Operating System:

* Provide consistent software abstraction to applications, **even on different hardware**
* Manage **sharing of resources** among multiple applications

Building Blocks:

* Processes, Threads, Concurrency, Scheduling, Coordination
* Address Spaces
* Protection, Isolation, Sharing, Security
* Persistent storage, transaction, consistency, resilience
* Interface to all devices

System

* Complex interaction of multiple components in multiple administrative domains
  + Systems, services, protocols

What does an Operating System do?

* Memory Management, I/O management
* CPU Scheduling
* Multitasking
* File System
* Multimedia Support

No universally accepted definition of an Operating System

* Special layer of software that provides application software access to hardware resources
  + Convenient abstraction of complex hardware devices
  + Protected access to shares resources
  + Security and authentication

What makes something a system?

* Multiple interrelated parts
* Robustness:
  + Meticulous error handling, defending against malicious careless users
  + Treating computer as concrete machine, with all of its limitations

OS **abstracts** these hardware details from the application

What is an OS?

* **Illusionist:**
  + Provide clean, easy to use abstractions of physical resources
    - Infinite memory, higher level objects: files, users, messages
    - Program behaves as if it has processor entirely to itself
    - Masking limitations, virtualization to make it easy to programing
  + Processor -> Threads
  + Memory -> Address Spaces
  + Storage -> Files
  + Networks -> Sockets
  + Process abstraction over all these ^
    - Execution environment with restricted rights provided by OS
  + A picture containing text, screenshot, diagram, rectangle

    Description automatically generatedCompiled Program
* Application’s “machine” *is* the process abstraction provided by the OS
* Each program runs in its own process (which gives access to threads, address space, etc)

A process consists of:

* Address space
* One of more threads of control executing in that address space
* Additional system state: Open files, open sockets, etc
* Isolated environment **protected from other processes**

Many processes can run at the same time, BUT THEY ARE ISOLATED FROM EACH OTHER

* **Referee:**
  + Manages protection, isolation, and sharing of resources
    - Resource allocation and communication

A screenshot of a computer program

Description automatically generated with medium confidence

OS can provide illusion that there is more than one processor -> More than on process running

* Periodically there is **a process switch, registers of each process** switched into **memory** in **process control block**
* Protection**:** 
  + If some process tries to use another process address space, OS intervenes, causes segmentation fault
  + OS provides protection boundary over hardware, part of virtual machine
  + Convenient programming abstraction so that you don’t have to worry about limited or shared resources
* **Glue:**
  + Common services:
    - Storage, window system, networking
    - Sharing, authorization
    - Look and feel
  + OS provides common services in the form of I/O

**Syllabus**:

* OS Concepts:
  + Processes, I/O, Networks, VMs
* Concurrency:
  + Threads, scheduling, locks, deadlocks
* Address Space:
  + Virtual memory, address translation, protection sharing
* File Systems
  + -I/O devices, file objects, storage, naming, caching, performance, paging, transactions, databases
* Distributed Systems
  + Protocols, N-Tiers, RPC, Consistency, Scalability
* Reliability & Security
  + Fault tolerance, protection, security

The world is a large distributed system

* Microprocessors in everything
* Vast infrastructure behind them
* Databases, Information Collection, Remote Storage, Online Games, Scalable Reliable Secure Services 🡨 Operating systems make all this happen

Moore’s Law meant computers were 2x fast every 2 years (shove more transistors in chip)

Moore’s Law ended 🡪 Multi-core chips created (need for parallelism)

**The world is Parallel**

Network capacity still increasing

Storage increasing

Number of devices increasing -> People-To-Computer ratio increasing

**OS -> Referee, Illusionist, Glue**

Does the programming need to write a single program that performs many different activities?

Does every program have to be altered for every piece of hardware?

Does a faulty program crash everything?

Does every program have access to all hardware?

Hopefully, no! The OS helps the programmer write robust applications!

A screenshot of a computer

Description automatically generated with medium confidence