Summary of Chapter 2: Operating System Structures

1 Chapter 2: Operating-System Structures

This chapter focuses on the internal structure of operating systems, providing insight into the services offered, system calls, and the overall design and implementation.

2 Objectives

The key goals are:

- To describe the services an OS provides to users, processes, and other systems.
- To discuss various methods for structuring an OS.
- To explain how OSes are installed, customized, and booted.

3 Operating System Services

Operating systems provide several services to programs and users:

- **User Interface**: May include Command-Line Interface (CLI), Graphical User Interface (GUI), or Batch interface.
- **Program Execution**: Load programs into memory and manage their execution.
- **I/O Operations**: Manage I/O requests from running programs.
- **File-System Manipulation**: Create, delete, and manipulate files.
- **Communication**: Enable inter-process communication either through shared memory or message passing.
- **Error Detection**: Detect errors in CPU, memory, and I/O devices, and take appropriate actions.

- **Resource Allocation**: Manage resource allocation among multiple users or jobs.
- **Protection and Security**: Control access to system resources, ensure user authentication, and prevent unauthorized access.

4 User Operating System Interface

There are different types of interfaces provided by OSes:

- **CLI (Command-Line Interface)**: Allows direct command entry by the user.
- **GUI (Graphical User Interface)**: Uses visual metaphors like icons and windows to interact with the OS.
- **Touchscreen Interfaces**: Designed for devices where mouse interaction isn't feasible, utilizing gestures and voice commands.

5 System Calls

System calls are the interface through which a program requests services from the OS:

- **Process control**: Create, terminate, and manage processes.
- **File management**: Open, close, read, and write files.
- **Device management**: Request or release a device, and manage device attributes.
- **Communication**: Send and receive messages, or access shared memory regions.
- **Protection**: Control access to resources and manage user permissions.

Parameters are passed to the OS via registers, tables, or the stack, depending on the architecture.

6 System Programs

System programs provide an environment for program development and execution. They include:

- **File management**: Tools for creating, deleting, and manipulating files.
- **Programming support**: Compilers, assemblers, and debuggers.

- **Communications**: Programs for virtual connections, message transfers, and remote logins.
- **Background services**: Utilities like disk checking, error logging, and printing.

7 Operating System Design and Implementation

The design and implementation of an OS involve specifying system goals, defining policies, and selecting mechanisms. Key principles include:

- **User Goals**: The OS should be easy to use, learn, and reliable.
- **System Goals**: The OS should be flexible, efficient, and easy to maintain.
- **Policy vs. Mechanism**: Policies decide what will be done, while mechanisms determine how it will be done.

8 Operating System Structures

Operating systems can have various structures depending on the design:

- **Simple Structure (MS-DOS)**: Minimal modularity with limited separation between layers.
- **Non-Simple Structure (UNIX)**: Contains a kernel with CPU scheduling, memory management, and file systems.
- **Layered Approach**: OS is divided into layers, each depending only on the lower layers, which simplifies debugging and implementation.
- **Microkernel Structure**: Moves as much as possible into user space, simplifying the kernel and making the system easier to extend and port.
- **Modules**: Many modern OSes use loadable kernel modules to add flexibility.
- **Hybrid Systems**: Modern OSes, like Mac OS X, combine layered, modular, and microkernel structures.

This chapter highlights the complexity of operating systems and the different design choices that impact performance, reliability, and usability.