CS 453/552: Operating Systems
An *Operating System* is a system software that acts as an intermediary between
- user and resources (could be hardware or abstract)
- application software and resources
- other system software and resources

**Application software versus System software.**
An operating system has *two* major functions.
- resource abstraction
- resource sharing
Resource Abstraction

- provide abstract models of hardware components.
- a good abstraction (or interface) is general across resources, yet easy to use.
- abstraction can be carried out at several levels.
Abstraction Example: A disk drive

Read/Write Head

Sector

Track (or Cylinder)

Surface
Abstraction Example: Logical Block Address (LBA)

Disk geometry:

\[
\text{track/cylinder} \in [0 \ldots c - 1] \\
\text{surface} \in [0 \ldots t - 1] \\
\text{sector} \in [0 \ldots s - 1]
\]

- How to calculate the LBA on a disk drive from a physical location?
- How to calculate the physical address from a LBA on a disk drive?
Resource Sharing

- Sharing can be of two types.
  - **Space-multiplexed.** Resource can be divided into two or more units. e.g. memory, disks.
  - **Time-multiplexed.** Resource must be given exclusively. e.g. processor.

- Should prevent unauthorized sharing while still allowing authorized sharing.

- Resource isolation relies on the operating system being trustworthy. The operating system, in turn, relies on hardware for protection.


  - IBM System/360. A family of machines with the same architecture and a batch operating system. Fred Brook’s *Mythical Man-Month* came out of this experience.
  - Interactive Timesharing introduced in CTSS at MIT and later in MULTICS and Unix.


Mobile generation (2000–now). Network is the new norm. Large scale deployment of embedded systems for mobile devices. Corresponding rise in large scale distributed systems.
Operating System Strategies

- Batch processing systems.
- Timesharing systems.
- Personal computers and workstations.
- Real-time computers.
- Networks of computers.
A **batch job** is a predefined collection of commands that are executed without any interaction with the users. The operating system reads in the entire job and sets it up for execution.

Useful for running long sequences of commands, off-line processing, performing tasks at certain times etc. All modern operating systems support some form of batch processing.

Batch processing systems in the 60’s led to early implementations of *multiprogramming*, scheduling and swapping.

The main issue is to optimize the use of the resources to finish a group of batch jobs.
A shell script for renaming a group of files. The ‘cmd’ executes the command and substitutes its output into the expression. An alternative syntax is $(cmd).

```
#!/bin/sh
for f in *.$1
do
    base=`basename $f .$1`
    mv $f $base.$2
done
```
A shell script for counting the number of lines in all C programs in the current directory and all subdirectories recursively.

```bash
#!/bin/sh

total=0
for currfile in `find . -name "*.c" -print`
do
  total=$((total+(`wc -l $currfile| awk '{print $1}')`))
  echo 'total=' $total
done
```
Several users interact with the computer simultaneously.
Each user has a virtual machine all to themselves.
The main criteria for CPU scheduling and memory sharing is to give a fair share to all the competing users.
Each users can also run several programs at the same time.
Important to establish barriers and safeguards between users and processes.
What is a Process?

- A program in execution.
- A process is a working structure, a (potentially) huge information refinery buzzing and blazing with activity as masses of information move around inside.
- A process is an information machine, merely enacted, temporarily embodied by an irrelevant hunk of metal, plastic and silicon called a computer. (From the book *Mirror Worlds*... by David Gelernter)
- Personal Computers and workstations led to development of many personal computing tools and to graphical user interfaces.

- A process control computer is used to control a single application. The primary issue is efficiency and correctness rather than generality and usability.

- Real-time computing is based on providing deadlines by which a response is received for performing some work. New issues in real-time computing have arisen from network-applications and embedded systems. For example, guaranteeing a minimum amount of deviation in the transfer rate of information on the network.

- Networks led to the development of client-server computing. Network operating systems and distributed operating systems have now become more prevalent.