Chapter 2:

Operating System Services

One set of operating-system **services** provides functions that are **helpful to the user**:

User interface: Varies between Command-Line (CLI), (GUI), Batch.

Program execution, **I/O operations**, **File-system manipulation**, **Error detection**

Communications – Processes may exchange information, on the same computer or between computers over a network.

Another set of OS functions exists for ensuring the efficient operation **of the system itself via resource sharing.**

Resource allocation

Accounting - To keep track of which users use how much and what kinds of computer resources.

Protection and security- concurrent processes should not interfere with each other

CLI or command interpreter allows **direct** command entry

implemented in **kernel** or by **systems program**.

Touchscreen devices require new interfaces

System Calls

System Calls: Programming **interface** to the services provided by the OS.

Typically written in **a high-level language** (C or C++).

Mostly accessed by programs via (**API**) rather than direct system call use.

Typically, a number associated with each system call.

System-call interface maintains a **table indexed** according to these numbers.

Most details of OS interface hidden from programmer by **API**

System Call Parameter Passing

Three general methods used to pass parameters to the OS:

Simplest: pass the parameters in registers

In some cases, may be more parameters than registers.

Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register.

Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system.

Block and **stack** methods do not limit the number or length of parameters being passed.

SYSGEN- Used to build system-specific compiled kernel or system-tuned. Can general more efficient code than one general kernel.

Types of System Calls

Process control, File management, Device management, Information maintenance (get time or date, set time or date), Communications, Protection

System Programs

They can be divided into:

File manipulation, Status information (ask for info - date, time, amount of memory, disk space) sometimes stored in a File modification

Programming language support, Program loading and execution, Communications, Background services.

Text editors to create and modify files.

Some systems implement a **registry** - used to store and retrieve configuration information.

Operating System Design and Implementation

Important principle to separate

Policy: **What** will be done?
Mechanism: **How** to do it?

Mechanisms determine how to do something, **policies** decide what will be done. The separation of policy from mechanism is a very important principle, it allows **maximum flexibility** if policy decisions are to be changed later (example – timer).

Specifying and designing an OS is highly creative task of **software engineering**

Operating System Structure

1- Simple Structure -- MS-DOS:

MS-DOS – written to provide the most functionality in the least space.

Not divided into modules.

its interfaces and levels of functionality are not well separated.

2-Non Simple Structure -- UNIX:

the original UNIX operating system **had limited structuring.**

Consists of everything **below** the system-call interface and **above** the physical hardware.

Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

3-Layered Approach:

The bottom layer (**layer 0**) is the **hardware**; the highest (**layer N**) is the **user interface**.

With modularity, layers are selected such that each uses functions (operations) and services of **only lower-level layers**.

4-Microkernel System Structure:

Moves as much from the kernel into user space. **Mach** example of **microkernel.**

Communication takes place between user modules using **message passing**

Performance **overhead** of **user** space to **kernel** space communication.

5-Modules:

Uses **object-oriented** approach, Each core component is separate, Each talk to the others over known interfaces. Each is **loadable as needed within the kernel**

Overall, similar to layers but with **more flexible**

6-Hybrid Systems:

Most modern operating systems are **not** one **pure** model. Hybrid combines multiple approaches to address performance, security, usability needs. so **monolithic**, plus **modular** for dynamic loading of functionality.

Android

Open Source, Adds power management.

Operating-System Debugging

Debugging is finding and fixing **errors**, or **bugs**.

OS generate **log files** containing **error information**.

Failure of an application can generate **core dump** file capturing memory of the process.

Operating system failure can generate **crash dump** file containing kernel memory.

Kernighan's Law: "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."