1.Intro

- OS
  - making it easy to run programs
  - allowing programs to share memory
  - enabling programs to interact with devices
  - making sure the system operates correctly and efficiently in an easy-to-use manner. => virtualization

"How does the operating system virtualize resources"

- the OS takes a physical resource and transforms it into a more general, powerful, and easy to use virtual form of itself. (OS == virtual machine)

- OS provides some interfaces (APIs) that you can call.
- OS provides a standard library to applications.

- Virtualization allows many programs to
  - run (sharing the CPU)
  - concurrently access their own instructions and data (sharing memory)
  - access devices (sharing disks and so forth)

- OS sometimes known as a resource manager

1. **Virtualizing the CPU** : Turning a single CPU into a seemingly infinite number of CPUs and thus allowing many programs to seemingly run at once

2. Virtualizing Memory

- Memory : just an array of bytes

- To read memory : one must specify an address to be able to access the data stored there.

- To write memory : one must also specify the data to be written to the given address

- Virtualizing Memory : each running program (=process) has its own private memory(virtual address space), instead of sharing the same physical memory with other running programs.

3. Concurrency

- OS is juggling many things at once -> 99 problems ~~
- Thread : function running within the same memory space as other functitons, with more than one of the active at a time.
- instruction should be executed atomically

4. Persistence

- hardware - I/O device : hard drive, solid-state drives
- software - file system : software in the OS that usually manages the disk.
  - responsible for storing any files the user creates in a reliable and efficient manner on the disks of the system.

- Unlike the abstractions provided by the OS for the CPU and memory, the OS does not create a private, virtualized disk for each application.
- Users will want to share information that is in files.

- For performance reasons, most file systems first delay such writes for a while, hoping to batch them into larger groups.
- Carefully ordering writes to disk to ensure that if a failure occurs during the write sequences, the system can recover to reasonable state afterwards.

5. Design Goals

OS

- Takes physical resources -> virtualizes them
- handles tough and tricky issues related to concurrency
- stores files persistently
- Finding the right set of trade-offs is a key to building systems.

Abstractions
: fundamental to everything we do in computer science

Goal : minimize the overheads of the OS

- Virtualization and making the system easy to use are well worth it, but not at any cost
- overheads
  - extra time = more instructions
  - extra space = in memory or on disk

Isolation

- Isolating processes from one another is the key to protection and thus underlies much of what an OS must do.

Reliability : OS strive to provide a high degree of reliability / non-stop

6. Some History

Beyond Libraries : Protection

- system call vs. procedure call
  - System call : transfers control into the OS while simultaneously raising the hardware privilege level.

The Era of Multiprogramming

- The desire to support multiprogramming and overlap in the presence of I/O and interrupts forced innovation in the conceptual development of OS along a number of directions. => memory protection / concurrency