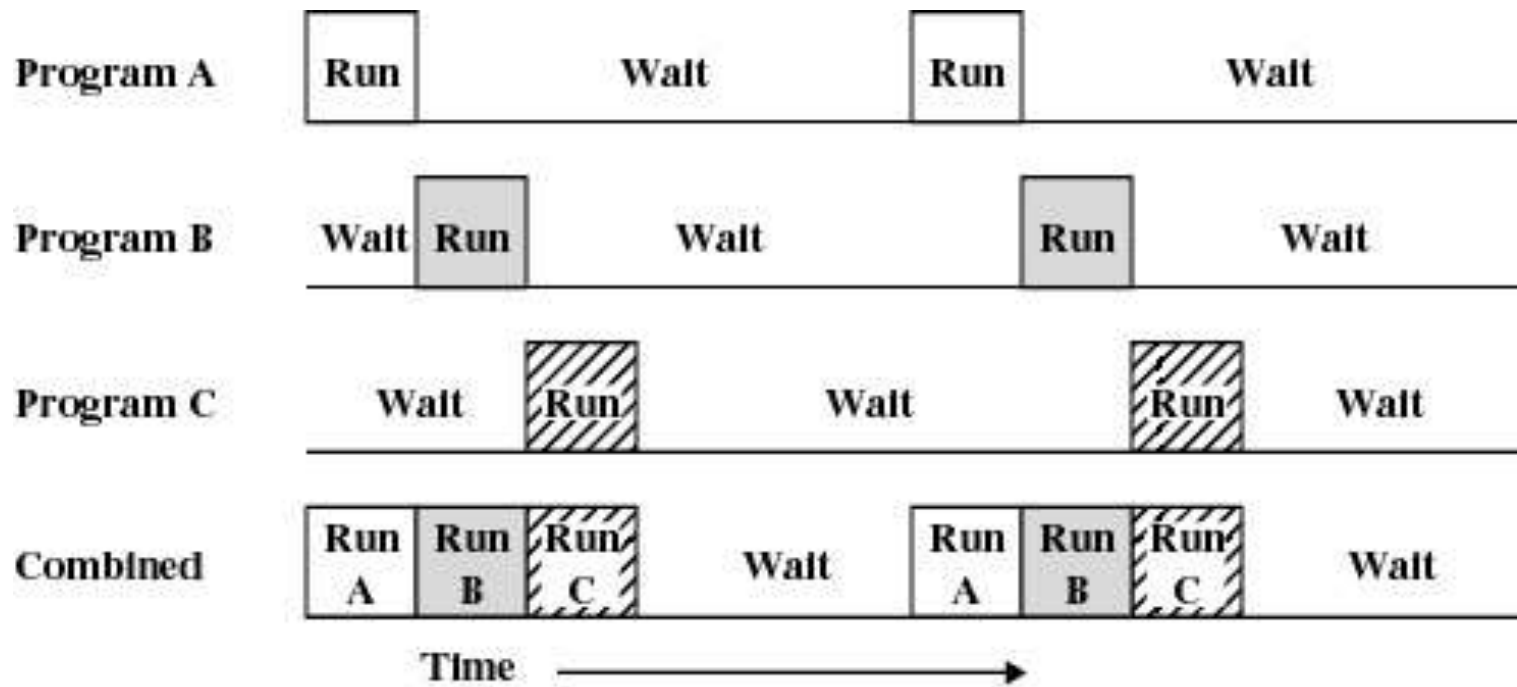
4.4 Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

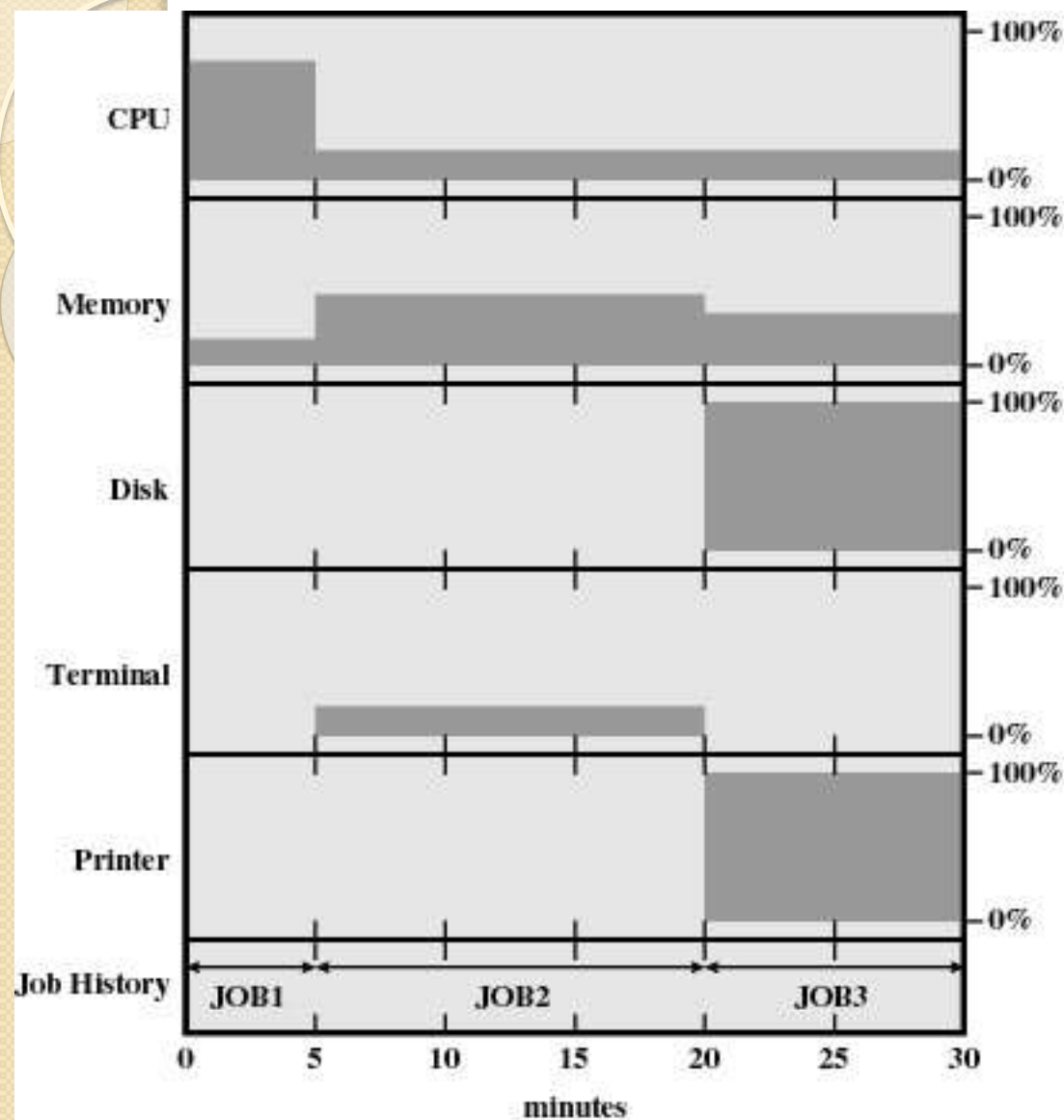
Multiprogramming



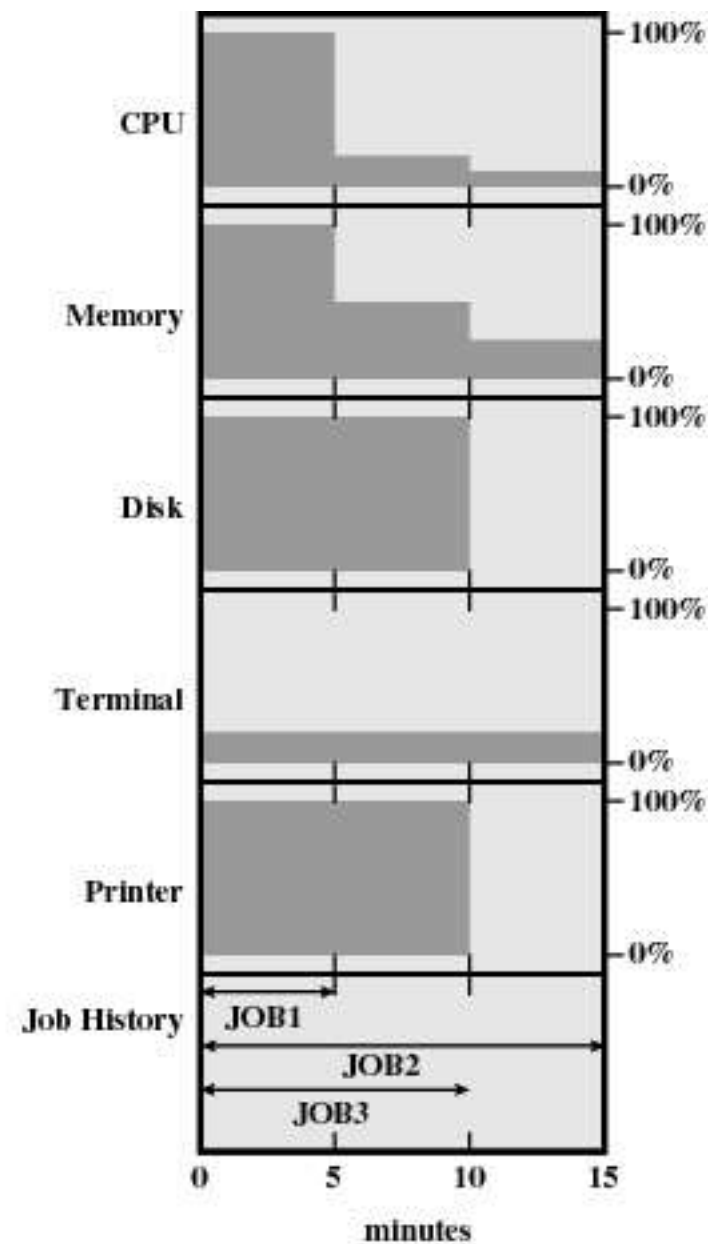
(c) Multiprogramming with three programs

Example

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min.	15 min.	10 min.
Memory required	50K	100 K	80 K
Need disk?	No	No	Yes
Need terminal	No	Yes	No
Need printer?	No	No	Yes



(a) Uniprogramming



(b) Multiprogramming

Figure 2.6 Utilization Histograms

Effects of Multiprogramming

	Uniprogramming	Multiprogramming
Processor use	22%	43%
Memory use	30%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min.	15 min.
Throughput rate	6 jobs/hr	12 jobs/hr

4.5 Time Sharing

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals

Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Compatible time sharing system (developed by MIT)

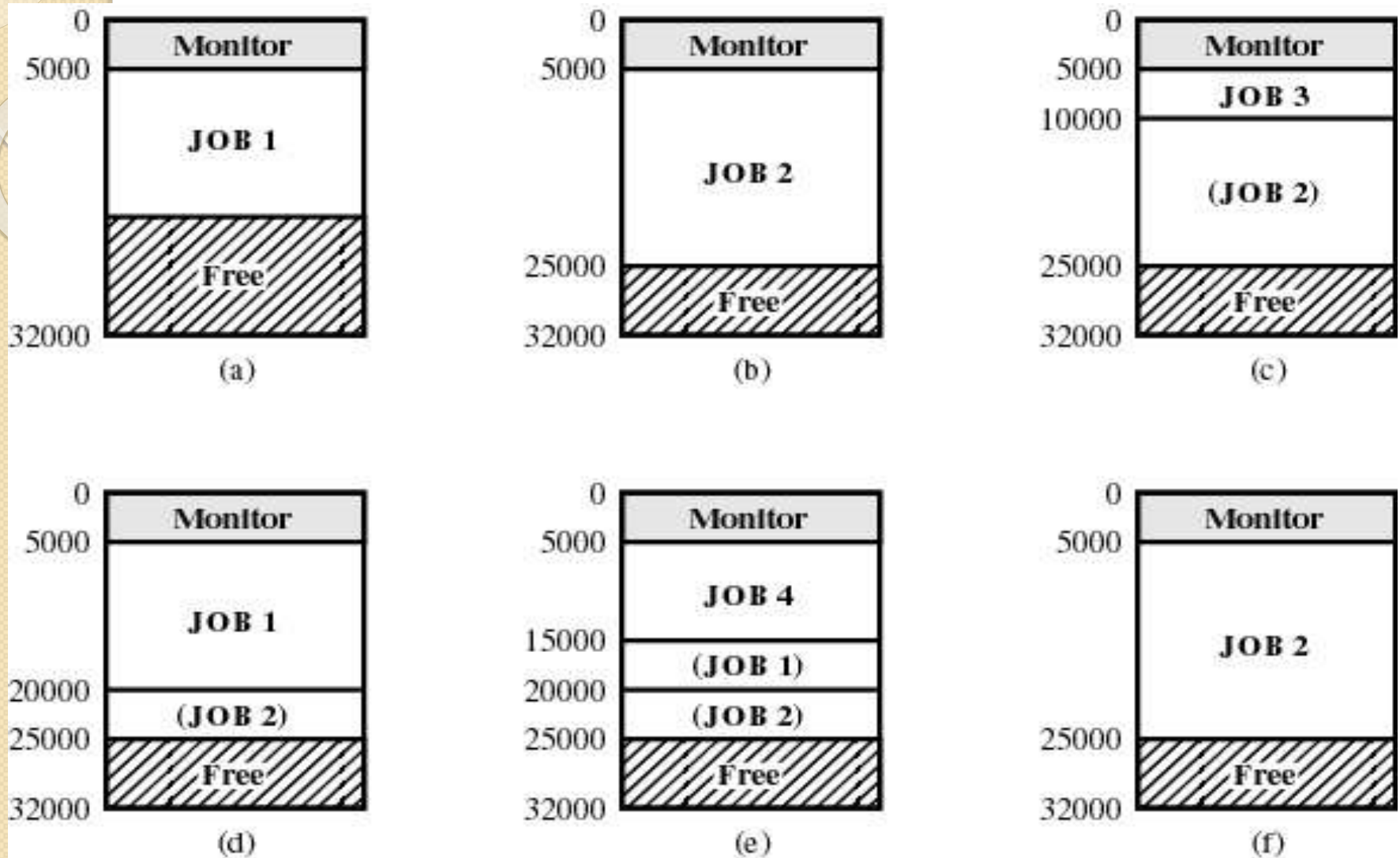


Figure 2.7 CTSS Operation

Major Achievements

- Running processes simultaneously
- Memory Management
- Information protection and security
- Scheduling and resource management



5.

Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

Process

- Consists of three components
 - An executable program
 - Associated data needed by the program
 - Execution context of the program
 - All information the operating system needs to manage the process

Process

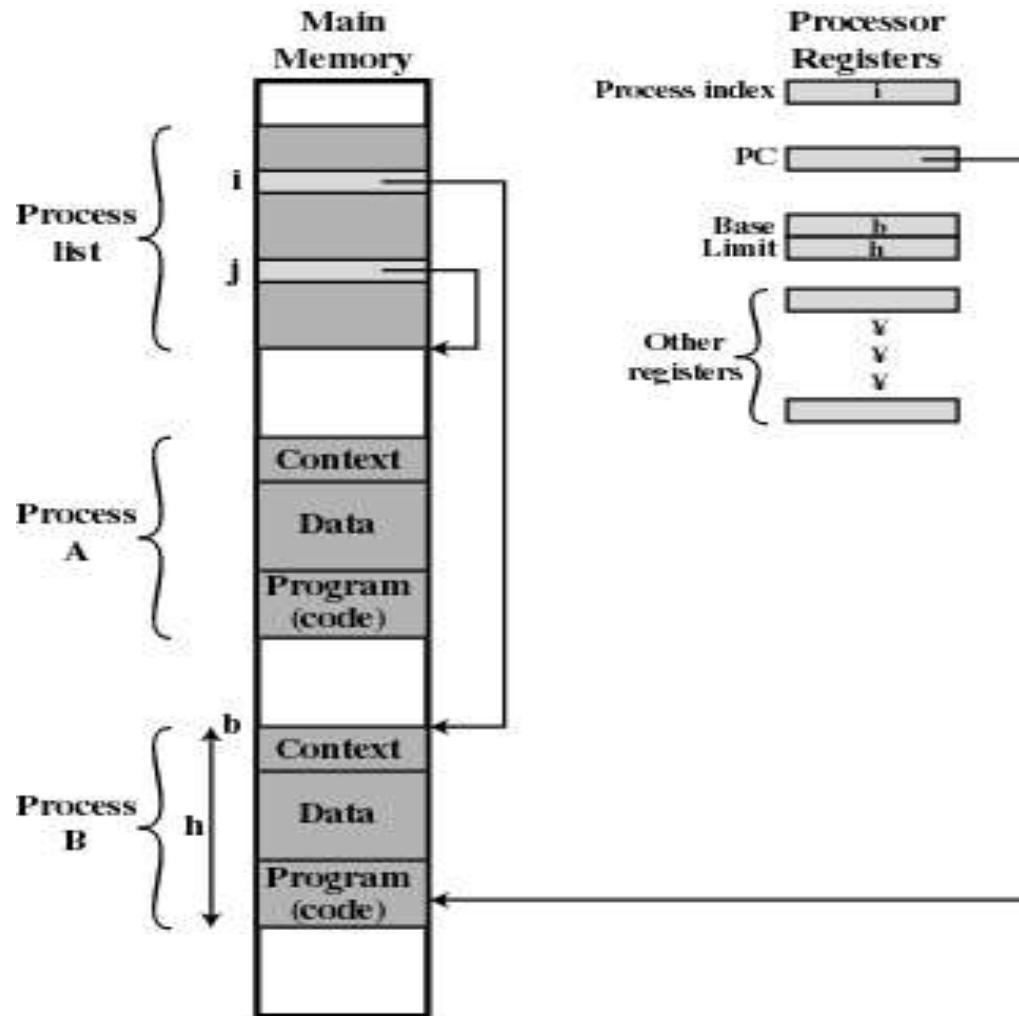


Figure 2.8 Typical Process Implementation



6. Memory Management

- Process isolation
- Automatic allocation and management
- Support for modular programming
- Protection and access control
- Long-term storage

Virtual Memory

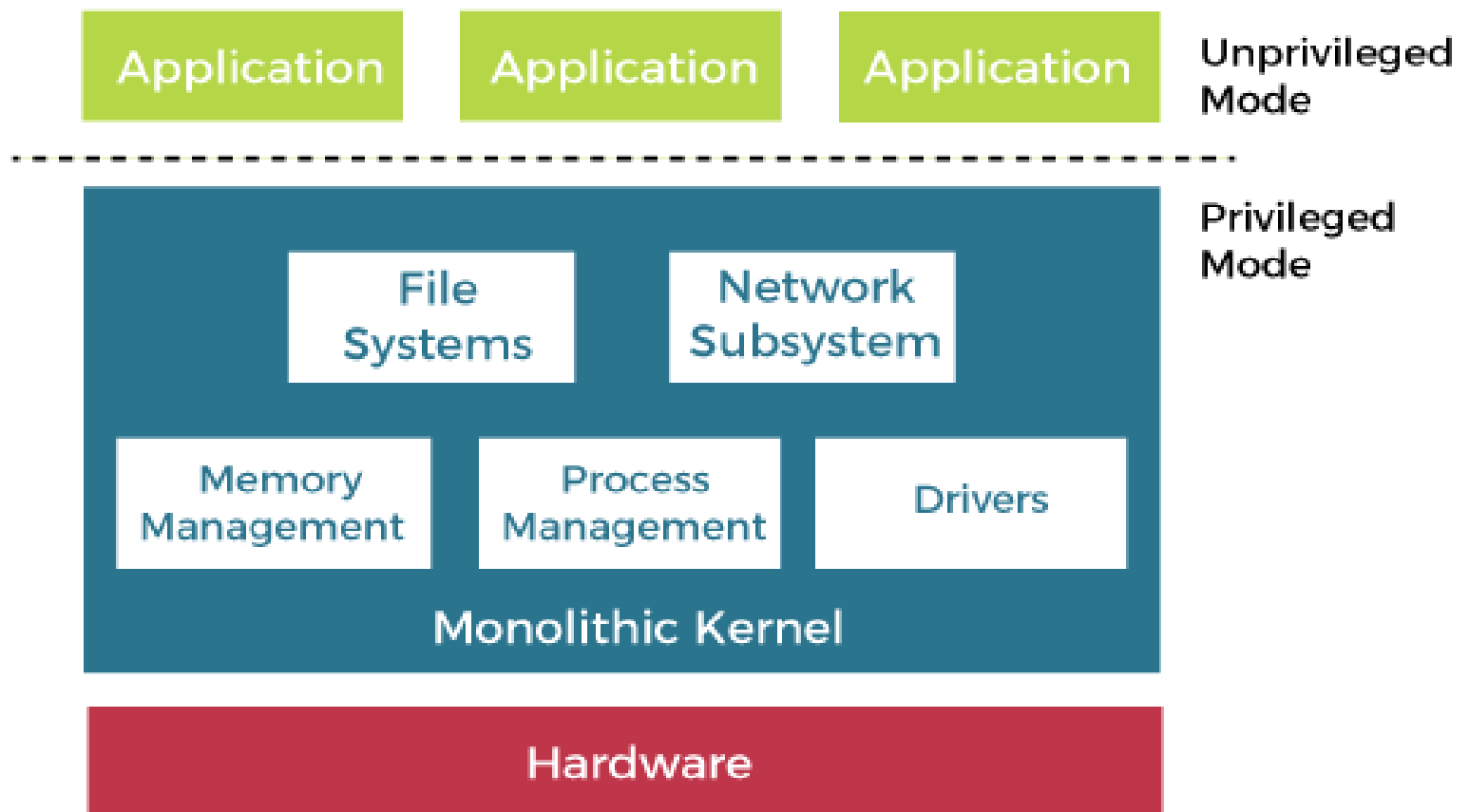
- Allows programmers to address memory from a logical point of view
- While one process is written out to secondary store and the successor process read in there in no interruption

File System

- Implements long-term store
- Information stored in named objects called files

Monolithic Architecture

Monolithic Kernel System





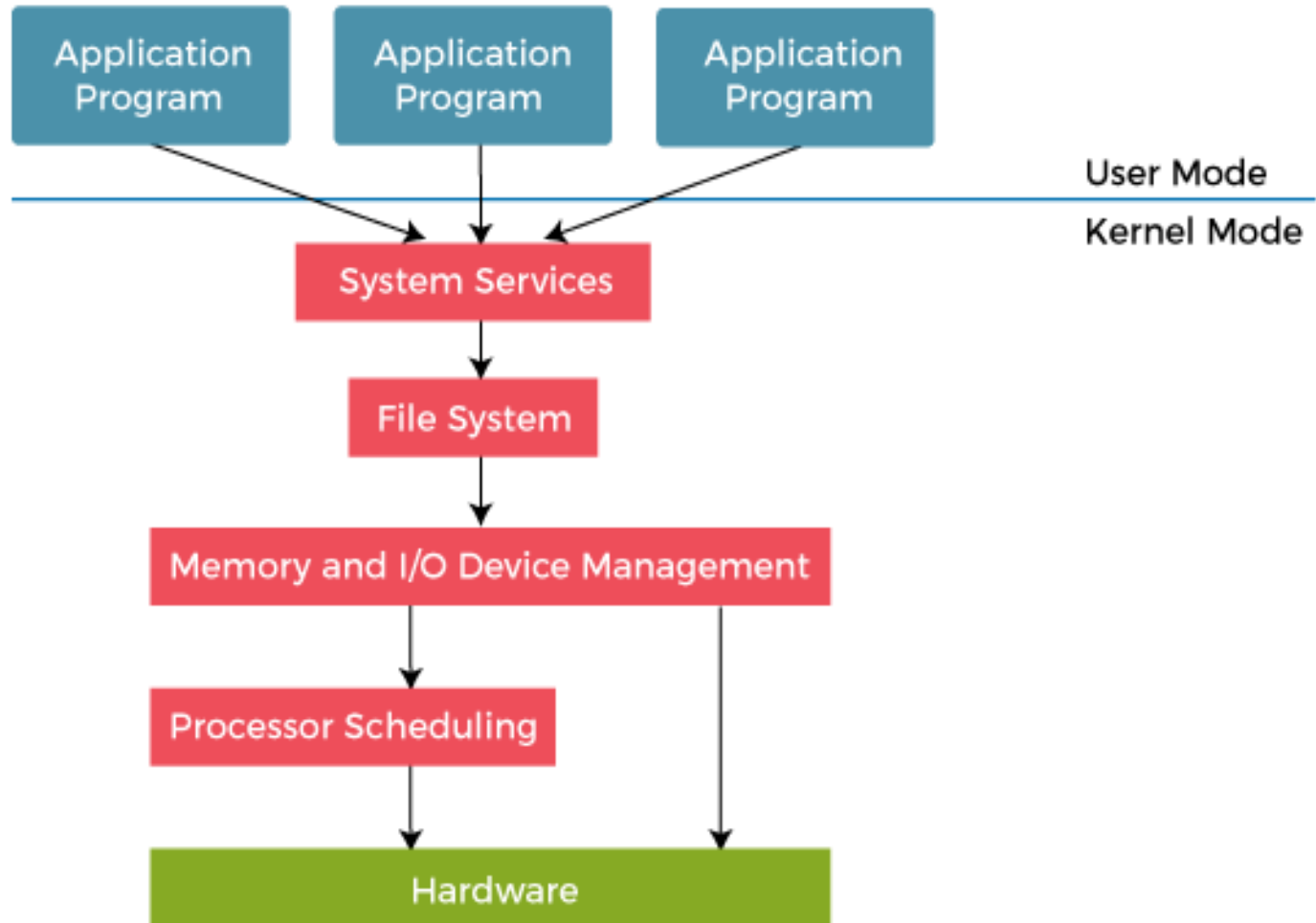
Advantages of Monolithic Architecture:

- Simple and easy to implement structure.
- Faster execution due to direct access to all the services

Disadvantages of Monolithic Architecture:

- The addition of new features or removal of obsolete features is very difficult.
- Security issues are always there because there is no isolation among various servers present in the kernel.

Kernel and User Space in Layered Architecture



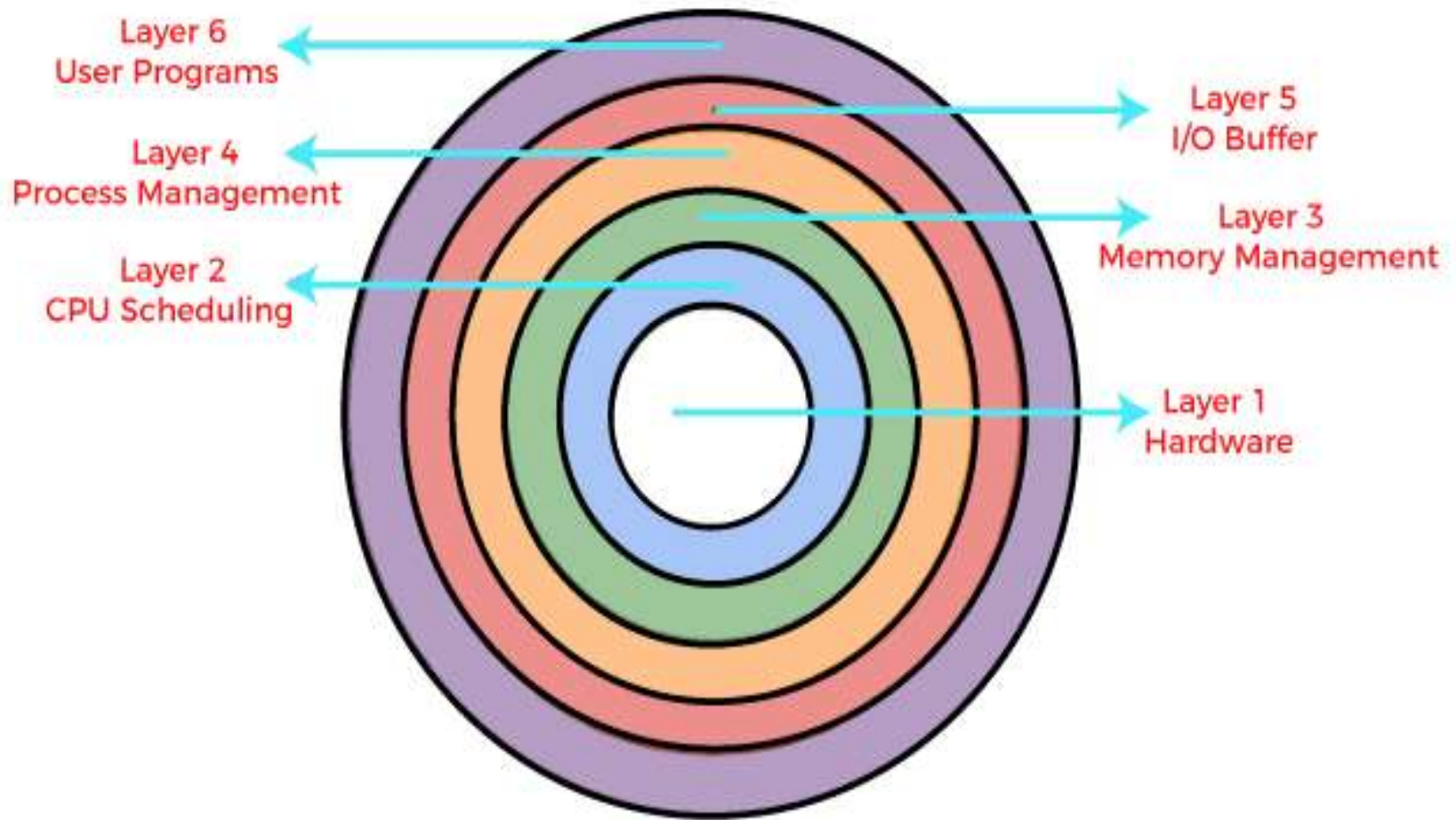
Advantages of Layered Structure

- **Modularity:** This design promotes modularity as each layer performs only the tasks it is scheduled to perform.
- **Easy debugging:** As the layers are discrete so it is very easy to debug. Suppose an error occurs in the CPU scheduling layer. The developer can only search that particular layer to debug, unlike the Monolithic system where all the services are present.
- **Easy update:** A modification made in a particular layer will not affect the other layers.
- **No direct access to hardware:** The hardware layer is the innermost layer present in the design. So a user can use the services of hardware but cannot directly modify or access it, unlike the Simple system in which the user had direct access to the hardware.
- **Abstraction:** Every layer is concerned with its functions. So the functions and implementations of the other layers are abstract to it.

Disadvantages of Layered Structure

- **Complex and careful implementation:** As a layer can access the services of the layers below it, so the arrangement of the layers must be done carefully. For example, the backing storage layer uses the services of the memory management layer. So it must be kept below the memory management layer. Thus with great modularity comes complex implementation.
- **Slower in execution:** If a layer wants to interact with another layer, its requests have to travel through all the layers present between the two interacting layers. Thus it increases response time, unlike the Monolithic system, which is faster than this. Thus an increase in the number of layers may lead to a very inefficient design.
- **Functionality:** It is not always possible to divide the functionalities. Many times, they are interrelated and can't be separated.
- **Communication:** No direct communication between non-adjacent layers.

Layered Architecture



Information Protection and Security

- Access control
 - regulate user access to the system
- Information flow control
 - regulate flow of data within the system and its delivery to users
- Certification
 - proving that access and flow control perform according to specifications

7. Scheduling and Resource Management

- Fairness
 - give equal and fair access to all processes
- Differential responsiveness
 - discriminate between different classes of jobs
- Efficiency
 - maximize throughput, minimize response time, and accommodate as many uses as possible

Major Elements of Operating System

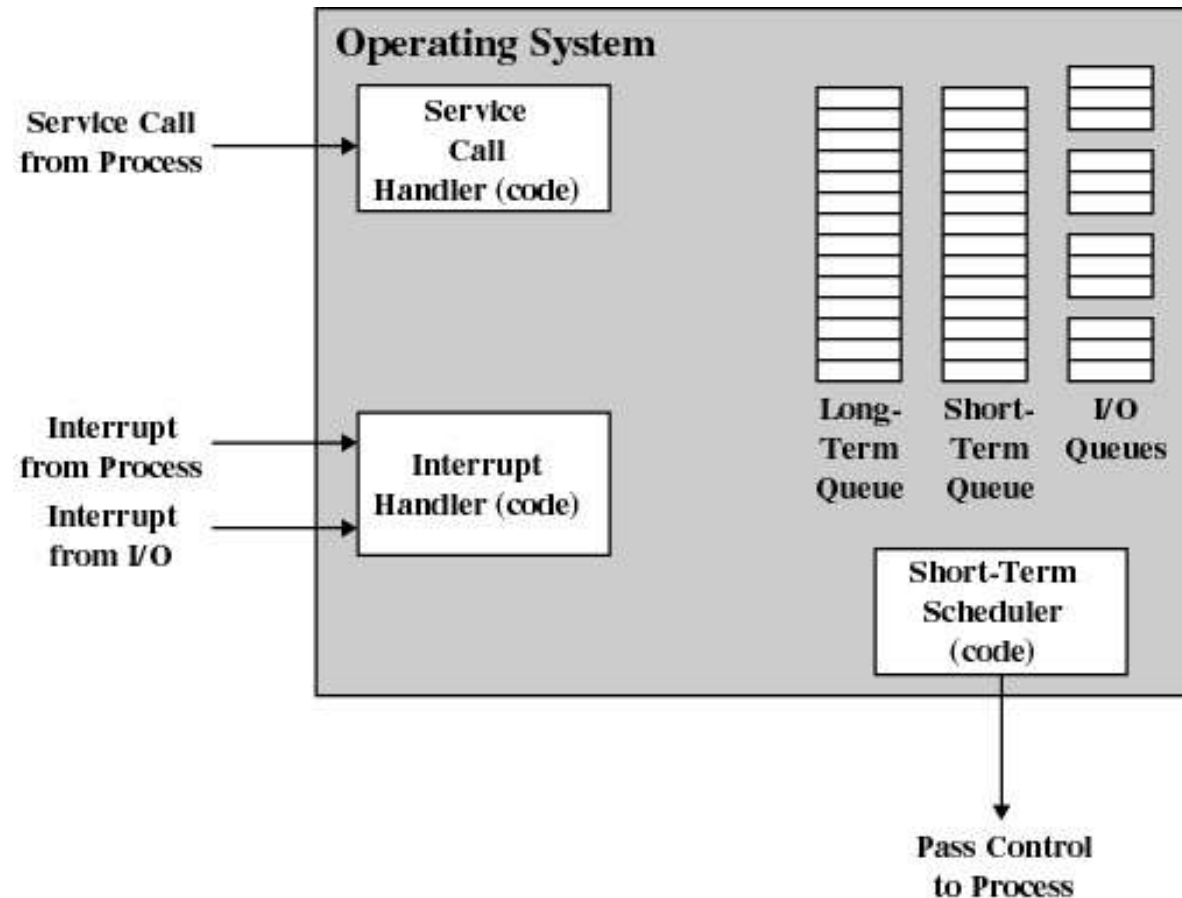


Figure 2.11 Key Elements of an Operating System for Multiprogramming



System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

8. Operating System Design Hierarchy

Level name

Objects

Example Operations

13. Shell

User programming

Statements in shell
language environment

12. User processes

User processes

Quit, kill, suspend, resume

11. Directories

Directories

Create, destroy, attach,
search, list

10. Devices

External devices, such
as printer, displays
and keyboards

Open, close,
read, write

9. File system

Files

Create, destroy, open, close,
read, write

8. Communications

Pipes

Create, destroy, open.
close, read, write

Operating System Design Hierarchy

Level name	Objects	Example Operations
7.Virtual Memory	Segments, pages	Read, write, fetch
6. Local secondary	Blocks of data, device	Read, write, allocate, free store channels
5. Primitive processes	primitive process,	Suspend, resume, wait, signal,semaphores, ready list


Operating System Design Hierarchy

Level name	Objects	Example Operations
4 Interrupts	Interrupt-handling	Invoke, mask, unmask, Programs retry
3 Procedures	Procedure, call stack,	Mark stack, call, return
2 Instruction Set subtract	Evaluation stack, micro- program interpreter, branch scalar and array data	Load, store, add,
1 Electronic circuit	Registers, gates, buses,	Clear, transfer, activate, etc.

5.


Characteristics of Modern Operating Systems

- Microkernel architecture
 - assigns only a few essential functions to the kernel
 - address space
 - Inter-process communication (IPC)
 - basic scheduling



Characteristics of Modern Operating Systems

- Multithreading
 - process is divided into threads that can run simultaneously
 - Thread
 - dispatchable unit of work
 - executes sequentially and is interruptible
 - Process is a collection of one or more threads




Characteristics of Modern Operating Systems

- Multithreading
 - process is divided into threads that can run simultaneously
 - Thread
 - dispatchable unit of work
 - executes sequentially and is interruptible
 - Process is a collection of one or more threads

Characteristics of Modern Operating Systems

- Distributed operating systems
 - provides the illusion of a single main memory and single secondary memory space
 - used for distributed file system



Characteristics of Modern Operating Systems

- Object-oriented design
 - used for adding modular extensions to a small kernel
 - enables programmers to customize an operating system without disrupting system integrity