Quiz 1 Compressed

1. Operating System (OS): Definition and Purpose

- **OS:** Software acting as an intermediary between user and hardware.
- **Goals:**
  - Execute user programs efficiently.
  - Make the system user-friendly.
  - Utilize hardware resources effectively.

2. Components of a Computer System

- **Hardware:** CPU, memory, I/O devices.
- **Operating System:** Manages hardware resources.
- **Application Programs:** Solve user problems (e.g., browsers, editors).
- **Users:** People or other systems.

3. What Operating Systems Do

- **User's View:** Provide convenience, ease of use, and performance.
- **System's View:** Manage resources efficiently and fairly.

4. Common OS Functions

- **Resource Allocator:** Manages and allocates hardware resources.
- **Control Program:** Controls execution to prevent errors.

5. Interrupts

- **Definition:** Signals indicating events.
- **Types:**
  - **Hardware Interrupts:** From hardware via system bus.
  - **Software Interrupts (Traps):** From software errors or requests.
- **Handling:** Control transferred to interrupt service routine; interrupts disabled during processing.

6. Memory and Storage Management

- **Memory Management:**
  - Keep track of memory usage.
  - Move processes/data in/out of memory.
  - Allocate/deallocate memory space.
- **Storage Management:**
  - **File System:** Create/delete files/directories, manage access, perform backups.
  - **Mass Storage:** Manage free space, storage allocation, disk scheduling.

7. Process Management

- **Process:** A program in execution.
- **Activities:**
  - Create/delete processes.
  - Suspend/resume processes.
  - Synchronize and communicate between processes.
  - Handle deadlocks.

8. Caching

- **Definition:** Store data temporarily from slower to faster storage.
- **Goal:** Improve performance through efficient cache use.

9. Protection and Security

- **Protection:** Control access to system resources.
- **Security:** Defend against external threats (viruses, DoS attacks).
- **User IDs & Privilege Escalation:** Manage access rights and permissions.

10. Operating System Services

- **User Services:**
  - **User Interface (UI):** CLI, GUI, Touchscreen.
  - **Program Execution:** Load, run, and terminate programs.
  - **I/O Operations:** Handle device and file I/O.
  - **File-System Manipulation:** Perform file/directory operations and manage permissions.
  - **Communication:** Between processes via shared memory or message passing.
  - **Error Detection:** Handle and debug errors.
- **System Services:**
  - **Resource Allocation:** Manage CPU, memory, storage, I/O devices.

- **Accounting:** Track resource usage.
- **Protection and Security:** Control access and authenticate users.

11. User Interfaces

- **CLI (Command-Line Interface):** Text-based commands; known as shell.
- **GUI (Graphical User Interface):** Visual interaction with windows, icons, menus.
- **Touchscreen Interface:** Uses gestures and virtual keyboards.

12. System Calls

- **Definition:** Interface for programs to request OS services via APIs.
- **Types:**
  - **Process Control:** Create/terminate processes, load/execute programs.
  - **File Management:** Open, read, write, close files.
  - **Device Management:** Request/release devices.
  - **Information Maintenance:** Get/set system data and attributes.
  - **Communication:** Send/receive messages.
  - **Protection:** Control access and set permissions.

13. System Programs

- **Purpose:** Provide environment for program development and execution.
- **Categories:**
  - **File Management:** Manipulate files/directories.
  - **Status Information:** Display system info.
  - **File Modification:** Editors and text manipulation tools.
  - **Programming Support:** Compilers, assemblers, debuggers.
  - **Program Loading/Execution:** Load and run programs.
  - **Communications:** Facilitate virtual connections.

14. OS Design and Implementation

- **Design Goals:**
  - **User Goals:** Usability, efficiency, functionality.
  - **System Goals:** Simplicity, efficiency, reliability.
- **Policy vs. Mechanism:**
  - **Policy:** What will be done.
  - **Mechanism:** How to do it.

    - **Separation:** Allows flexibility and adaptability.
- **Implementation Languages:**
    - **Assembly:** Low-level operations.
    - **C/C++:** Main OS code.
    - **Scripting Languages:** For utilities and system programs.

15. Operating System Structures

- **Simple Structure:** Minimal separation; e.g., MS-DOS.
- **Monolithic Structure:** Large kernel with many functions; e.g., UNIX.
- **Layered Approach:** OS divided into layers; modularity and ease of debugging.
- **Microkernel Structure:** Minimal kernel with services in user space; communication via message passing; e.g., Mach.
- **Modules:** Core kernel with dynamically loadable modules; e.g., Linux.
- **Hybrid Systems:** Combine multiple structures for performance and security; e.g., Windows, Mac OS.

16. Operating-System Debugging

- **Debugging:** Finding and fixing errors.
- **Tools:**
    - **Log Files:** Record errors.
    - **Core Dump:** Memory image of failed process.
    - **Crash Dump:** Memory image after OS failure.

17. System Boot

- **Process:**
    - **Initialization:** Starts at a fixed location when powered on.
    - **Bootstrap Loader:** Loads OS kernel into memory.

18. Important Concepts and Quiz Tips

- **Key Terms:**
    - **OS as Intermediary:** Between user and hardware.
    - **Components:** Hardware, OS, Applications, Users.
    - **OS Functions:** Resource Allocator, Control Program.
    - **Trap:** Software-generated interrupt.
    - **Interfaces:** CLI, GUI, Touchscreen.

- **CLI:** Also called shell.
- **GUI Origin:** Xerox PARC.
- **System Call:** Programming interface to OS services.
- **Design Principle:** Separate Policy from Mechanism.
- **OS Structures:** Simple, Monolithic, Layered, Microkernel, Modules, Hybrid.
- **Microkernel Communication:** Message passing.
- **Debugging:** Process of finding/fixing errors.
- **Bootstrap Loader:** Code that loads the kernel.

- **True/False Example:**
  - "A single type of system call manages the whole process in a system." — **False.**

19. Processes (Preview of Chapter 3)

- **Process Concept:**
  - **Process:** Program in execution; basis of computation.
  - **Process States:** New, Running, Waiting, Ready, Terminated.
- **Process vs. Threads:**
  - **Process:** Independent execution with its own memory space.
  - **Thread:** Execution unit within a process; shares memory with other threads.