Introduction

CS3008 Operating Systems

Lecture 01

What is an Operating System

An Operating System is a program that controls the execution of user programs and acts as an intermediary between users and computer hardware

- It is a software layer between application programs and computer hardware

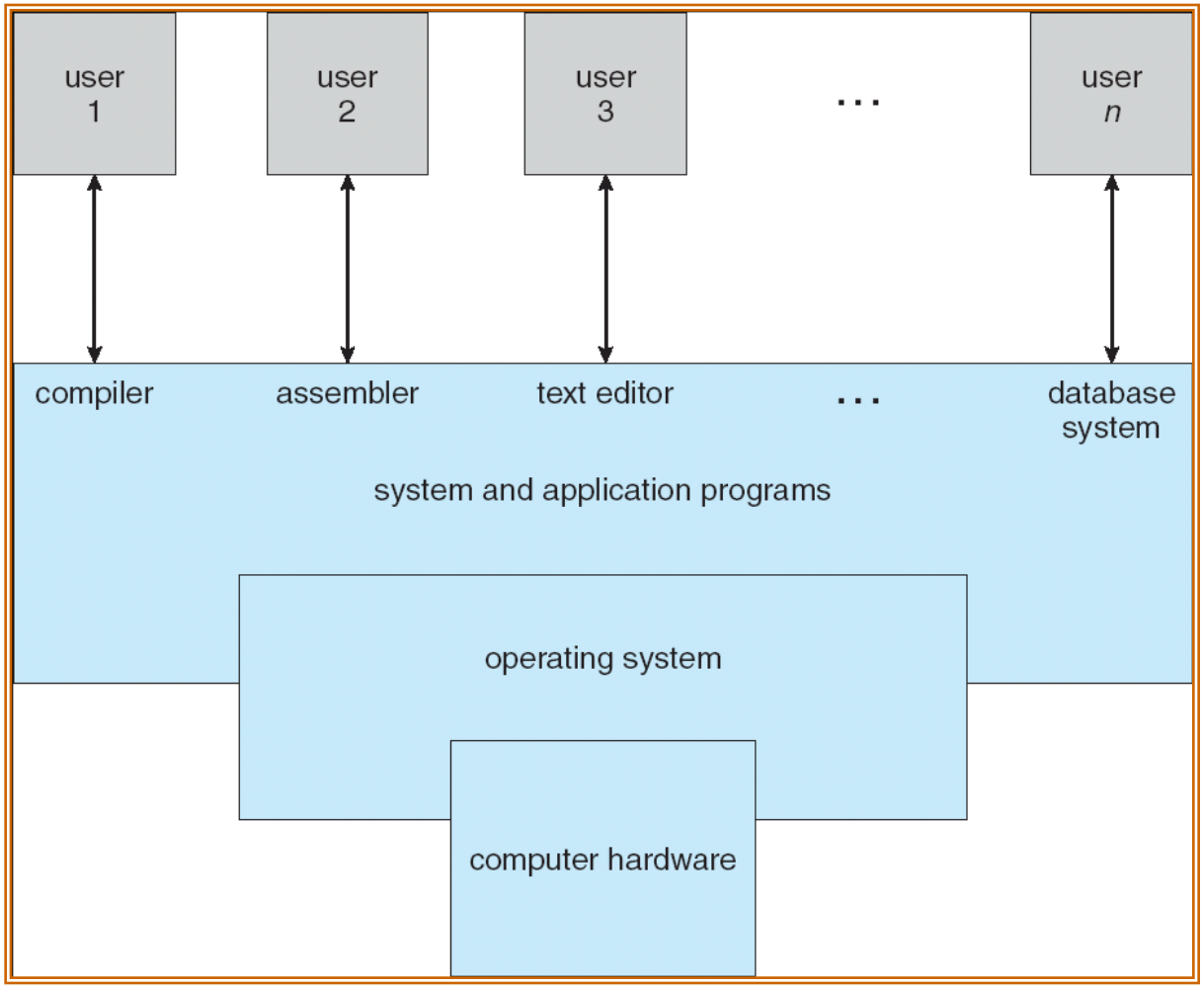
What is an Operating System

- It provides a basis and execution environment for application programs
  - Uniform abstract representation of resources that can be requested and accessed by applications
    - Processor, memory, I/O (disk, network)
  - Exploits the hardware resources of one or more processors
  - Manages secondary memory and I/O devices
- Goals
  - Make a computing system convenient to use
  - Make a computer system efficient to use
  - Make a computing system secure to use

Computer System Components

- Hardware
  - Basic computing resources: processor (CPU), memory, I/O devices
- Operating system
  - Controls and coordinates the use of this hardware among multiple programs running on a computer
- Application program
  - Solve user-specific problems: compilers, database systems, business applications
- User
  - People, other application programs (inter-process communication, distributed systems)

Computer System



The Role of an Operating System

- Service provider
  - Provide a set of services to system users
- Resource allocator
  - Exploit the hardware resources of one or more processors and allocate it to user programs
- Control program
  - Control the execution of programs and operations of I/O devices
    - interrupt them to send/receive data via I/O or to re-allocate hardware resources to other user programs
- Protection and Security
  - Protect multiple programs running from each other
  - Secure user access to data and define ownership of files and processes

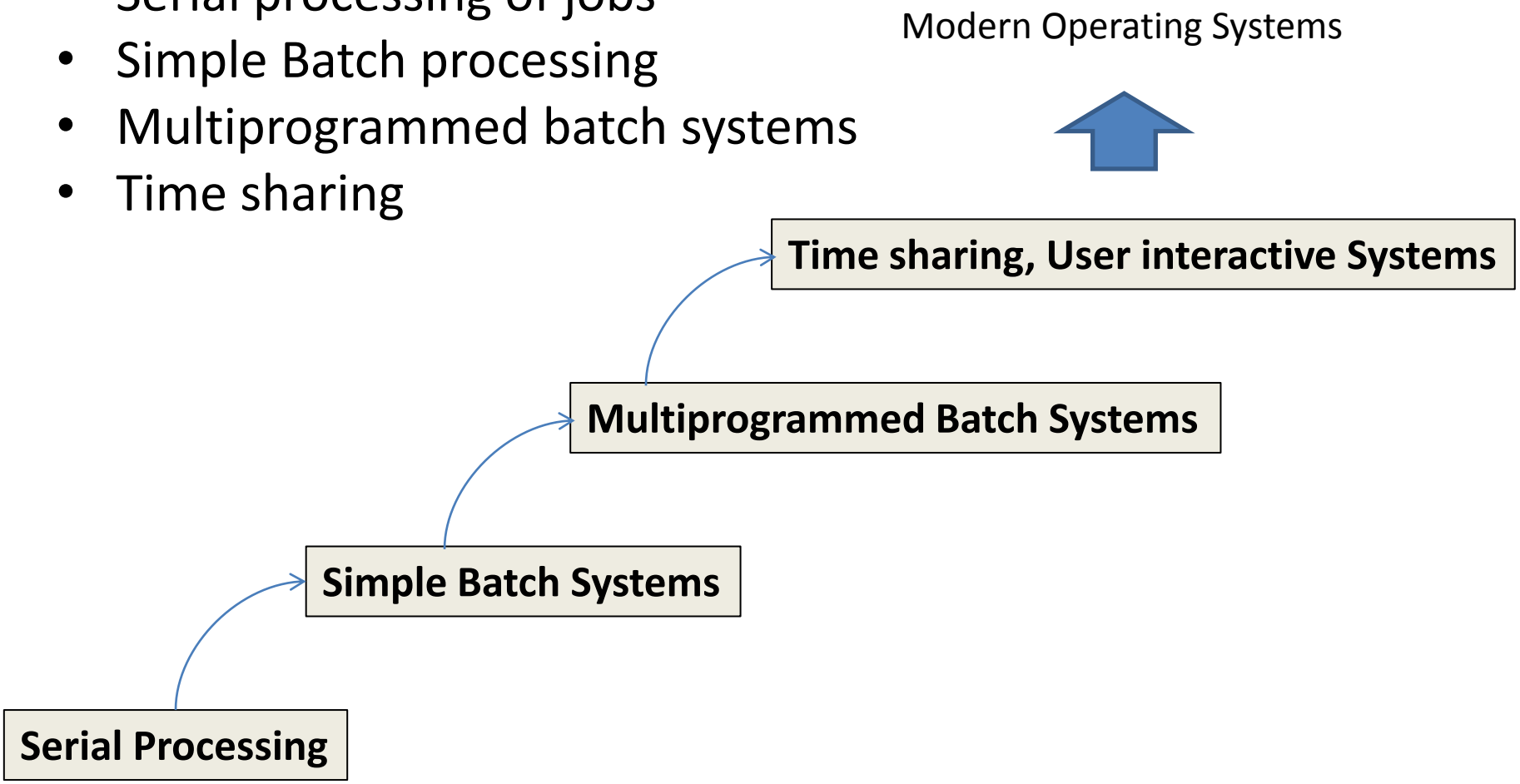
Basic Problems

- Multiple users run multiple programs on the same hardware
- Space and time sharing
  - Share processor time between multiple programs in a fair and optimised manner
  - Share access time to I/O devices
  - Share memory space among multiple programs
  - Share hard disk space
- Protection and Security
  - Protect applications from each other
  - Protect the operating system from malfunctioning and malicious applications
  - Protect data from unauthorized access

History

Evolution of Operating Systems

- Serial processing of jobs
- Simple Batch processing
- Multiprogrammed batch systems
- Time sharing



Earliest Computers: Serial Processing

- No operating system
- A programmer interacted directly with the computer hardware
- Problem
  - Setup time: considerable time spent on setting up the program to run
    - Direct access to all hardware
    - Difficult to program
  - No concepts of automated job scheduling
    - Users had to reserve computer time on a signup sheet
    - Waste of capacity

Simple Batch Processing

- Batch processing
  - Reduced setup time by batching similar “jobs”
- These were the first “Mainframe” systems
- Automatic job sequencing, automatic transfer from one job to another
  - Job Control Language
- Resident Monitor:
  - First rudimentary operating system
  - Control of processor is switched between monitor and user program

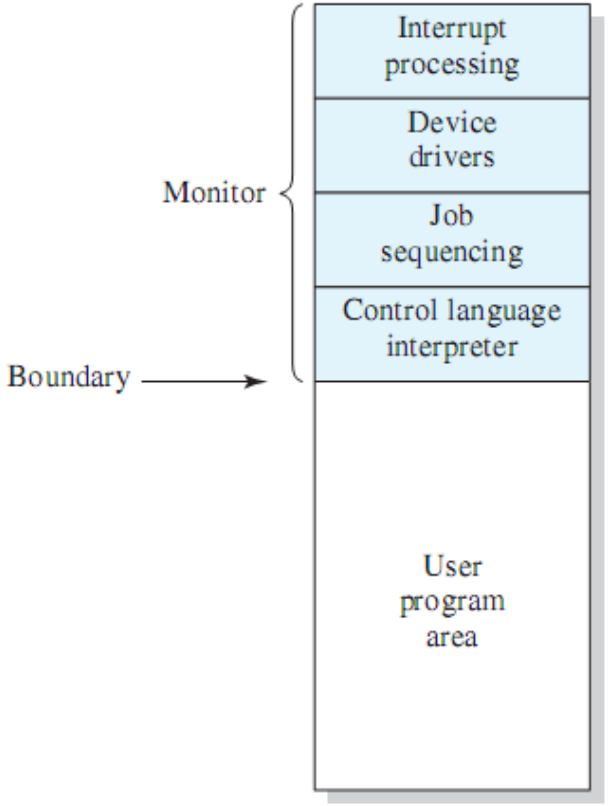


Figure 2.3 **Memory Layout for a Resident Monitor**

Simple Batch Processing

- Resident Monitor
  - Is software that is held permanently memory
  - Controls sequence of events
  - Includes interpreter for a job control language
- Activities
  - Loading jobs
    - User program
    - Additional programs such as compilers
    - Data to be processed
  - Load additional non-resident monitor elements and common functions needed by a program as sub-routines on demand

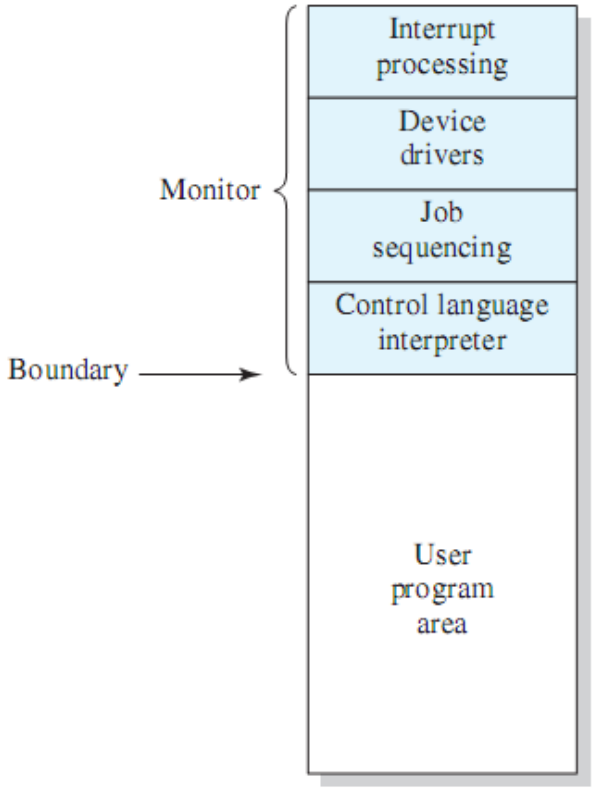


Figure 2.3 **Memory Layout for a Resident Monitor**

Simple Batch Processing

- Job Control Language (JCL)
  - Is a specialised type of programming language used to provide instructions to the monitor
    - What compiler to use
    - Loading the program code
    - Loading data

Simple Batch Processing

- Monitor point-of-view
  - Controls sequence of events
  - Execution cycle
    - Monitor loads a job and hands over control to the loaded program
    - Program executes
    - When finished, job returns control to monitor
- Processor point-of-view
  - Processor is first executing instructions from the memory where the monitor resides
  - During this execution, a job may be loaded and the processor will execute the user program
    - “control is passed to a job”: processor is fetching and executing instructions in a user program
    - “control is returned to the monitor”: processor is fetching and executing instructions from the monitor program

Simple Batch Processing

- Fundamental Observations
  - User programs can be faulty:
    - Endangers the whole computer system
    - May overwrite the memory area where the monitor/operating system resides
    - Job is not returning control to monitor (e.g. Running in an endless loop)
  - Separation of concerns:
    - Many user programs will perform similar activities:
      - Provide a library of subroutines that implements functions needed by all programs, e.g. I/O operations etc.
- These are problems that still exist and which influence the architecture of operating systems

Desirable Hardware Features

- Memory protection for monitor
  - User program is not allowed to address the memory area of the monitor
- Privileged Instructions
  - Can only be executed by the monitor
- Timer
  - Set time limits for activities, prevents a job from monopolising a system
- Interrupts
  - Gives OS more flexibility in controlling user programs

Concepts

- Memory protection
  - While a user program is executing, it must not alter the memory containing the operating system
  - Solution
    - Separation of memory in operating system and user-specific areas
    - Processor hardware detects such an error and aborts a job
- Privileged Instructions
  - Certain instructions only the operating system is allowed to execute
    - E.g.: I/O instructions – a user program must relinquish control to the operating system
    - Processor hardware detects such an error and aborts a job
- In modern operating systems, we distinguish between “modes of operation”:
  - “user mode”: certain areas of memory and instructions are protected
  - “kernel mode”: operating system functions, allows access to protected areas of memory and the execution of reserved instructions

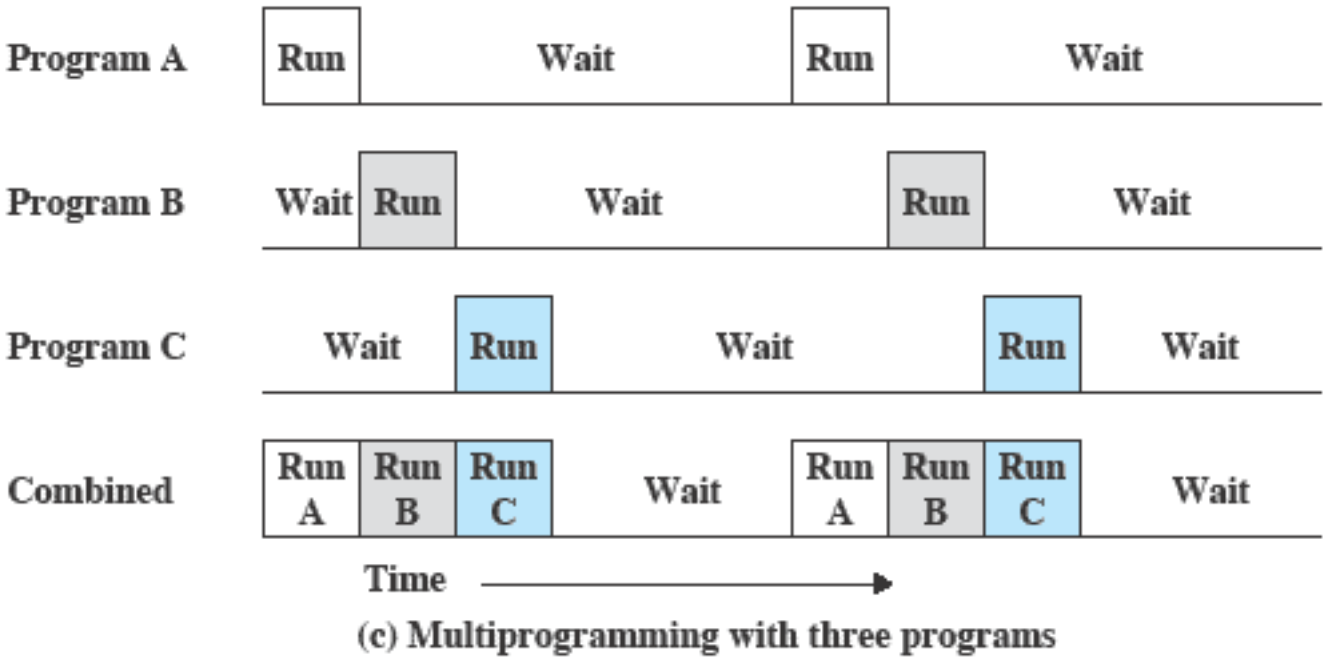
Modes of Operation Protection

- User Mode
  - User programs execute in user mode
  - Certain areas are protected from user access
  - Certain instructions may not be executed
- Kernel Mode
  - Monitor executes in “kernel mode”
  - Privileged instructions may be executed
  - Protected areas of memory may be accessed

Batch Processing - Multiprogramming

- Problem:
  - During I/O operations, processor is idle (processor utilisation was usually ca 5%)
- Solution:
  - Load more than one job into memory
  - Switch between jobs, whenever one of the jobs performs I/O operations
- Goal of multiprogramming
  - Maximise processor utilisation

Multiprogramming

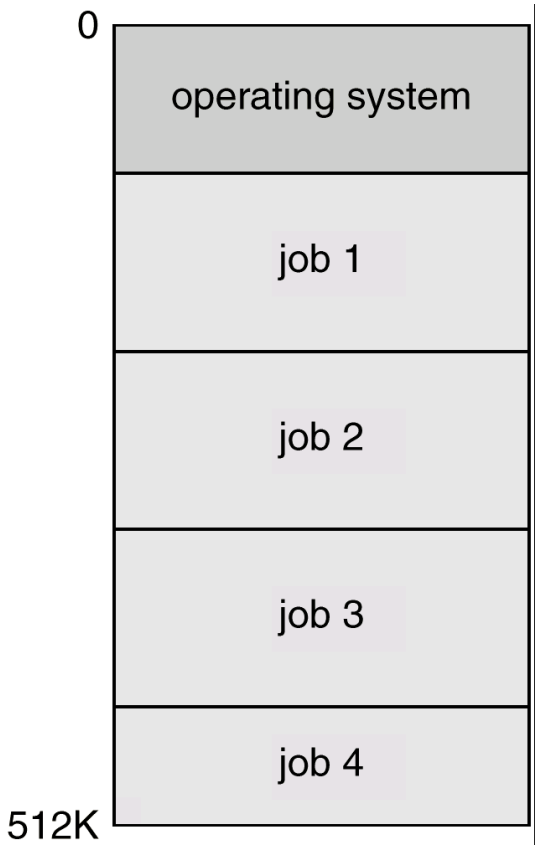


(c) Multiprogramming with three programs

Multiprogramming

Context Switch, Preemption

- Also known as “multitasking”
- Introduces the concept of a *context switch* and *preemption*
  - When one job needs to wait for I/O, the processor can switch to another job (which is not likely waiting for I/O)
  - Jobs are “preempted”: they are interrupted in their current execution
- Multiprogrammed Batch Processing
  - Memory was divided into fixed blocks, holding the monitor and a number of jobs



Concepts

- Multitasking / Concurrent execution of processes
  - Multiprogramming a precursor to this central concept of modern operating systems
- Operating system features
  - Memory management:
    - Multiple jobs held in memory
    - Swapping: storing the current state of a job on disk and restoring state of next job
    - Protection of memory areas
  - Scheduling:
    - Decision, which job to run next
- Hardware features
  - I/O interrupt handling
  - DMA Direct Memory Access

Time-Sharing Systems

Making Computer Systems User-Interactive

- Introduction of interactivity
  - Multiple users simultaneously access the system through terminals
  - They interact with a terminal session / shell that understands commands, allows to start programs
- The OS interleaves the execution of multiple user programs
- Each user program is allocated a short burst or “quantum” of computation
  - With n users online, each user will see ca 1/n effective computer capacity (there is operating system overhead)
  - As human reacting is slow compared to processor speed, such a shared computer’s response time may be very similar to that of a dedicated computer
- Goals
  - Responsiveness:
    - A user wants the computer to respond as fast as possible
    - Time sharing creates the “illusion” that the complete computing resources are available to a user
  - Maximise Processor use
    - Better utilisation allows more user programs to be executed and higher response time

Time Sharing vs Batch Processing

	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

Operating System as a Resource Manager

- Efficient use of limited resources
  - Improving utilisation
  - Minimizing overhead
  - Improving throughput
- Can be achieved:
  - Multi-user / Multiprogramming: multiple programs are executed concurrently

- Allocating resources to applications across space and time
  - Time sharing a resource:
    - Schedule access to resource by different users
  - Space sharing a resource:
    - Allocate memory (or parts of it) to different users

Operating System as a Security Manager

- Protecting applications from each other
  - Enforcing boundaries between programs running on a computer
  - Protect the operating system itself from malfunctioning user programs
- Protecting data
  - Regulate and restrict access to data
  - Determine ownership and access rights of data and processes
  - Execute programs

Major Advances

- Historical developments and solutions to shortcomings resulted in the following main concepts:
  - Processes
    - Implementing the concepts of multiprogramming / multitasking, context switching and preemption
  - Memory management
    - Caching, virtual memory, protection and isolation of tasks
  - Security
    - Identifying users, data protection
  - Scheduling and resource management
    - Fair allocation of processor time to tasks
  - System structure
    - Layered approach to operating system design, separation of user programs and kernel structures

Processes

- Fundamental concept of operating systems
- A process is a program in execution
  - Program code
  - Associated data needed by the program (static variables, stack, heap, buffers etc.)
  - Execution context (process state)
- Execution context is essential for managing processes
  - Is the data structure used by the operating system to control a process
  - Records processor registers at context switch
  - Records process priority and other state information

Memory Management

- An operating system has five principal storage management responsibilities
  - Process isolation
  - Automated allocation and management
  - Support for modular programming
  - Protection and access control
  - Long-term storage

Scheduling and Resource Management

- Operating system manages and allocates processor and memory resources
- Resource allocation policies must consider
  - Efficiency: maximize throughput
  - Fairness: all processes are served in a fair manner
  - Differential responsiveness: processes may be have different priorities and different service requirements

Information Protection and Security

- Access to computer systems and data must be controlled
- Main issues
  - Availability: protect system against interruption
  - Confidentiality: prevent unauthorized access to data
  - Data integrity: prevent unauthorized modification
  - Authenticity: verify identity of users and their credentials, verify validity of transmitted messages and data