# Chapter 1 - OS Introduction

1. **Introduction – Outline:**

   The major objectives of this chapter are:
   1. To provide a grand tour of the major components of operating systems, and
   2. To describe the basic organization of computer systems.

2. **What is an Operating System?:**

   - OS provides an abstraction. Example, The OS abstracts the tracks/sectors for disk storage with the file system abstraction. The user does not need to know where each data byte is saved on the physical storage medum.
   - The embedded device may not use an OS, but might still need to interact with some IO devices, like the display, but that will be handled by the application itself.

3. **User's View of the Operating System:** The OS goal here is to make the user's life easier.

   - From the user's viewpoint, the OS provides an easier environment to work in than that provided by pure hardware, which includes the processor and the I/O devices. The OS also provides more powerful instruction and functions that those provided by the hardware, eg., write(fileno, buf, len);
   - Note the difference between library call and system call. While the library call is provided by the language *runtime*, the system call is provided by the OS.
   - Maintaining memory and program state are complex operations that are abstracted by the OS.
   - When multiple users are accessing the same computational resources then, the OS manages the CPU time, memory and I/O resources, schedules them fairly among all the users of the system. Other users should not be able to view or modify other user's program state. Each user feels that she has the computer all to herself.
     Fairness: If some user program goes into an infinite loop, then the other programs should still run unaffected.
     Performance: The OS should not take very long for its own activities, to the detriment of application performance.

4. **System's view of the Operating System:** Hardware resource allocation, management and resource monitoring.

   From the system;s point of view it is not critical to make the user's life easier, but more important to use hardware resources efficiently.

   The OS is most intimately involved with managing the hardware resources. Conflict resolution is important whenever the system has multiple simultaneous programs executing, from one or multiple users.

   What is fair? Higher priority processes, but who gets higher priority? Real-time processes, critical system applications, high-paying users on batch systems.

   The view of the OS as a control program only provides a slight variation over the previous view.

5. **What Does an Operating System Do?** A good analogy of an OS is the government. It does no useful work on its own, but provides an environment for other people and industries to do their work.

   However, we still do not have a very good definition of the OS. Some OSes are less that 1Mb and others are in GB sizes.

   OS vendors need to be careful regarding what they ship as part of the OS. Microsoft has got into numerous problems for shipping applications with the OS that the courts decided were not a core functionality of the OS, and thawrted competition for other vendors, like music player, and web browzer.

6. **Components of a Computer Systen:**

   - The OS is mostly written in software, although that is not necessary. In fact, there have been attempts to write OSes in hardware, particularly using FPGAs for embedded applications.
   - The programs provided by the OS are called *system calls*. Application programs access the hardware by invoking these system functions.

7. **Computer System Operation:**

   - Bootstrapping ("to pull oneself up by one's bootstraps") refers to techniques that allow a simple system to activate a more complicated system. The bootstrap program does not need to modify often, and hence is present on ROMs, EEPROMS, or firmware in general. It initializes all aspects of a system.
   - The first program is called *init* on Unix-based systems.
   - Software interrupts can be triggered by the software, either by implicitly (such as divide by zero errors or by calling privileged instructions), or explicitly by invoking a system call. These are called traps or exceptions. Exceptions are triggered by the hardware, and so are not associated with a specific software instruction.
   - Occurence of an interrupt stops normal execution of the current program. The interrupt vector contains the addresses of all the service routines for a fixed number of interrupts supported by that system. This interrupt vector table is a table of pointers stored in low memory addresses. This interrupt mechanism is used to perform all useful tasks on a modern computer.
     eg., the execve() function call to start a new program.
     eg., the timer interrupt to schedule programs, etc.

8. **Operating System Operations:** What is an interface? Communication link between 2 computer layers. We will talk more about the abstraction provided by the OS, and how it simplifies the hardware environment for the user in Chapter 2.

9. **Evolution of Operating Systems:**

   - **Batch OS:** Goal: Simplify OS structure.

   - **Multiprogramming OS:** Goal: Maximize system throughput / performance. Keep CPU busy all the time.
     Multiprogramming is also how people generally operate. We have several tasks on our hands, and if one is waiting for something, then we go on to the others.

- **Time-sharing OS:** Goal: Provide an interactive OS environment.

  Time sharing enables an interactive environment for program execution. Allows the OS to produce a virtualization in which each user believes that the entire computer system is dedicated to her use.

  Virtual memory allows execution of programs that are larger than the available physical memory. It also abstracts memory into a uniform array of storage, separating *logical memory* as seen by the user from physical memory. This is a very useful abstraction, and we will study it in more detail later in Chapters 8 ad 9.

10. **Operating System Tasks:**

11. **Process co-ordination and security:**

    - The OS needs to ensure that an error or malicious activity in one process does not affect any any program in a multiprocess/multiuser environment. For example, if a program got stuck in an infinite loop, then this loop could affect the operation in other concurrent processes. Without protection, the computer will be restricted to execute only one process at a time.
    - Kernel mode is also called system mode, privileged mode, or supervisor mode. Mode bit is provided by the hardware. System executing normal user applications is in user mode. User programs can invoke system calls to request OS to perform a privileged service. The system then transitions to system mode. Instructions that can affect the correct operations of other processes in the system are called privileged instructions. We will see an example on the next slide with the system timer.
    - MS-DOS was written for the 8088, which had no mode bit, and hence no dual mode. DOS therefore allowed user programs unrestricted access to the system, and could erase the entire OS, if the program were not careful. Most current architectures provide the mode bit, and OSes take advantage of it to provide security.
    - In Unix-like systems, ”fork” and ”execve” are C library functions that in turn execute instructions that invoke the ”fork” and ”execve” system calls. Making the system call directly in the application code is more complicated and may require embedded assembly code to be used (in C and C++) as well as knowledge of the application binary interface; the library functions are meant to abstract this away.

12. **Transition from User to Kernel Mode:** Timer ensures that the OS remains in control of the CPU resources. Timer can prevent a single user program from running too long, and being unfair to other time-shared programs. All instructions managing the timer setting, decrement, reset, etc. are privileged instructions.

13. **Process Management:**

    - All programs running on the computer, web-browser, word, games, compiler, etc. run as processes. A static program when initialized for execution or doing useful work, is a process. Thus, we can start multiple instrances of a web-browser as distinct processes, although they are running the same program.
    - Resources needed by a process include: cpu, memory and I/O, files, and well as initialization data. Typically system has many processes, some user, some operating system running concurrently on one or more CPUs. Concurrency is achieved by multiplexing the CPUs among the processes / threads.
    - We will also study threads, which are light-weight processes, sharing some resources.

14. **Process Management Activities:** We will discuss mechanisms for performing all these process management activities in this class.

15. **Memory Management:** Memory can be considered as a large array of addresses, each address storing a byte or a word of data. The CPU can only address data and instructions from memory, not from disk, or from the cache. Addresses produced by the CPU are memory addresses. The only other option is to use registers.

    Why memory management? Because memory is limited, and is often less than the combined requirement of multiple processes wanting simultaneous access to it.

    Memory management is to manage the requirement of the multiple processes present in memory at the same time. This is to improve CPU utilization, and user response.

16. **Storage Management:**

    - Providing storage is important because users cannot use memory to store information for long: memory is limited in size, and is volatile.
    - Several kinds of mass storage media are available for permanant storage of often-used information, as well as for backups of archieval data. Different media have different properties that are difficult and cumbersome to remember. OS makes computer system easier to use by abstracting away details of the various storage media such as magnetic disk, magnetic tape, optical disk, etc. Thus, we do not need to remember the various tracks and sectors locations on a disk where bits of our data might be stored. The OS provides a common abstraction to store information, called a file.
    - When files are shared among multiple users, providing access controls is important. At the same time, providing effecient access is very important.

17. **Storage Heirarchy:** The closer the storage level is to the CPU the better its performance. Smaller size allows faster access, and less power consumption. Faster process technology makes the component more expensive.

    Note that registers and caches are not typically managed by the OS, but the cache may be in some cases. Also note that a data item is backed up all the way, it appears in multiple locations simultaneously, which makes it important to maintain its consistency among the various storage location.

18. **I/O subsystem:**

    - File system mamagement can also be also part of the I/O subsystem. *Drivers* are the OS programs used to manage each I/O device. That is why you are required to install drivers for any new device you want to use with your computer, disks, USB mouse, keyboard, monitors, etc.
    - Buffering is done while printing data to file to reduce overhead of small writes.
    - Disk caches are used to enable faster disk access.
    - Spool refers to the process of placing data in a temporary working area for another program to process. Spooling is useful because devices access data at different rates. Spooling allows one program to assign work to another without directly communicating with it. The most common spooling application is print spooling.

19. **Protection and Security:**

    - We already saw the concepts of address spaces and mode bits and timer for process-level protection.

- A computer virus attaches itself to a program or file. it actually cannot infect your computer unless you run or open the malicious program. A virus cannot spread without human action; worm like virus, but can travel autonomously.
- setuid and setgid are Unix access rights flags that allow users to run an executable with the permissions of the executable's owner or group. setuid and setgid are needed for tasks that require higher privileges than those which common users have, such as changing their login password. Some of the tasks that require elevated privileges may not immediately be obvious, though such as the ping command, which must send and listen for control packets on a network interface.

20. **Summary - Operating Systems:** The OS frequently creates the VM that presents a different view of the system that reality.

    A time-shared OS provides the view of multiple CPUs. Virtual memory provides a view of infinite memory capacity. File system abstraction provides a view of simple, flexible, random-access data access. Insecure networks are made secure as well as reliable by implementing protocols such as TCP/IP. Distributed OS, or multi-core OS takes multiple processors or systems and provides the view of a single high-performance system.