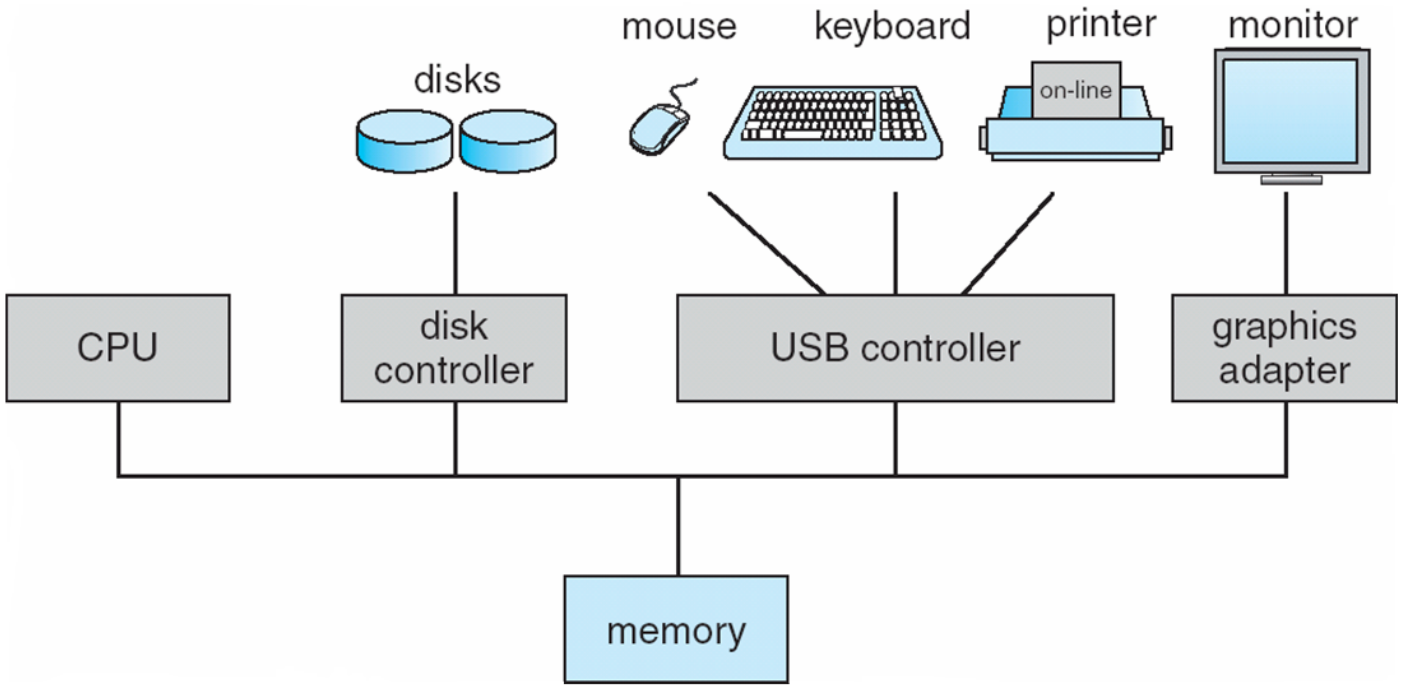
Introduction

CS3026 Operating Systems

Lecture 01

Computer System

- One or more CPUs
- Device controllers (I/O modules)
- Memory
- Bus
- Operating system ?

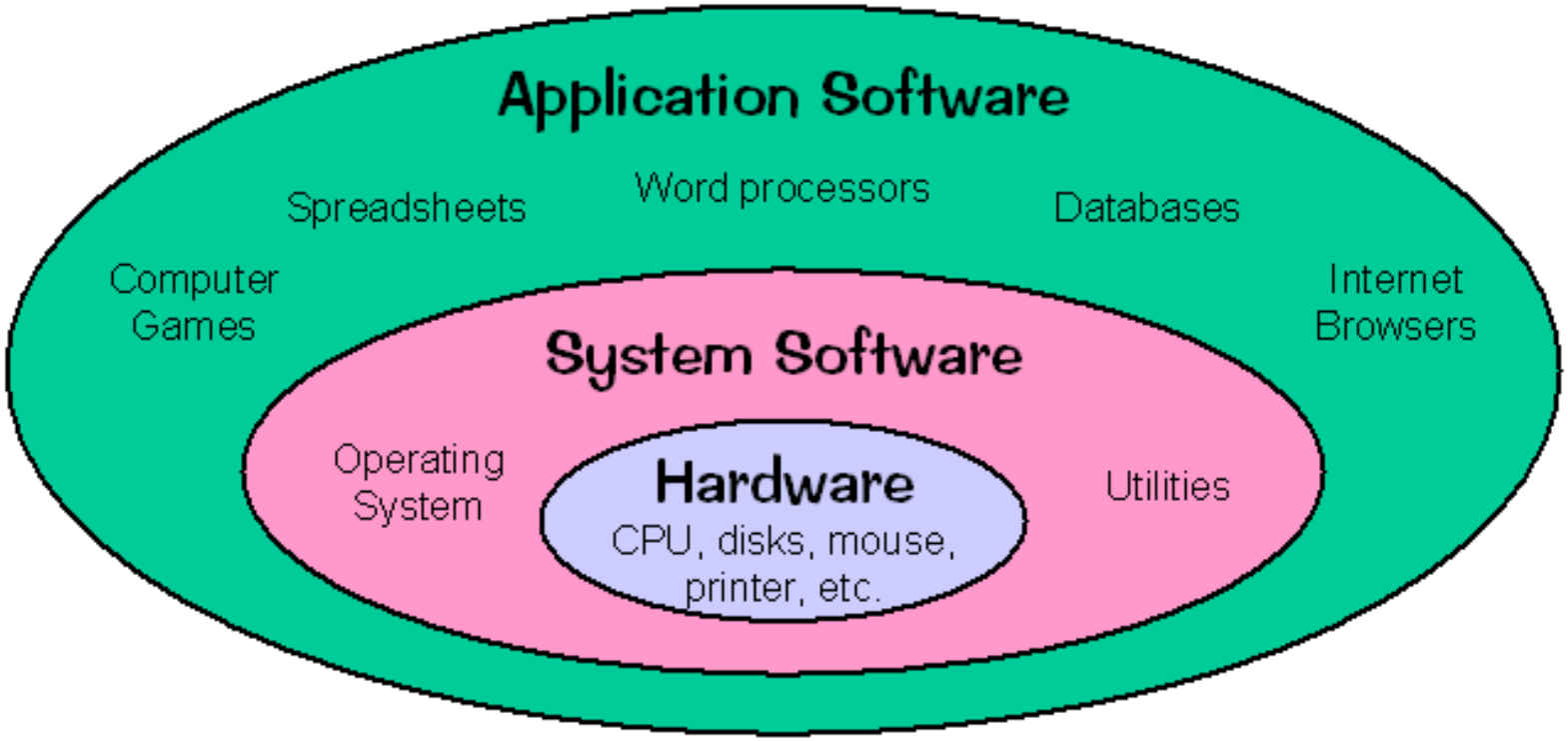


What is an Operating System

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

Layers of a Computer System



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)

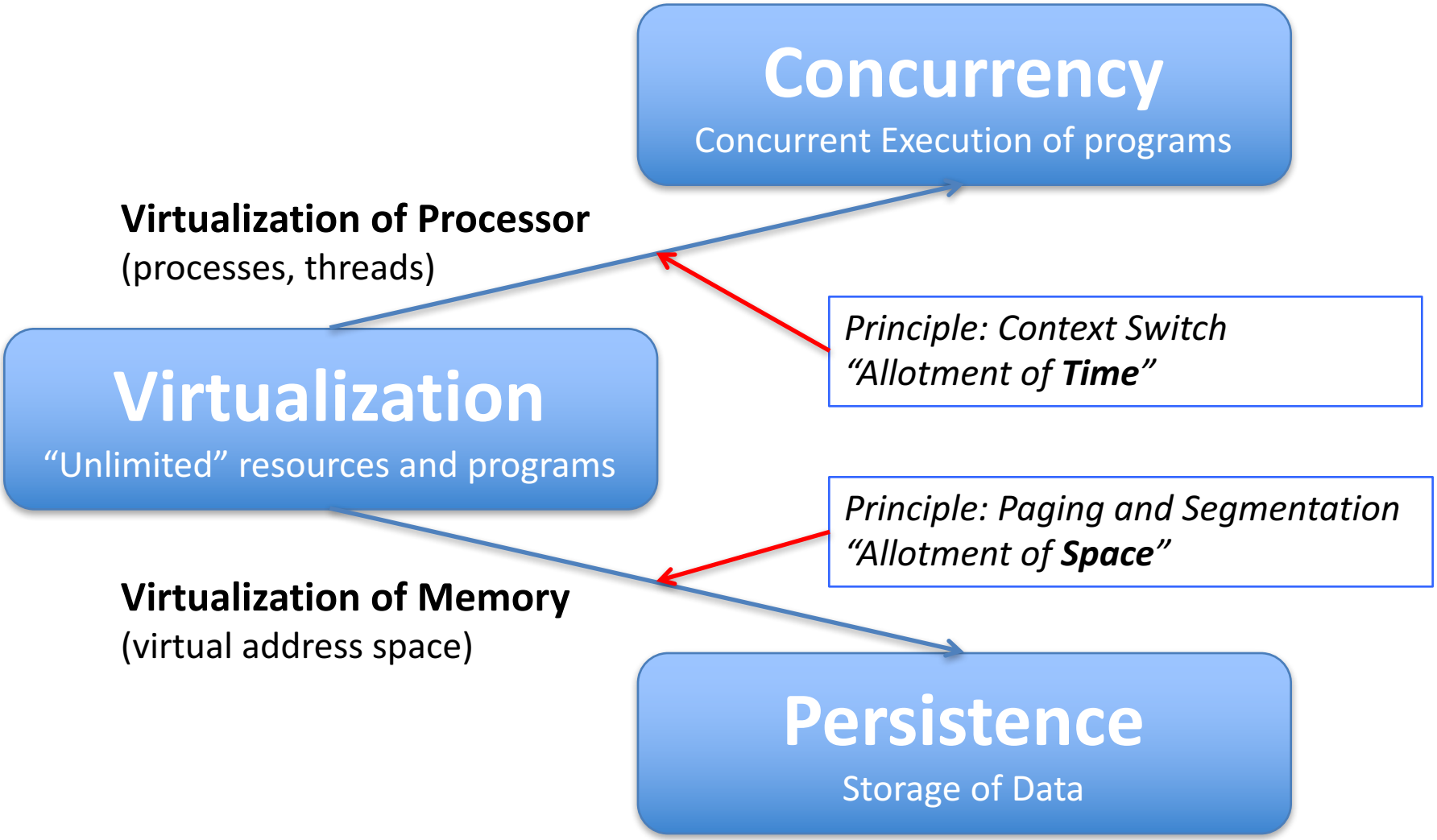
Execution Environment for Programs

- Operating system has to provide an environment for executing programs
  - Programs need resources for execution
    - Processor for executing program instructions
    - Computer memory to hold programs and data
    - I/O devices to communicate with the outside world
  - Operating system provides a uniform abstract representation of resources that can be requested and accessed by applications
    - Programmatic API: system calls for memory access, starting processes, accessing I/O devices (disk, network)

The User Experience

- Modern Operating Systems
  - Multiple programs may execute “at the same time”
    - Virtual processing, context switch, scheduling, concurrency
  - Programs execute in their own save environment
    - Virtual processing, virtual memory management, context switch
  - Multiple users may use a computer system “at the same time”
    - Virtual computer system, multi-user ability
  - Programs may be much larger than actual physical memory
    - Virtualization of memory, persistent storage, paging, swapping
- Objectives
  - Make a computing system convenient to use
  - Make a computer system efficient to use
  - Make a computing system secure to use

Core Concepts



Core Concepts

- Virtualization
  - Virtual processing: multiple programs may execute due to process management
  - Virtual memory: programs may be larger than physical memory due to virtualization of memory resources
- Concurrency
  - Multiple programs may run concurrently and access shared data
  - This leads to problems we have to solve!
- Persistence
  - Data can be stored persistently
  - OS has to protect data against corruption (recovery from system failure) and unauthorized access

The Evolution of Computer Systems

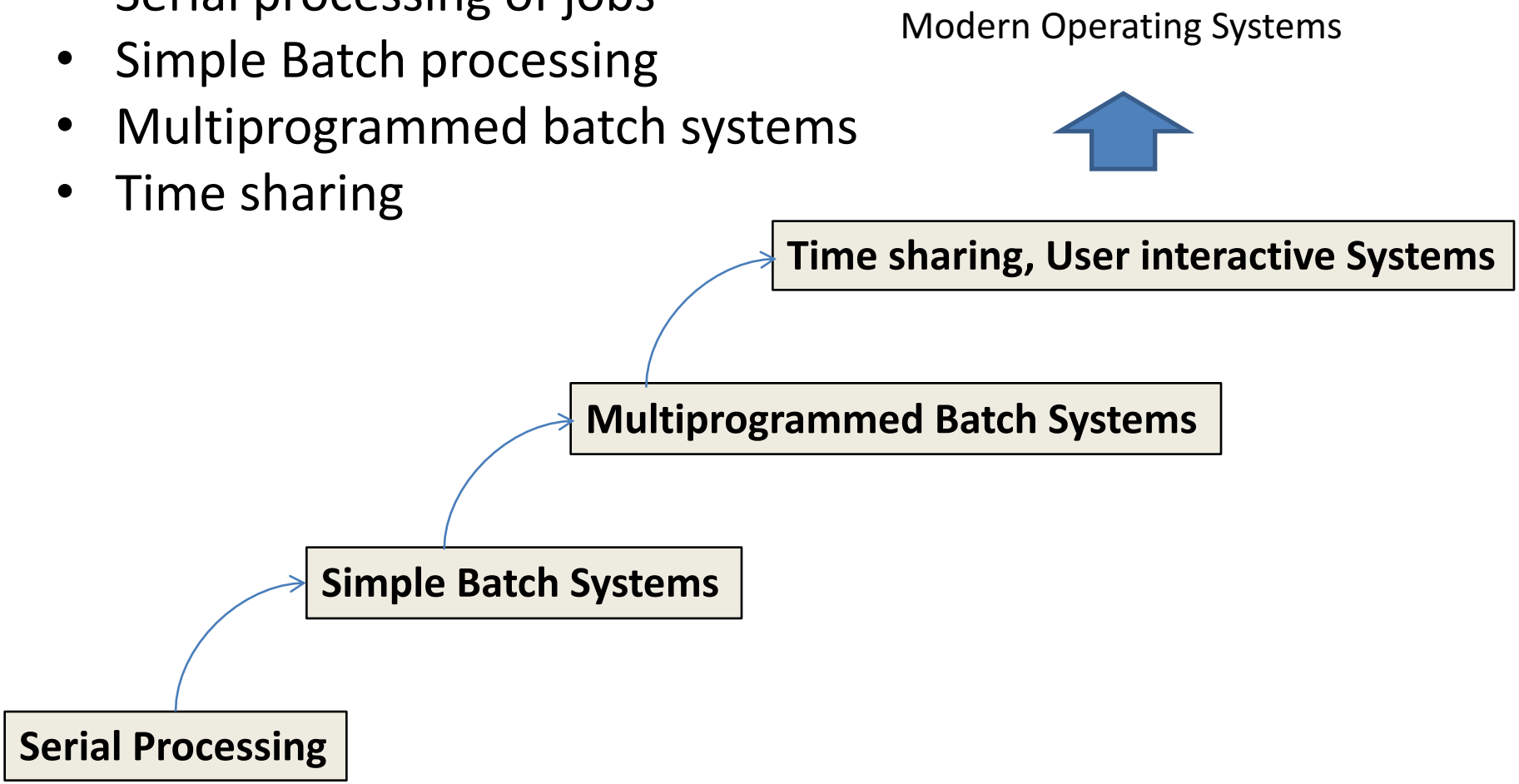
From Single-Program / Single User

To

Multi-program / Multi-user

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

- Automatic execution of a sequence of “jobs”
  - user program, additional programs such as compilers, data to be processed
- Resident Monitor
  - First rudimentary operating system
  - Is a small program that is held permanently in memory
  - Controls the execution of jobs
    - Control of processor is switched between monitor and jobs (user programs)
    - “Job Control Language”: describes how to execute a job, is interpreted by monitor

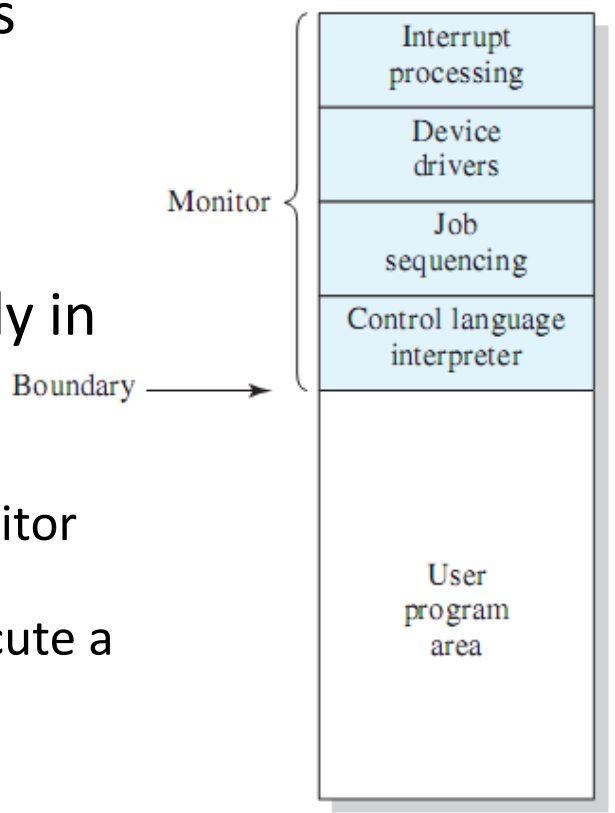


Figure 2.3 **Memory Layout for a Resident Monitor**

Simple Batch Processing

- Resident Monitor
  - 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

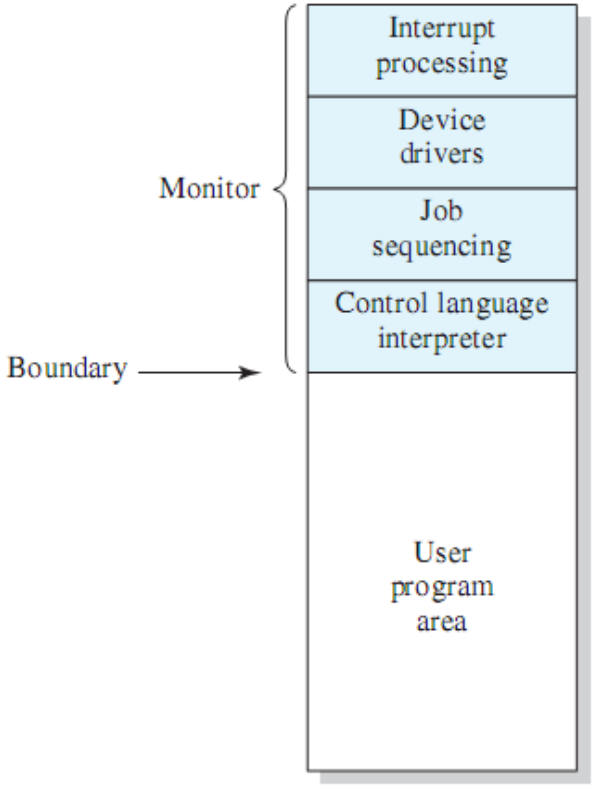
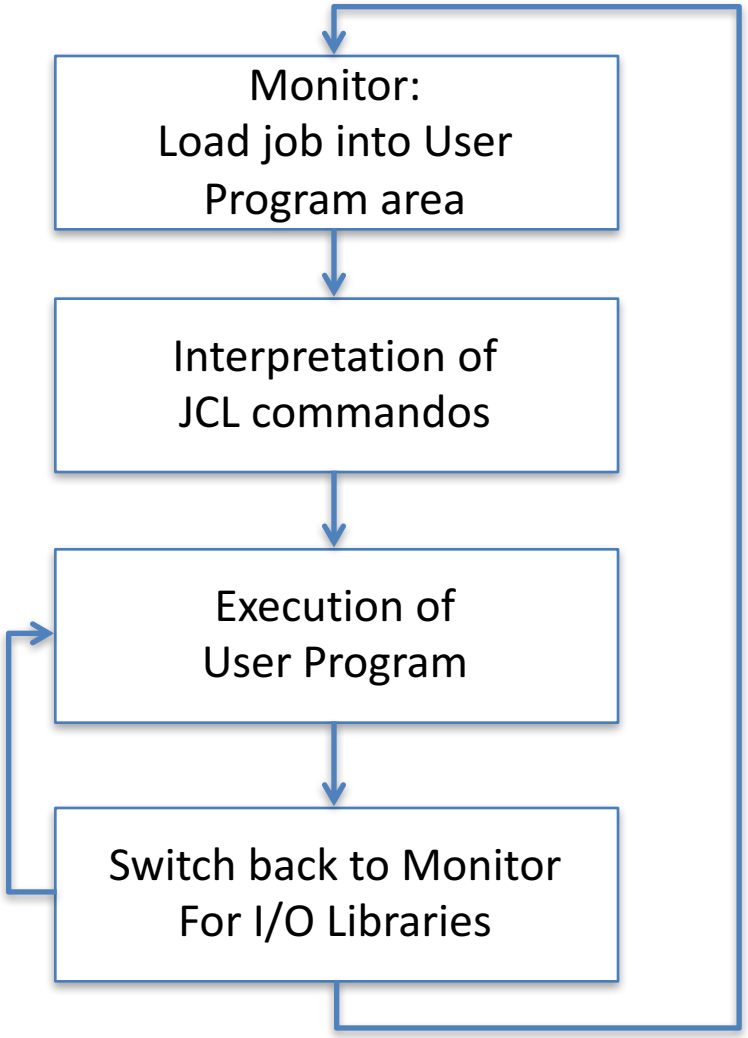


Figure 2.3 **Memory Layout for a Resident Monitor**

Monitor – Job Control
Simple Context Switch between Programs



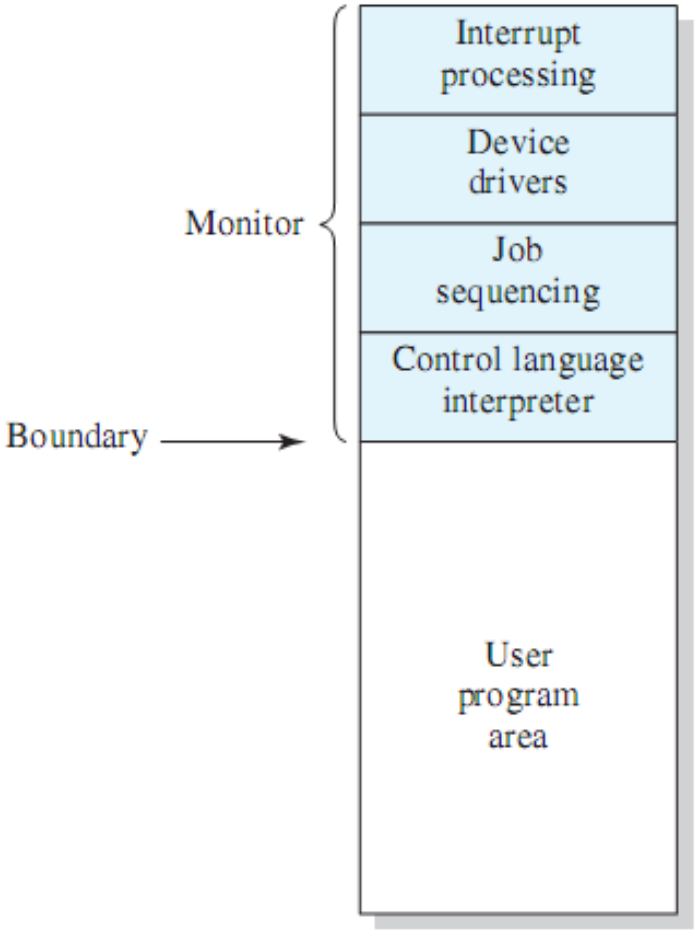


Figure 2.3 **Memory Layout for a Resident Monitor**

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.
      - Perform a switch between a user program and a library subroutine when needed
- These are problems that still exist and influence the architecture of operating systems

Protecting the Operating System Early Hardware Features

- Memory protection for monitor
  - Hardware features for controlling user programs in their behaviour of accessing memory – disallow access to 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
  - CPU reacts to ***interrupts***: interrupts the execution of the current program and executes an interrupt handling routine
  - Interrupts give OS more flexibility in controlling user programs, allows I/O devices to communicate with CPU (interrupts as a communication means between hardware components)

Protecting the Operating System Modern OS Concepts

- Memory protection
  - User program is allowed to alter only specific areas of memory
    - Separation of memory in operating system and user-specific areas
  - 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
  - Hardware detects such an error and aborts a job
- In modern operating systems, this is implemented with "***modes*** of operation":
  - ***user mode***: certain areas of memory and instructions are protected / not accessible for user programs
  - ***kernel mode***: only for operating system functions, allows access to protected areas of memory and the execution of reserved instructions

Protecting the Operating System Modes of Operation

- User Mode
  - User programs execute in ***user mode***
  - Certain areas are protected from user access
  - Certain instructions may not be executed

- Kernel Mode
  - Operating system executes in ***kernel mode***
  - Privileged instructions may be executed
  - Protected areas of memory may be accessed

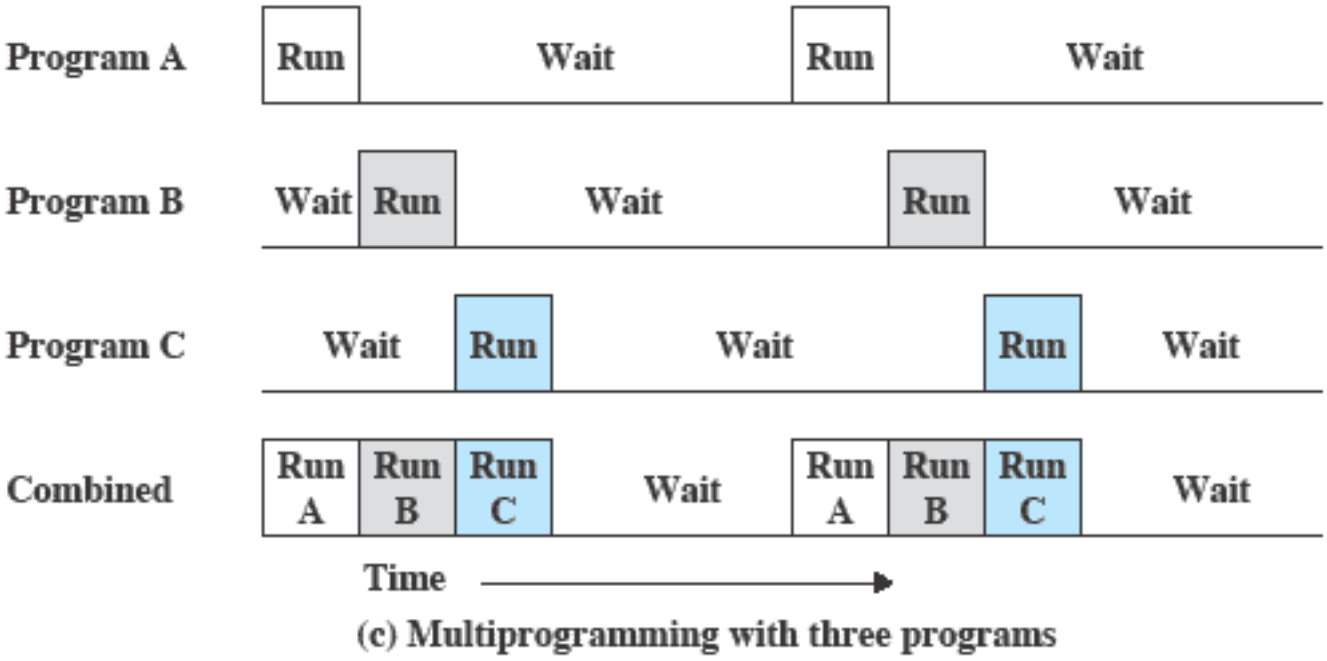
Simple Batch Processing - Legacy

- “Kernel”:
  - Monitor can be regarded as a simple operating system kernel
- “Memory protection”, “User / Kernel mode”
  - User programs are not allowed to access and overwrite memory occupied by monitor
- “System calls”, “Shared libraries”:
  - Monitor may contain program code that may be used as sub-routines by user programs – early form of “system calls”
  - Only small part of monitor has to be resident, additional non-resident parts can be loaded on demand
- Context switch
  - Switch between different programs

From Sequential Batch Processing to 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

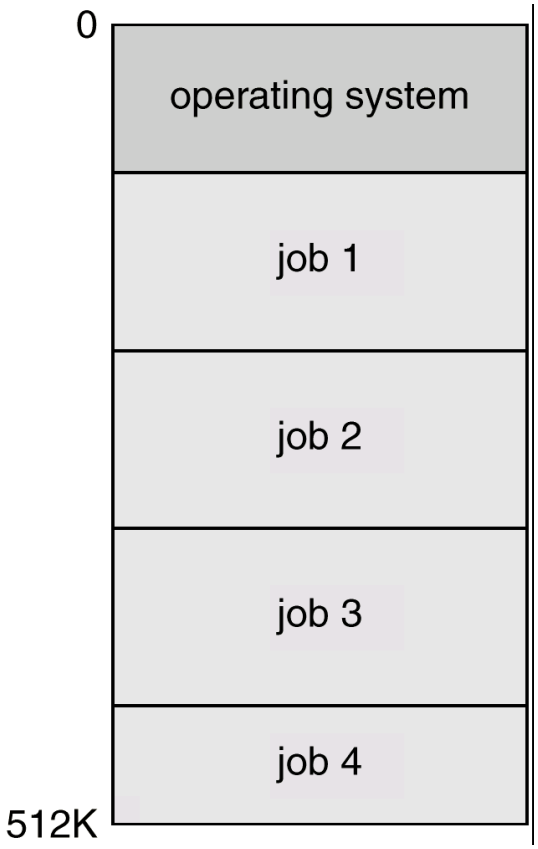
Multiprogramming



(c) Multiprogramming with three programs

Multiprogramming
Context Switch, Preemption

- Also known as “multitasking”
- Introduces the concept of a ***context switch***
  - When one job needs to wait for I/O, the processor can switch to another job (which is not likely waiting for I/O)
- Introduces the concept of ***preemption***
  - 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



Multiprogramming - Legacy

- **Concurrent** execution of processes
  - Multiprogramming a precursor to the central concept of **concurrency** in 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 (I/O devices directly access memory, no processor involvement)

Time-Sharing Systems

Making Computer Systems User-Interactive

- Introduction of interactivity
  - Multiple users simultaneously access the system through terminals
  - They interact via a terminal session / shell that understands commands, allows to start programs
- The OS interleaves the execution of multiple user programs
- 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

- Batch processing
  - Maximise CPU usage
- Time Sharing
  - Minimise response time
- Both interactive and batch processing activities exist in today's operating systems

Multi-user / Time Sharing - Legacy

- Process management and scheduling in modern operating systems
- 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 some operating system overhead)
  - As human reacting is slow compared to processor speed, such a shared computer’s response time may create the illusion that a user interacts with a dedicated computer

Today …

- New developments:
  - Embedded systems
  - Small computing devices, such as Arduino, Beagleboard, etc.

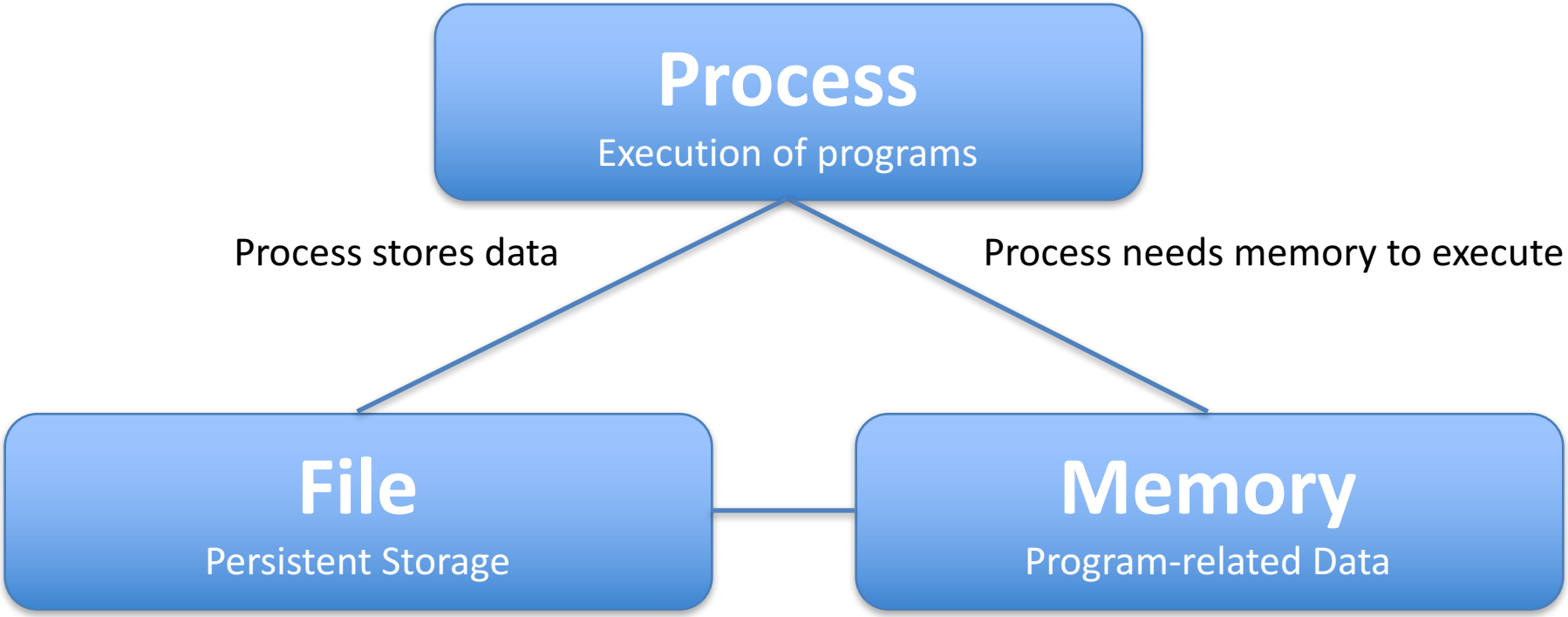
  - History repeats itself:
    - Programming: one user, loads one program onto device, no operating system

Modern Operating Systems

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 executing programs from each other
  - Secure user access to data and define ownership of files and processes

Operating System Objects

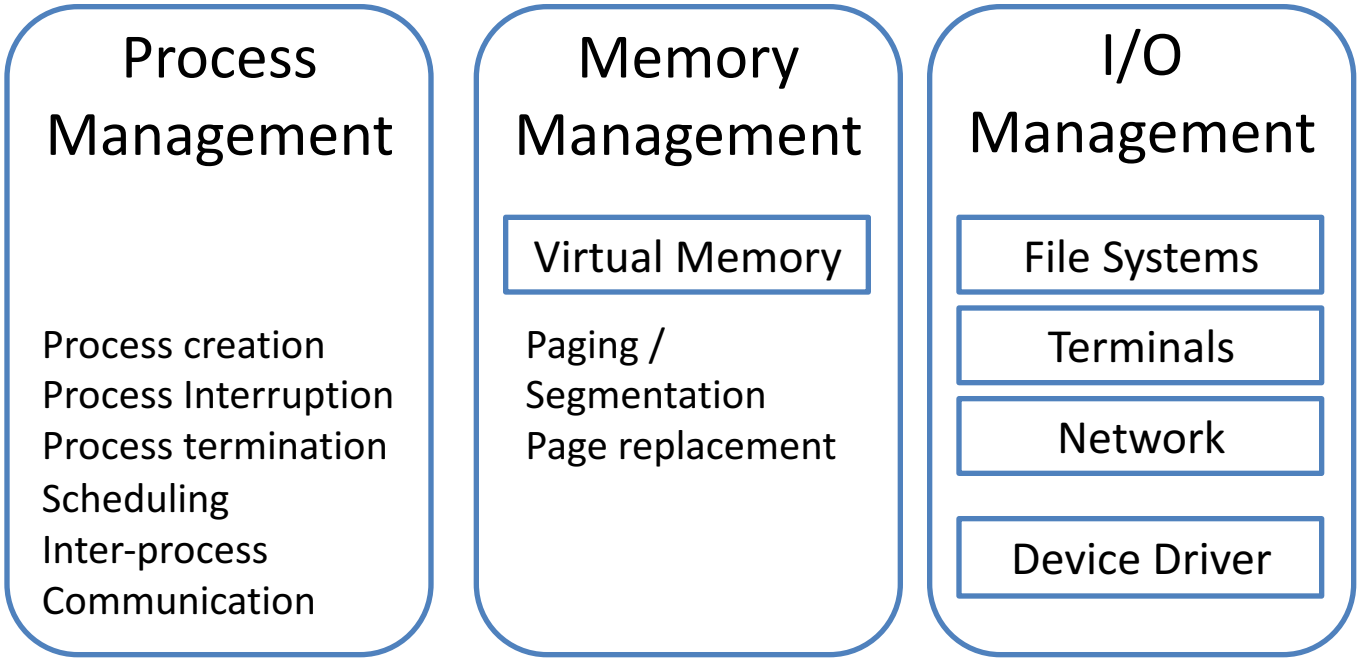


Virtualization:

OS may store part of process-related memory in a file

OS isolates processes from each other – virtual address spaces

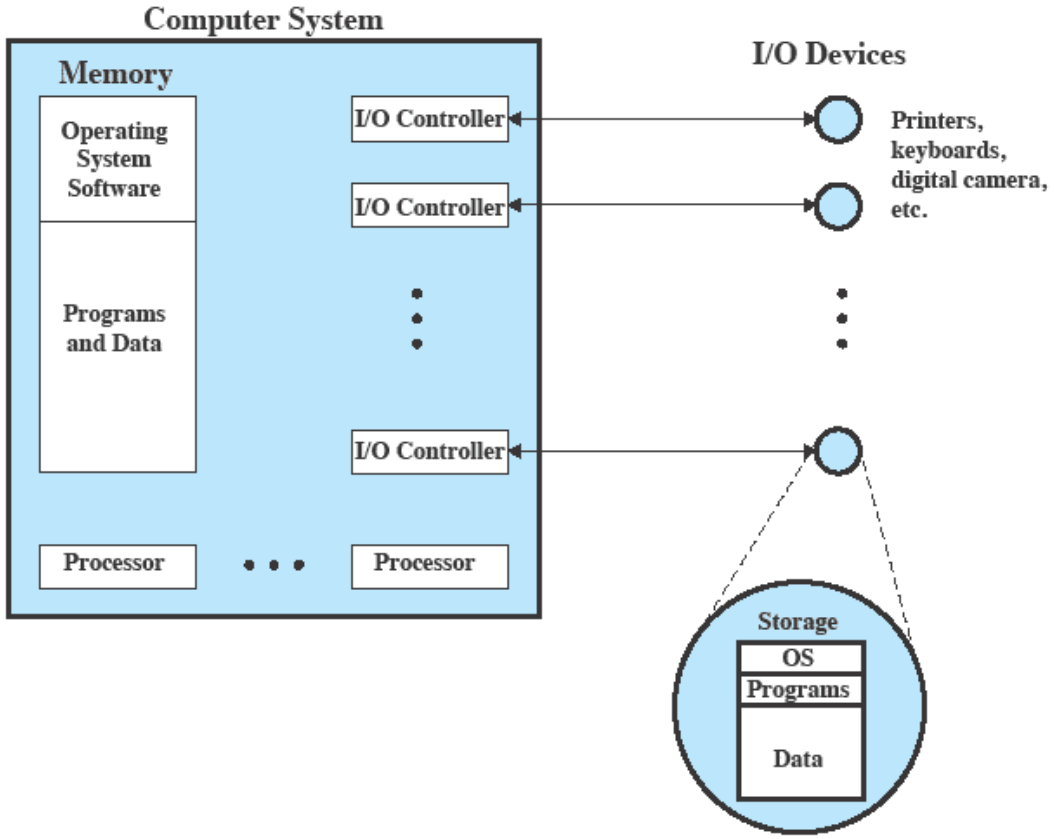
OS Management Tasks



- Process management
- Memory management
- I/O systems management
  - File system management

Operating System as a Resource Manager

- Operating System manages system resources
  - Processor
  - Memory
  - I/O modules and devices



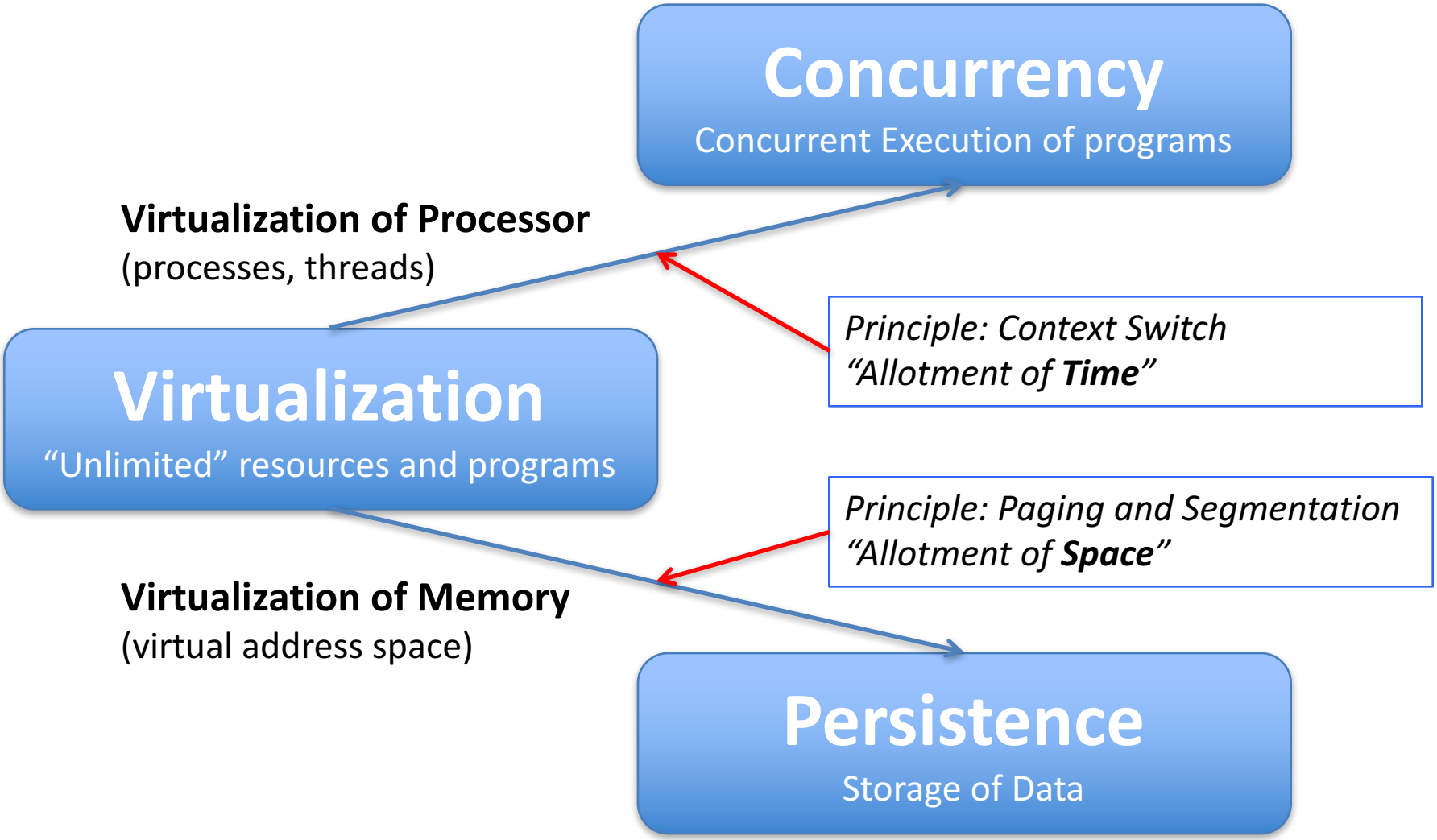
Operating System as a Resource Manager

- Operating system manages and allocates processor and memory resources to programs
- Allocation policies for limited resources
  - Efficiency: maximize throughput, minimize overhead
  - Fairness: all processes are served in a fair manner
  - Differential responsiveness: processes may have different priorities and different service requirements
- Can be achieved:
  - 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
    - Execute programs in a protected environment
  - 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
- 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

Core Concepts

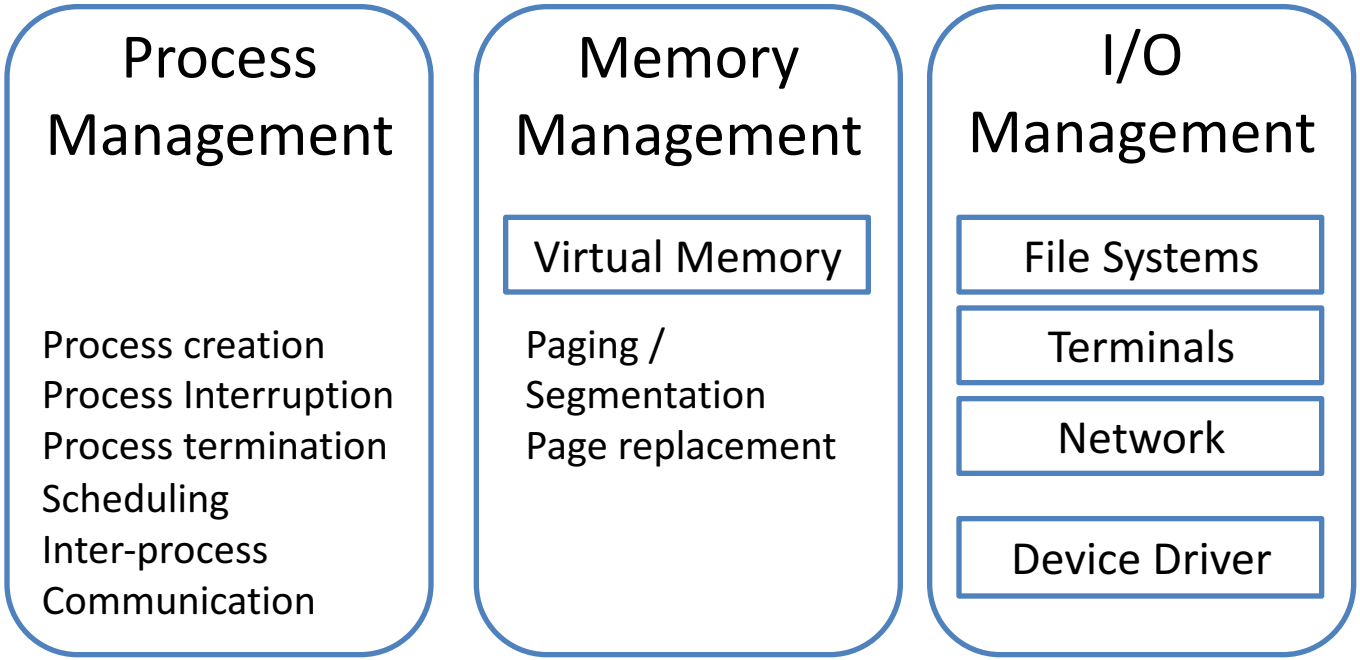


Core Concepts

- Virtualization
  - Virtual processing: multiple programs may execute due to process management
  - Virtual memory: programs may be larger than physical memory due to virtualization of memory resources
- Concurrency
  - Multiple programs may run concurrently and access shared data
  - This leads to problems we have to solve!
- Persistence
  - Data can be stored persistently
  - OS has to protect data against corruption (recovery from system failure) and unauthorized access

OS Management Tasks

OS Management Tasks



- Process management
- Memory management
- I/O systems management
  - File system management

Processes

- Fundamental concept of operating systems
  - Software structures within the OS that represents the execution context of a program
- A process is a program in execution
  - Program code
  - Associated data needed by the program (static variables, stack, heap, buffers etc.)
  - A process records the current state of execution - execution context
    - Current content of processor registers
    - Current progress of execution (instruction pointer)
    - Various CPU flags
  - A process owns resources: CPU time, physical memory, open files, I/O devices (computer screen, keyboard, etc.)
- 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

Process Management Task

- Processor Virtualization
  - a process can be regarded as an abstract virtualized form of the physical processor
  - Many processes may operate “at the same time” (concurrently) and share execution time on a processor
- Responsibilities
  - Process creation, termination
  - Process scheduling, suspension and resumption
  - Process synchronization, inter-process communication

Memory

- Memory is a large array of addressable storage elements or bytes (8-bit)
  - each byte has a unique address
  - The CPU will load a sequence of bytes from a particular address depending on the “word” size of the processor, e.g. on a 64-bit architecture, a word contains 8 bytes
- A process needs memory to load program code and data

Memory Management Task

- Memory virtualization
  - a process operates within a logical address space, large part of it held on hard disk, only small part is mapped onto actual physical memory
  - Replacement algorithms for loading those parts of a process that are currently needed
- Responsibilities
  - Allocate and de-allocate physical memory
  - Efficient utilization of a limited memory resource
  - Keep track of which parts of physical memory are currently allocated to which process

I/O System Management Task

- Interaction with the physical world
  - Console / terminal, keyboard, mouse, display
  - Persistent storage
  - Networking
- Responsibilities
  - Buffer, caching
  - Device drivers

File Management Task

- Files as collections of information stored persistently
  - Files have a name that is recorded in a directory and occupy disk space
- Files as abstract data streams
  - OS regards a file as a data stream and provides operations for user programs to open, read, write and close these data streams
  - These can be physical files stored on disk, but also network connections (sockets), inter-process communication such as pipes, or abstract representations of physical devices that deliver data streams (see the /dev directory on a Unix/Linux system)
- Responsibility of File Management
  - Manipulation of files and directories
  - Management of file systems
- Responsibility of Storage Management
  - Allocation of storage space
  - Disk scheduling