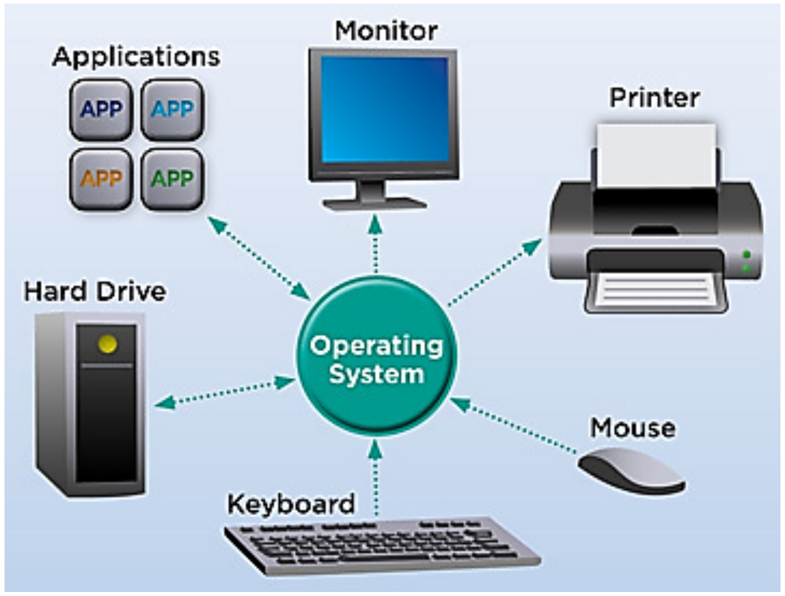
Introduction to Operating Systems

What is an Operating System?

- **Definition**: An OS is a program that acts as an intermediary between a user of a computer and the computer hardware.
- **Goals**:
    - Execute user programs
    - Make the computer system easy to use
    - Utilize hardware efficiently



Evolution and History of Operating Systems

- **Early Days:** Computers had no operating systems; each program required different code, making data processing complex and time-consuming.
- **1956:** General Motors developed the first OS for a single IBM computer.
- **1960s:** IBM began installing OS in their devices.
- **UNIX:** First version launched in the 1960s, written in C.
- **Microsoft:** Developed OS at IBM's request.
- **Today:** All major computer devices have OS, performing similar functions with unique features.

Common Operating Systems

Microsoft
Apple
Mac™ OS
Microsoft Windows Server
Linux™
MS DOS
ubuntu

Fundamental Goals of an Operating System

1

Efficient Use: Optimize computer resource utilization

2

User Convenience: Provide easy and convenient system usage

3

Non-Interference: Prevent user activity interference

Advantages and Disadvantages of Operating Systems

- **Advantages:**

✓**Memory Management:** Manages data in the device efficiently.

✓**Hardware Utilization:** Optimizes use of computer hardware.

✓**Security:** Maintains device security.

✓**Application Efficiency:** Helps run different applications smoothly.

- **Disadvantages:**

✓**Usability:** This can be difficult for some users.

✓**Cost and Maintenance:** Some are expensive and need heavy maintenance.

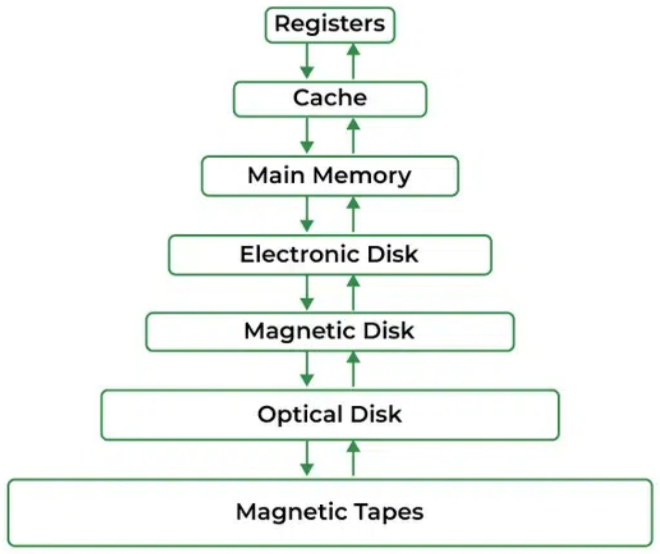
✓**Security Risks:** Vulnerable to hacking threats.

Computer System Structure

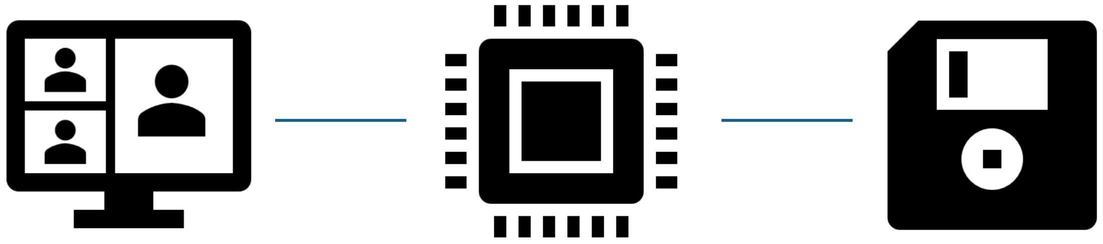
- **Hardware**: Physical components
- **OS**: Manages hardware, provides services for applications
- **Applications**: Software performing specific tasks
- **Users**: People interacting with the computer

Functions of an OS

- **Resource Allocator**: Decides between conflicting requests for efficient and fair resource use.
- **Control Program**: Controls execution of programs to prevent errors and improper use of the computer.



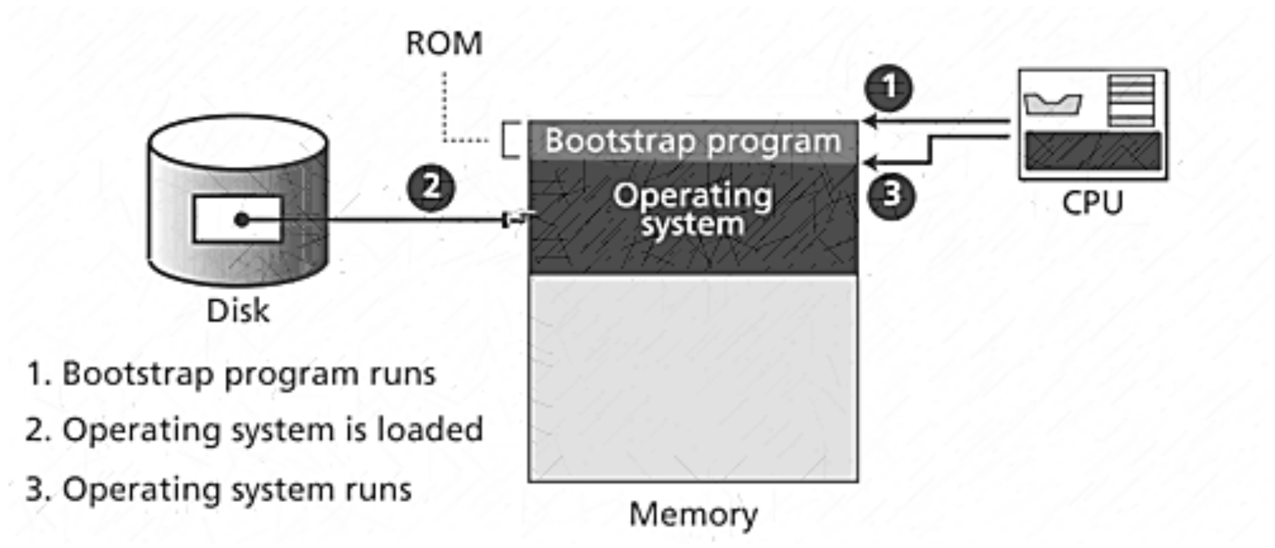
The Kernel



- **Definition**: The one program running at all times on the computer.
- **Role**: Manages system resources and communication between hardware and software.

Bootstrap Program

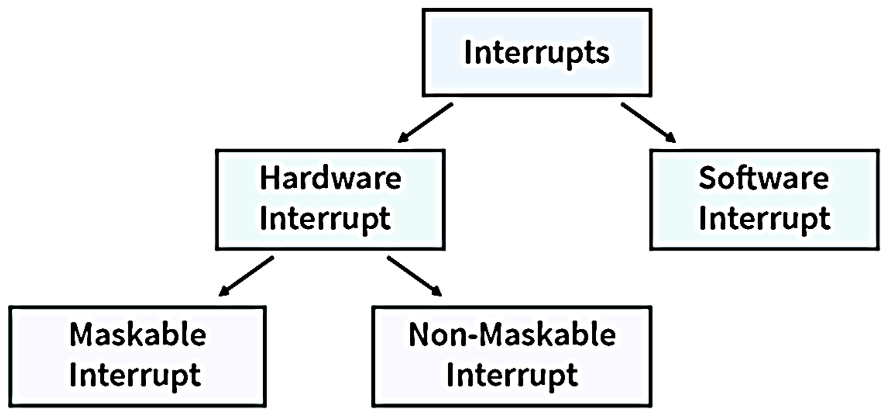
- **Definition**: Loaded at power-up or reboot.
- **Storage**: Stored in ROM or EPROM (firmware).
- **Function**: Initializes system aspects, loads OS kernel, and starts execution.



Concurrency and Interrupts

- **Concurrency**: I/O and CPU can execute concurrently. Example: A user can type on a keyboard while a file is being downloaded.
- **Interrupts**:
  - Device controllers inform CPU of operation completion via an interrupt.
  - Control is transferred to the interrupt service routine through the interrupt vector.
  - Example: A printer controller signals the CPU that printing is complete.

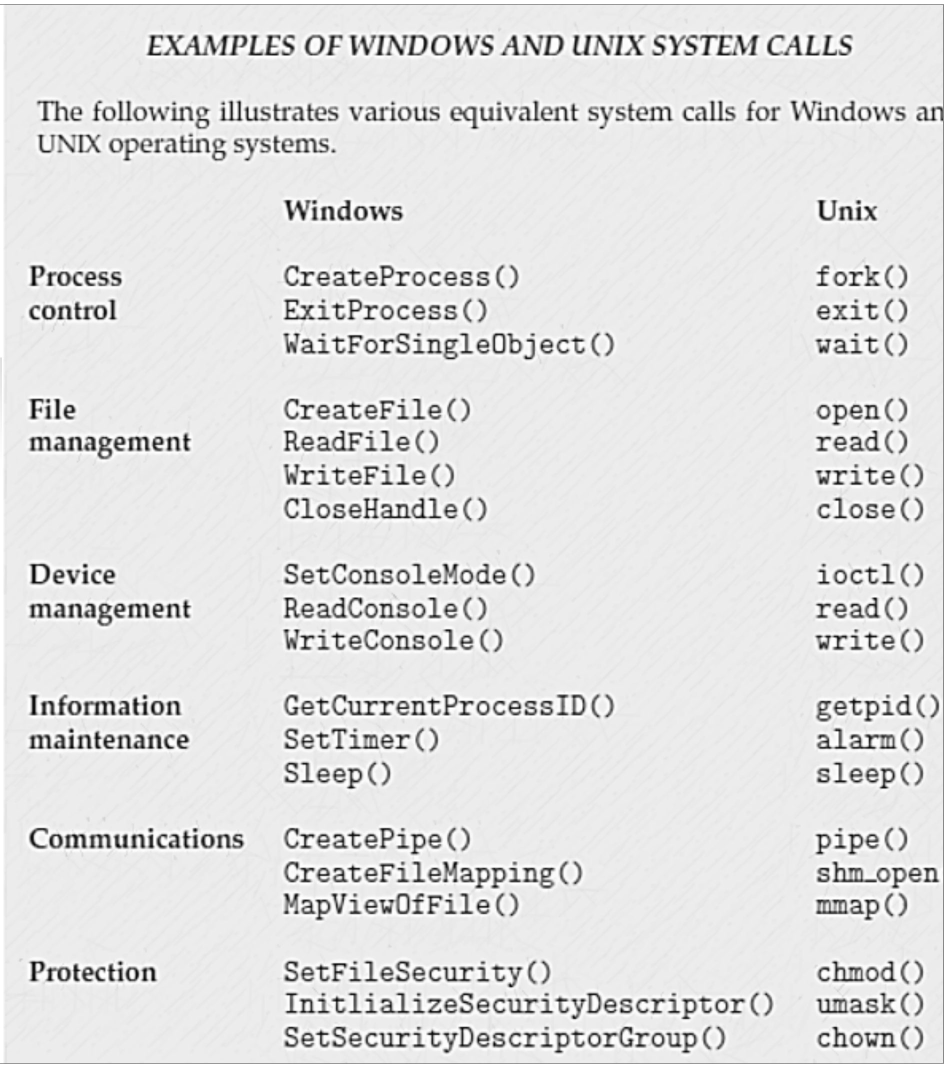
Types of Interrupts



- **Interrupts**: Incoming interrupts are disabled while another is processed.
- **Trap**: A software-generated interrupt caused by an error or user request.
- **OS Role**: Determines interrupt type via polling or vectored interrupt system.

System Calls and Device-Status Table

- **System Call**: Request to the OS for I/O completion.
- **Device-Status Table**: Entry for each I/O device indicating its type, address, and state.



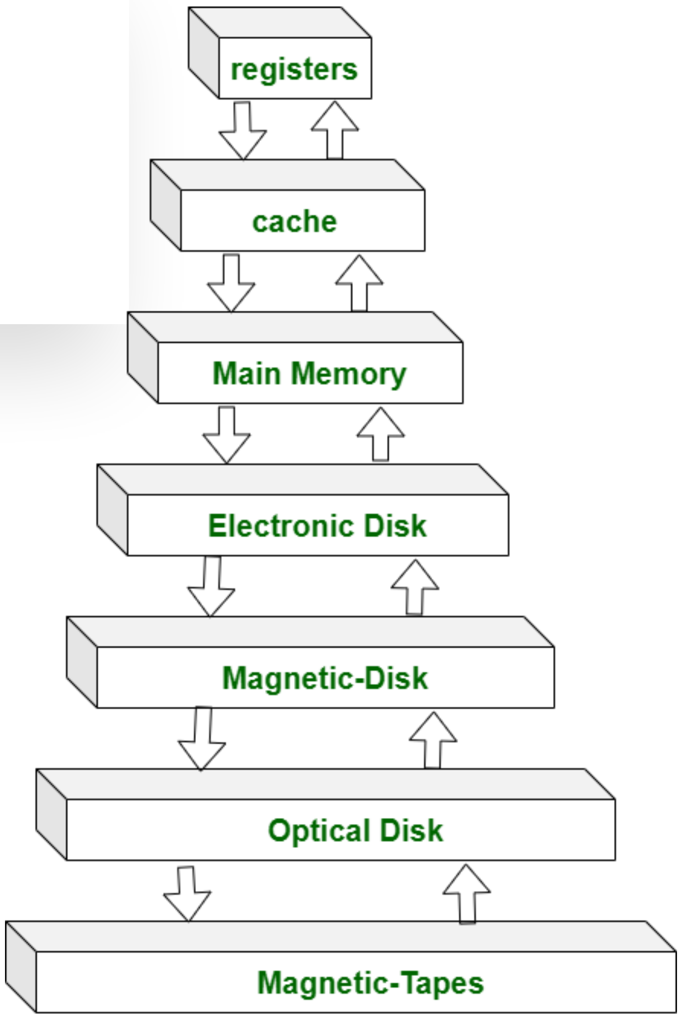
EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows an UNIX operating systems.

	Windows	Unix
Process control	`CreateProcess()`	`fork()`
	`ExitProcess()`	`exit()`
	`WaitForSingleObject()`	`wait()`
File management	`CreateFile()`	`open()`
	`ReadFile()`	`read()`
	`WriteFile()`	`write()`
	`CloseHandle()`	`close()`
Device management	`SetConsoleMode()`	`ioctl()`
	`ReadConsole()`	`read()`
	`WriteConsole()`	`write()`
Information maintenance	`GetCurrentProcessID()`	`getpid()`
	`SetTimer()`	`alarm()`
	`Sleep()`	`sleep()`
Communications	`CreatePipe()`	`pipe()`
	`CreateFileMapping()`	`shm_open`
	`MapViewOfFile()`	`mmap()`
Protection	`SetFileSecurity()`	`chmod()`
	`InitlializeSecurityDescriptor()`	`umask()`
	`SetSecurityDescriptorGroup()`	`chown()`

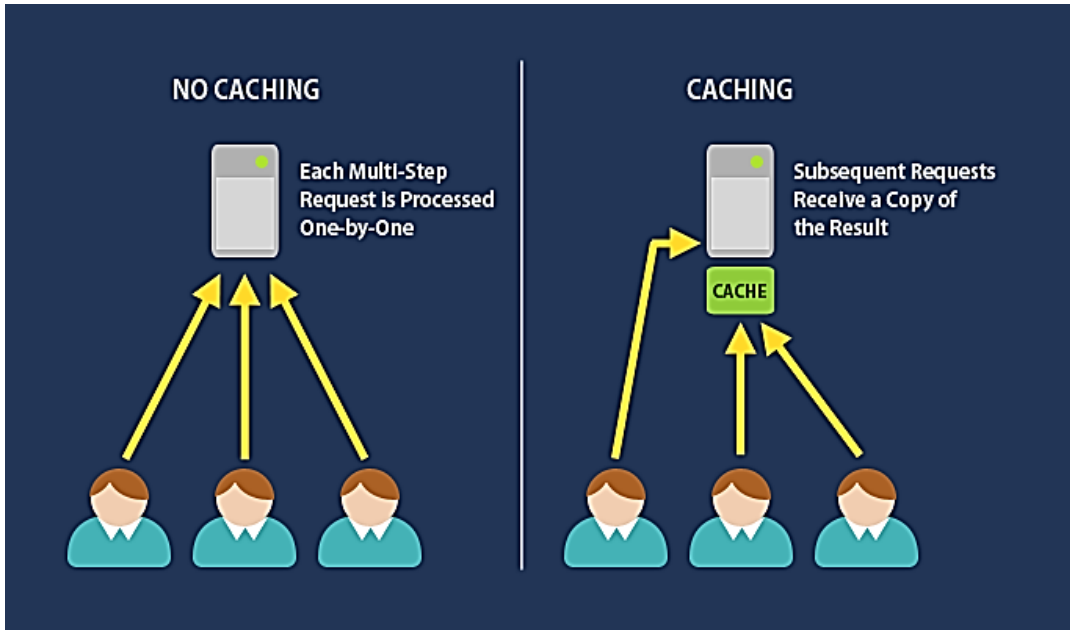
Storage Structure

- **Main Memory**: Random access, volatile. Example: RAM, used for currently running programs.
- **Secondary Storage**: Large, non-volatile extension of main memory. Example: Hard drives, SSDs, for long-term data storage.
- **Disk**: Divided into tracks and sectors, managed by disk controller. Example: A hard disk is divided into tracks and sectors.



Caching

Copying information into faster storage systems to speed up access.

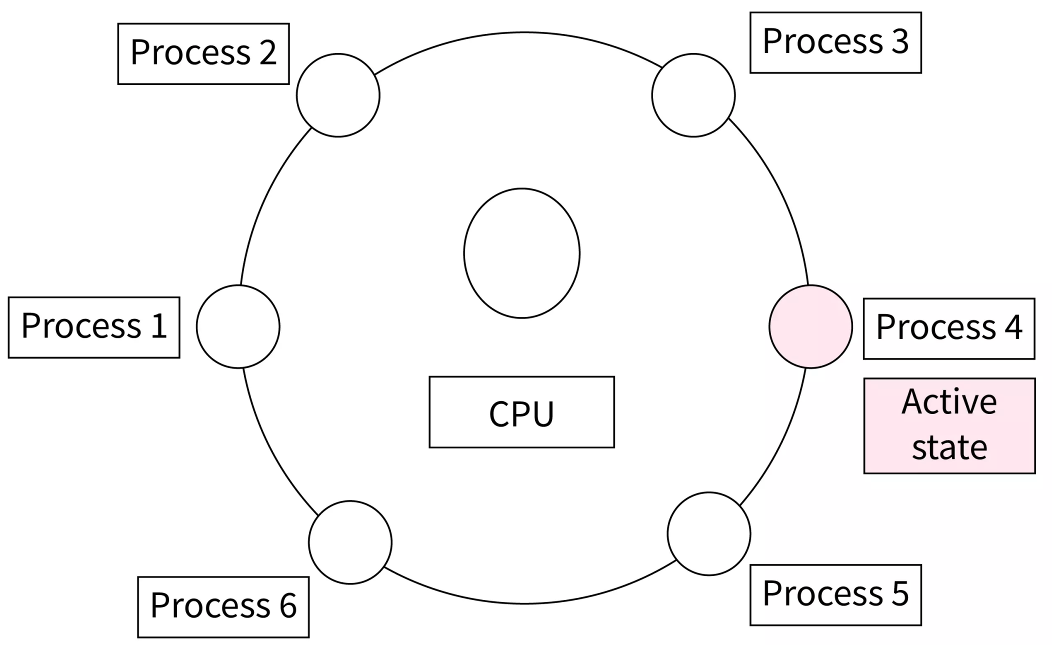

NO CACHING
Each Multi-Step Request Is Processed One-by-One
CACHING
Subsequent Requests Receive a Copy of the Result
CACHE

Multiprocessor Systems

- **Benefits**: Increased throughput, economy of scale, increased reliability.
- **Types**:
    - Asymmetric
    - Symmetric
    - Clustered systems: Linked multiprocessor systems.

Multiprogramming and Timesharing

- **Multiprogramming**: Provides efficiency via job scheduling.
- **Timesharing**: CPU switches jobs frequently for interactive computing.

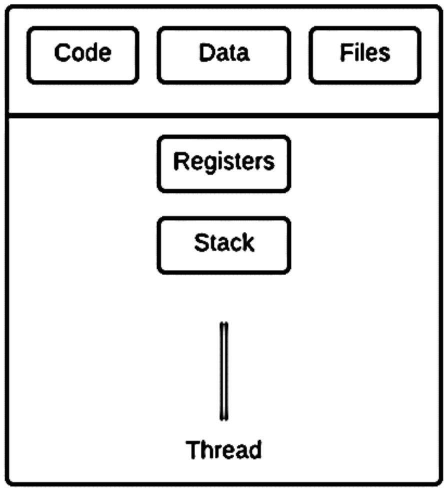


Dual-Mode Operation

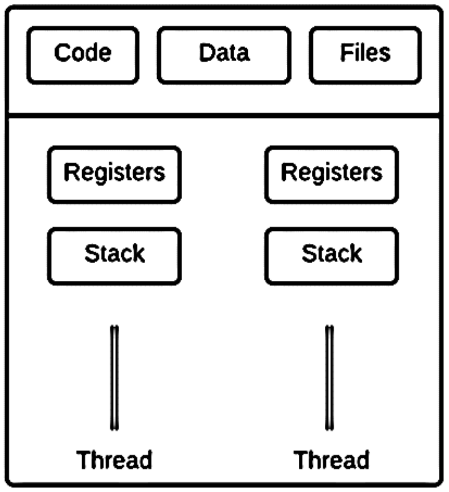
- **Definition**: Allows OS to protect itself and other system components.
- **Modes**:
  - User Mode
  - Kernel Mode
- **Privileged Instructions**: Executable only in kernel mode.

Processes and Threads

- **Single-threaded Processes**: One program counter.
- **Multi-threaded Processes**: One program counter per thread.



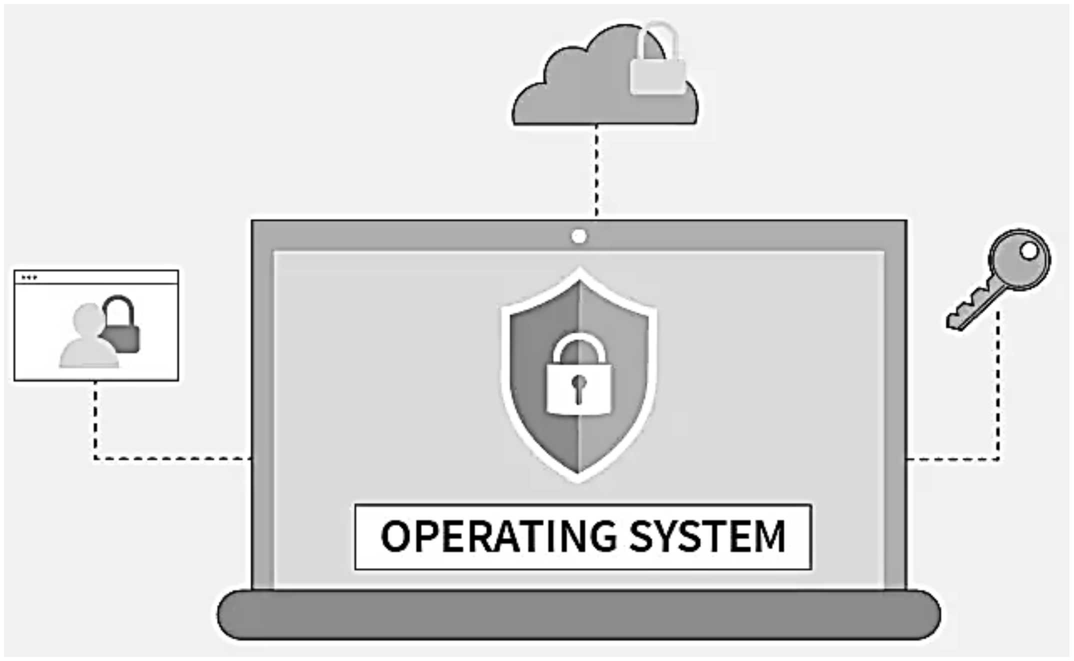
Single-threaded Process



Multi-threaded Process

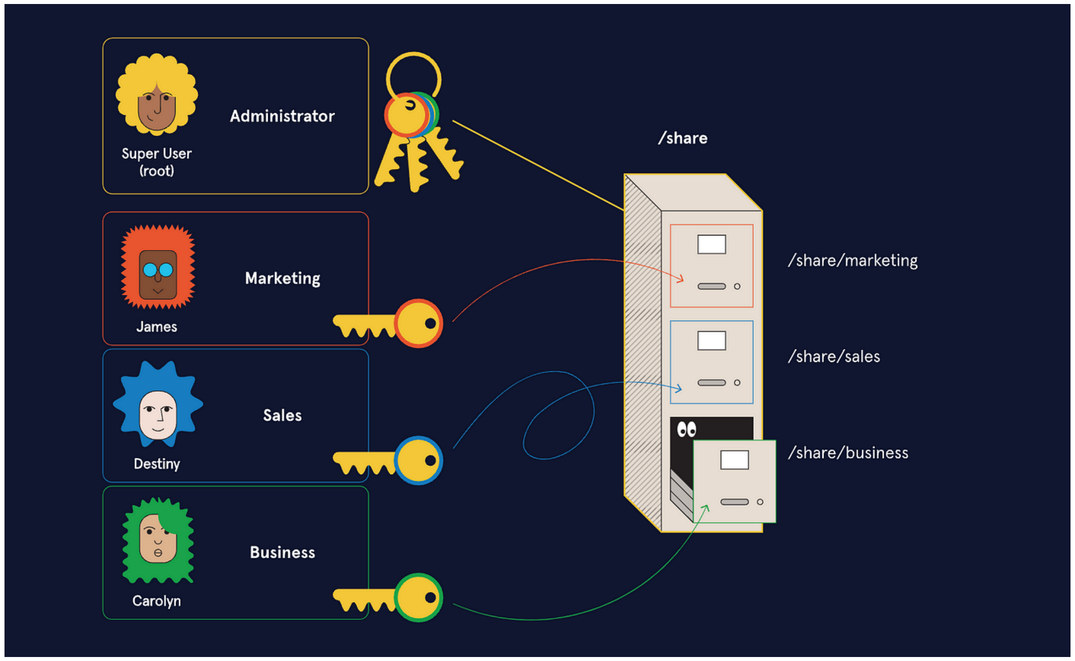
Protection and Security

- **Protection**: Mechanism for controlling access to resources.
- **Security**: Defense against attacks.



User and Group IDs

- **User IDs (UID)**: One per user.
- **Group IDs**: Determine privileges for users and groups.



Activity

- **Choose an operating system such as Windows, macOS, or Linux.**
- **Answer these questions about the operating system you chose:**

1. What are the key features?
2. How does the user interface look and feel?
3. What security features are included?
4. What are the common use cases?

Sample Answer

- macOS is known for its sleek and intuitive user interface, featuring the Dock and Mission Control for easy navigation. Key features include seamless integration with other Apple devices, built-in applications like Safari and Photos, and robust performance. Security features include FileVault for disk encryption, Gatekeeper for app security, and regular software updates. Common use cases for macOS include graphic design, video editing, and general productivity due to its stable performance and user-friendly interface.