What is an operating system

- An operating system is a program that acts as an intermediary between the user of a computer and the computer hardware.
- It provides an environment in which user can execute programs in a convenient and efficient manner.
- It performs no useful function on its own.
- Pure overhead of real work. (No useful work)
- Application programs have the real value to person who buys the computer.

Why study Operating System?

- It helps you to understand the **model of the operation**.
    - Easier to see how to use the system.
    - Enables you to write efficient code.

Computers and Software

System software vs application software

- Computer systems consist of software and hardware.
- Software is differentiated according to its purpose.

Application Software

- Software that allows the user to perform some intended task, function or activity and includes productivity tools.

System Software

- Software that provides an interface with hardware and serves as a platform for running programs and maintaining the efficiency of the system. It can be divided into operating systems and utility programs.

3 Perspectives of the Computer

- End User View (e.g. cut, save, print)
- Application Programmer View (e.g. malloc(), open(), fork())
- OS Programmer View (e.g. read-disk, track-mouse, start-printer)

System Software

- System software provides two kinds of environment.
    - Allows human users to interact with the computer.
    - Provides tools and subassemblies used with application programs.
- Independent of individual applications, but common to all of them.
- E.g.
    - C library functions
    - A windowing sub-system
    - A Database Management System
    - Resource management functions
    - The OS

  - CLI
  - Complier

Hierarchy of Software

1. Human-Computer Interface
2. Application Software
   - API
3. System Software
   - OS Interface
4. Trusted OS (Abstract Resources)/ Kernel
   - Software-Hardware Interface
5. Hardware Resources

- The OS uses the functionality at the software-hardware interface.
- The system software uses the OS interface to export the API.
- Application programs use the API to create software that implements the human-computer interface.

The OS as Resources Manager

- Resource (Anything that is needed for a executing program to run):
  - Memory
  - Space on the disk
  - CPU
- Operating system can be viewed as a resource manager
  - "An OS creates resource abstractions"
  - "An OS manages resource sharing"

Resource Abstraction

- Abstraction is when an OS hides the actual tasks needed to manage and use resources.
- Allows user programs to use these resources by using simple commands to access these resources.
- Makes it easy for user programs to use resources in a computer system.
- E.g.
  - Writing a file to disk
  - Displaying text/ graphics on screen
  - Running an application
- Simplifies usage but limits flexibility
  - Certain operations become easy to perform while other operations may be impossible to achieve.

```
load(block, length, device);
seek(device, 236);
out(device, 9);

──────────────────────────────────────
write(char *block, int len, int device, int track, int sector) {
        ...
        load(block, length, device);
        seek(device, 236);
        out(device, 9);
        ...
    }
______________________________________
write(char, *block, int len, int device, int addr);
______________________________________
fprintf(fileID, "%d", datum);
```

Disk Abstractions

3 Different abstractions

- Direct Control
- `write()` abstraction
- `fprintf()` abstraction

Abstract Resources

- User Interface
- Application
- Abstract Resources (API)
- Middleware
- OS Resources (OS Interface)``
- OS
- Hardware Resources

Resource Sharing

- Two kinds of sharing
  1. Space-multiplexed sharing
     - The resource is divided into two or more distinct units and each unit it allocated to different processes.
  2. Time-multiplexed sharing
     - The entire resource is allocated to a process for a period of time, after which it is then allocated to another process and so on.

Multiprogramming

- Refers to the technique for sharing the CPU among runnable processes
- How does it work?
  - Process may be blocked on I/O.
  - Process may be blocked waiting for other resource, including the CPU.
  - While one process is blocked, another might be able to run.
  - Increase CPU utilization.
- Multiprogramming OS accomplishes CPU sharing "automatically" - scheduling

OS Strategies

- Different strategies have been used to provide OS services.

- Refers to the general characteristics of the programmer's abstract machine.
- Depends on business and engineering criteria.
    - How will the computer be used?
    - Is human interaction important?
    - Will there be more than one person using?
    - Is response time critical?
- Batch processing
- Timesharing
- Personal computer & workstations
- Others
    - Process control & real-time
    - Network
    - Distributed
    - Small computers

Batch Processing

- Uses multiprogramming
- Job (file of OS commands) prepared offline
- Batch of jobs given to OS at one time
- OS processes jobs one-after-the-other
- No human-computer interaction
- OS optimizes resource utilization
- Batch processing (as an option) still used today

Timesharing Systems

- Uses multiprogramming
- Support interactive computing model (Illusion of multiple consoles)
- Different scheduling & memory allocation strategies than batch
- Tends to propagate processes
- Considerable attention to resource isolation (security & protection)
- Tend to optimize response time

Kernel basic facilities

- The kernel's primary purpose is to manage the computer's resources and allow other programs to run and use these resources.
- Typically, the resources consist of:
    - CPU
    - Memory
    - I/O Devices