Summary of Andrew S. Tanenbaum - Modern Operating Systems:

<https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Modern%20Operating%20Systems.pdf>

Introduction:

1.1:

A modern computer is a complex system comprising various hardware components like processors, memory, disks, and I/O devices. To manage these efficiently and provide a simplified interface for programmers, computers use an operating system (OS). The OS serves as an intermediary, offering a cleaner, abstracted view of the hardware while managing resources.

Users typically interact with a shell or GUI, but these are not part of the OS itself. The OS operates primarily in kernel mode, with full access to the hardware, while other software runs in user mode, with limited access. This separation ensures security and stability. Some systems blur this distinction, especially in embedded or interpreted systems.

Operating systems are complex and long-lasting, often evolving over time rather than being completely rewritten. For instance, different versions of Windows or UNIX-based systems like Linux show how OSs maintain compatibility while evolving.

A key function of the OS is to provide abstractions that simplify hardware interactions. For example, instead of dealing directly with complex disk operations, the OS offers file systems that are easier to use. This abstraction is crucial for managing the complexity of hardware and making programming more accessible.

Ultimately, the OS serves application programs by offering consistent, manageable interfaces, transforming the complexities of hardware into user-friendly operations. While users see different interfaces (e.g., Windows desktop vs. command line), these are built on the same underlying OS abstractions.

The concept of an operating system can be viewed in two ways: from the top-down, as a provider of abstractions for application programs, or from the bottom-up, as a manager of a computer's complex components. The bottom-up view emphasizes the OS's role in managing and allocating resources like processors, memory, and I/O devices to ensure orderly operation, especially when multiple programs run simultaneously.

Modern operating systems allow multiple programs to run concurrently, managing potential conflicts, such as when several programs try to use the same printer. The OS handles this by buffering output and ensuring that resources are used in an organized manner.

In multi-user environments, resource management and protection become even more critical. The OS is responsible for tracking resource usage, granting access, and mediating conflicts. Resource management involves multiplexing, which can occur in two ways: time multiplexing, where programs take turns using a resource (e.g., CPU or printer), and space multiplexing, where resources like memory and disk space are divided among users or programs. The OS ensures fairness, protection, and efficient resource use in both cases.

1.2: