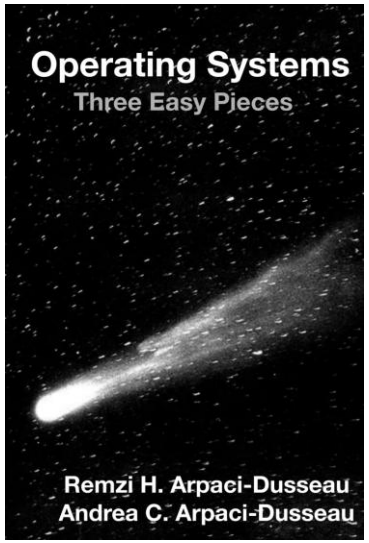
CS3.301: Operating Systems and Networks
Lecture 1 & 2: Introduction to OS

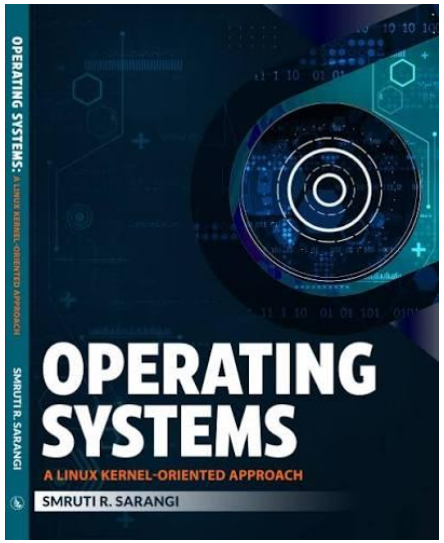
INSTRUCTOR: DR. DEEPAK GANGADHARAN

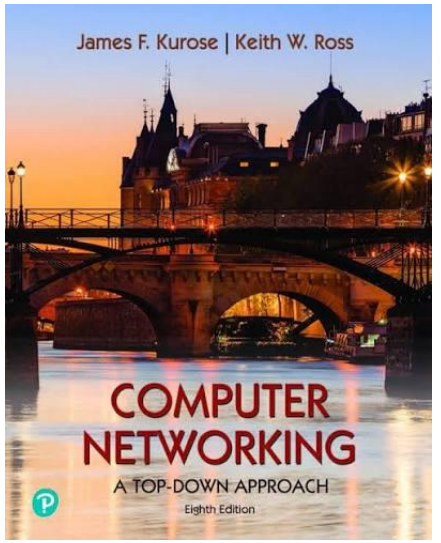
Acknowledgement

Materials in the slides prepared based on the following books and some online materials

1. Operating systems: Three easy pieces by Andrea Arpaci-Dusseau and Remzi Arpaci-Dusseau, 2018 (https://pages.cs.wisc.edu/~remzi/OSTEP/)
2. Operating Systems: A Linux Kernel-Oriented Approach by Smruti R. Sarangi, 2025
3. Computer Networking by James F Kurose and Keith W Ross, 2020







Grade Distribution

Component	Weightage
Quiz	10%
Mid-term Exam	15%
End-Sem Exam	25%
Mini Projects	25%
Project	25%
In-class participation/Bonus	5%

Course Logistics

- Course Announcements, Materials and Information – Moodle
- Instructor Office Hours – Will be announced soon!
- TA Office Hours – Will be announced after finalization of TAs (need to strictly follow timing)
- Zero Tolerance on Plagiarism! – Can result in F grade
- AI tools can be used for the mini projects and project, but use them wisely!
  - You need to acknowledge which tool has been used and how did you use it
  - Provide what prompt was used and screenshot of the output
  - However, you need to understand the code you have generated
  - If not properly acknowledged, we reserve the right to award a 0
- Respect the class timings!

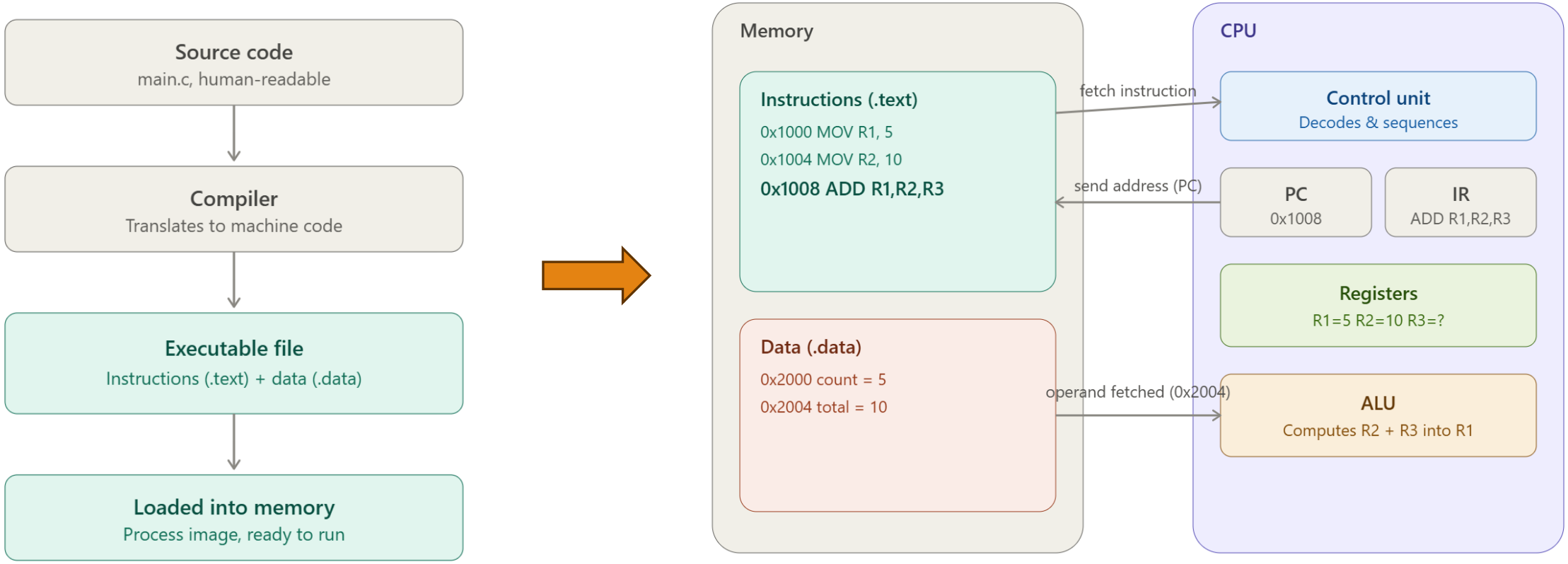
What is an OS?

Wikipedia
An operating system (OS) is system software that manages computer hardware and software resources, and provides common services for computer programs.

GPT 5.5
An Operating System (OS) is system software that acts as an intermediary between computer hardware and application software. It manages the computer's hardware resources and provides services that applications need to run.

Claude Sonnet 5
An operating system (OS) is the software that manages a computer's hardware and provides a foundation for everything else running on the machine. Think of it as the layer between the raw physical components (CPU, memory, storage, screen, etc.) and the applications you actually use, like a browser or a word processor.

Application to Execution – What we know!



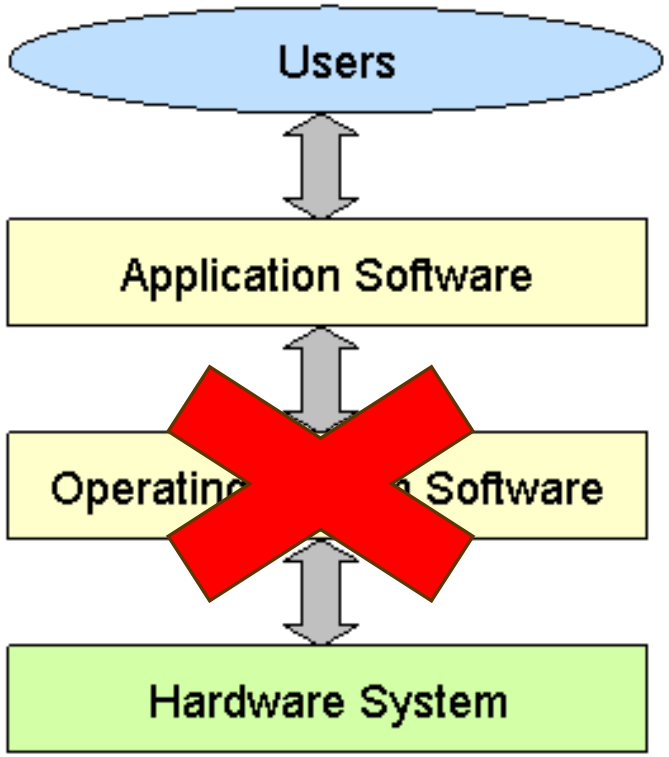
Source: Claude Sonnet 5

What is an OS?

Let us try to answer that by first thinking “what if there was no OS”!!

What if there was no OS?

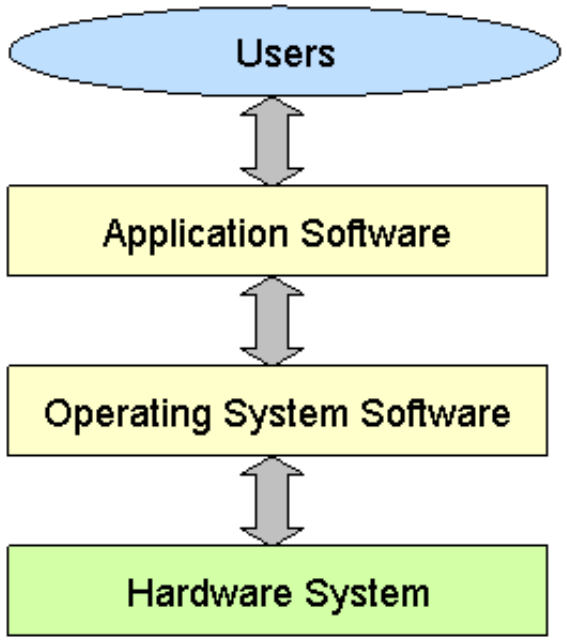
- Application development burdened with hardware details - - - Programmers to write complex routines in application code to use appropriate protocols to interact with hardware
- In multi application scenario, who will manage resources?
- Who will ensure security of applications?



Source: https://softwareg.com.au/blogs/computer-hardware/application-software-interacts-directly-with-the-computer-hardware?srsltid=AfmBOoq58Qe9M09PNgTaQg-j6nztrNzUn_szuvqXEOgc25eL6rkjIXbg

Operating Systems: Overview

- A middleware or abstraction layer between the application programs and the underlying hardware
- Goals of the OS:
  - Execute user programs and make it easy to develop programs
  - Make the hardware system convenient to use
  - Efficient usage of hardware resources



Source: https://softwareg.com.au/blogs/computer-hardware/application-software-interacts-directly-with-the-computer-hardware?srsltid=AfmBOoq58Qe9M09PNgTaQg-j6nztrNzUn_szuvqXEOgc25eL6rkjIXbg

What does an OS abstract?

- CPU Resources
- Memory
- Disk resources
- Other hardware resources such as camera, printer, etc.

OS provides the mechanism to communicate with and utilize the resources!!

Fundamental Concepts of Operating Systems

Virtualization

1. OS gives every process the illusion that it has exclusive access to CPU
2. Every process feels that there is enough memory

Concurrency

1. Multiple processes can run at the same time without any problems
2. OS provides methods to resolve any issues via synchronization

Persistence

1. Disk needs to be managed and handled.
2. Handles interactions with the disk and performs storage management

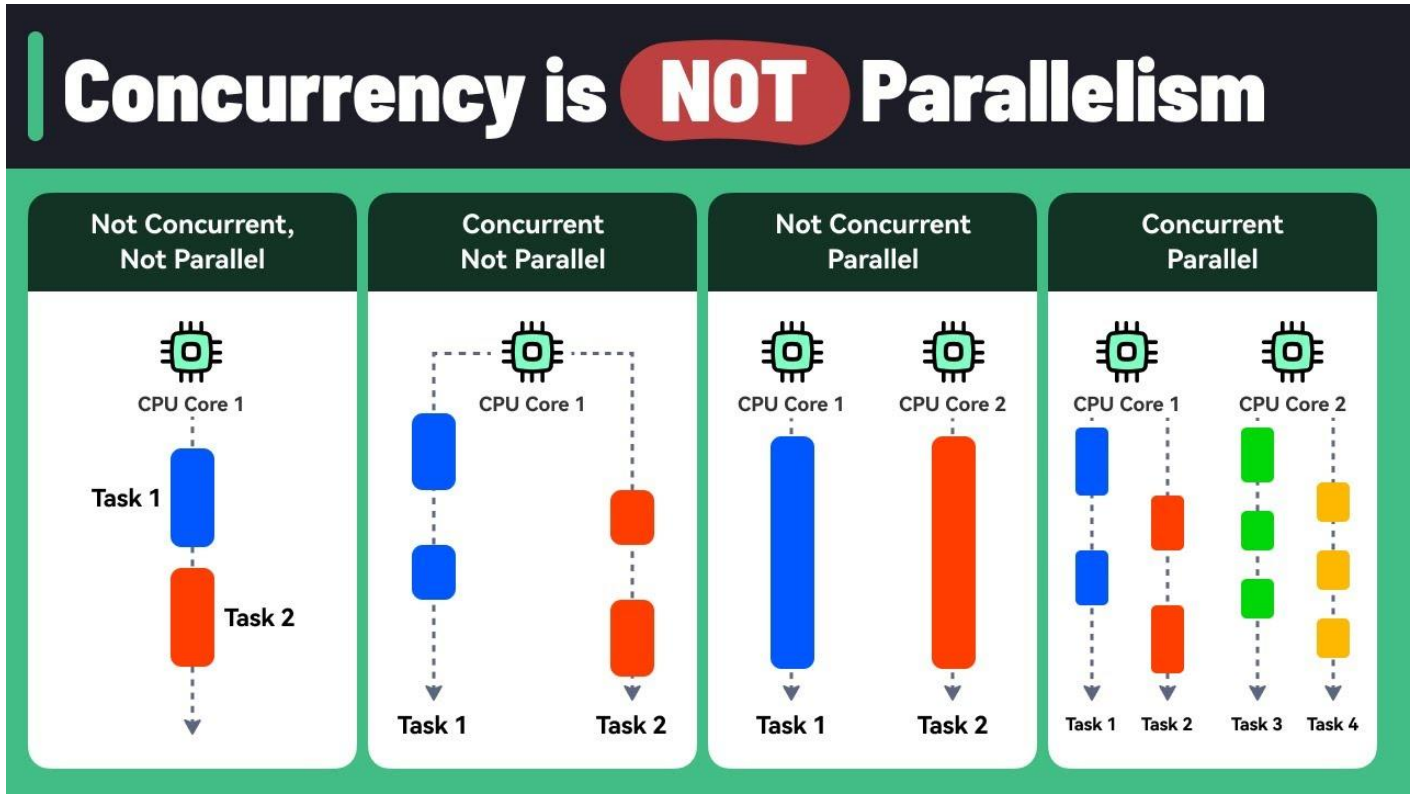
Process Virtualization

- Number of Processes in a system > Number of processors or CPUs in the system
- Virtualization allows all processes to get a share of the CPU resources → OS facilitates this!
- If two programs P1 and P2 want to run at a particular time, which should run? → A Policy of the OS determines this!

Memory Virtualization

- Every process requires memory
- More often,
  Memory requirement of active processes > Available memory
- How is this managed?

Concurrency



Source: https://cs.lmu.edu/~ray/notes/introconcurrency/

Persistence

- RAM is volatile
- Hardware and software required to store data persistently
  - Hardware: I/O devices such as hard drive, SSDs, etc.
  - Software:
    - File System manages the disk
    - File system responsible for managing the files created
    - Read and writes are handled by the file system which interacts with low level device drivers

Example – CPU Virtualization

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>
#include "common.h"

int
main(int argc, char *argv[])
{
    if (argc != 2) {
        fprintf(stderr, "usage: cpu <string>\n");
        exit(1);
    }
    char *str = argv[1];
    while (1) {
        Spin(1);
        printf("%s\n", str);
    }
    return 0;
}
```

Program

```
prompt> gcc -o cpu cpu.c -Wall
prompt> ./cpu "A"
A
A
A
A
^C
prompt>
```

Execution 1

```
prompt> ./cpu A & ./cpu B & ./cpu C & ./cpu D &
[1] 7353
[2] 7354
[3] 7355
[4] 7356
A
B
D
C
A
B
D
C
A
```

Execution 2

Example - Concurrency

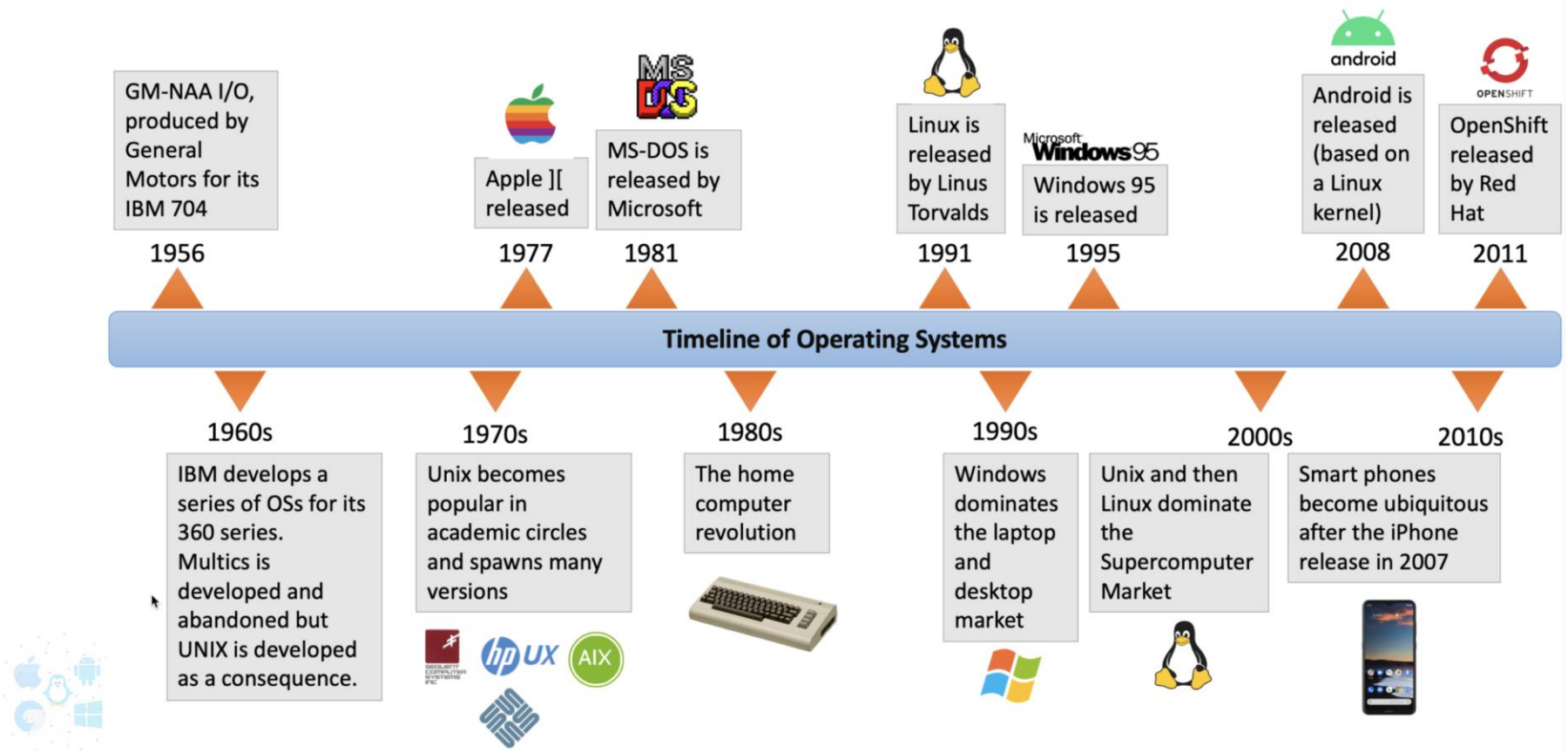
```c
#include <stdio.h>
#include <stdlib.h>
#include "common.h"
#include "common_threads.h"

volatile int counter = 0;
int loops;

void *worker(void *arg) {
    int i;
    for (i = 0; i < loops; i++) {
        counter++;
    }
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "usage: threads <value>\n");
        exit(1);
    }
    loops = atoi(argv[1]);
    pthread_t p1, p2;
    printf("Initial value : %d\n", counter);

    Pthread_create(&p1, NULL, worker, NULL);
    Pthread_create(&p2, NULL, worker, NULL);
    Pthread_join(p1, NULL);
    Pthread_join(p2, NULL);
    printf("Final value   : %d\n", counter);
    return 0;
}
```

```
prompt> ./threads 100000
Initial value : 0
Final value   : 143012    // huh??
prompt> ./threads 100000
Initial value : 0
Final value   : 137298    // what the??
```

Design Goals of OS

- Abstraction: Convenient and easy to use
- High Performance: Minimize overhead – Virtualization not at any cost
- Protection: Between applications and between OS and application
- Reliability: OS must continuously run without crashing
- Other goals: Energy efficiency, Security, Mobility

A Brief History of OS

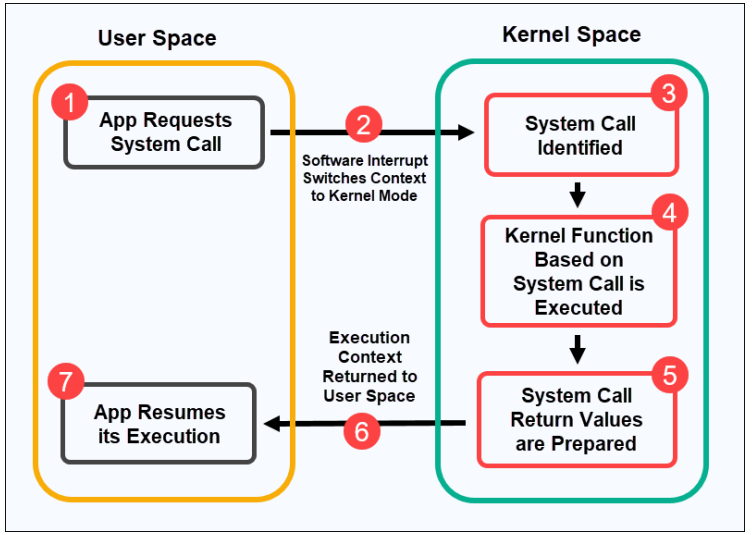


Early Operating Systems: Just Libraries!

- Just a set of libraries of commonly used functions
- Example: To avoid each programmer write low-level i/o, OS would provide APIs to do it.
- One program ran at a time and a human operator decided what programs ran in what order.
- Not interactive

OS more than a set of libraries!

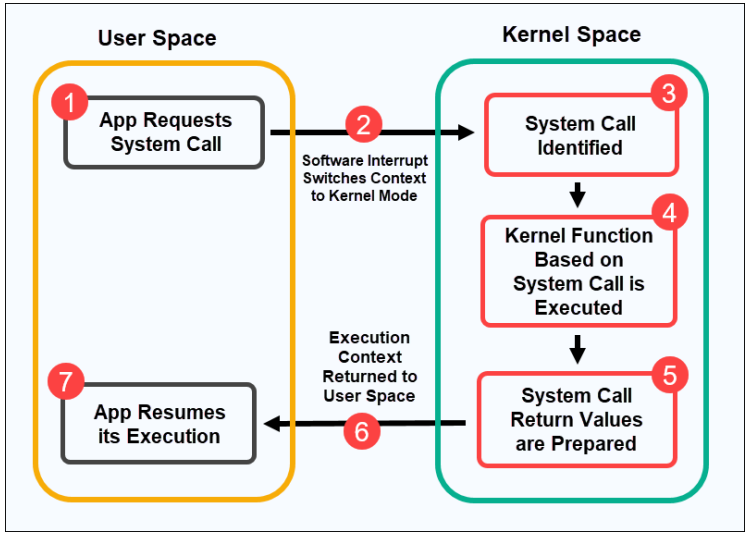
- Code run on behalf of OS is special → Should be treated differently
- What if any application could read from anywhere on the disk?
- **System Call** was invented → A special pair of hardware instructions and hardware state was added to make the OS function access more formal and controlled process.
- System call transfers control into the OS while raising the **hardware privilege level.**
- User applications run in **user mode** → hardware restricts what it can do.



Source: https://phoenixnap.com/kb/system-call

OS more than a set of libraries!

- System call initiated through special hardware instruction called a **trap**
- Hardware transfers control to a pre-specified **trap handler** and raises privilege level to **kernel mode**
- In kernel mode, OS has full access to the hardware of the system
- When OS completes the service → passes control back to the user via a special **return-from-trap** instruction
- Reverts to user mode and switches context

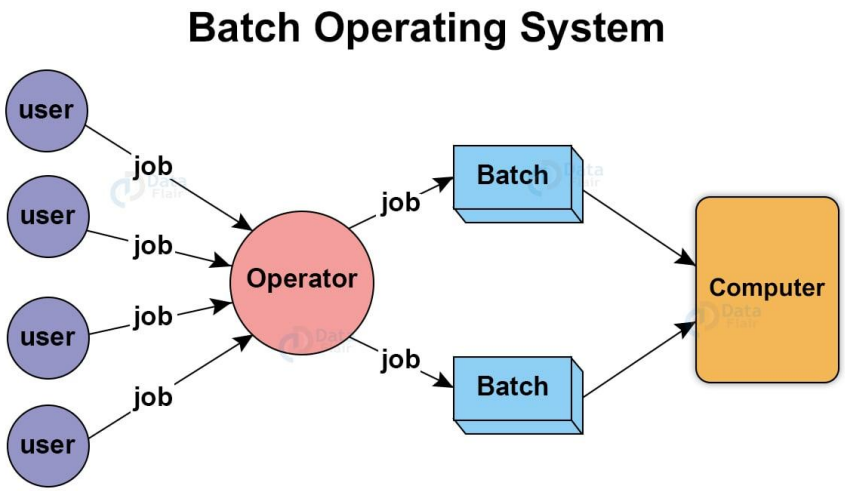


Source: https://phoenixnap.com/kb/system-call

Types of Operating Systems

Batch Operating System

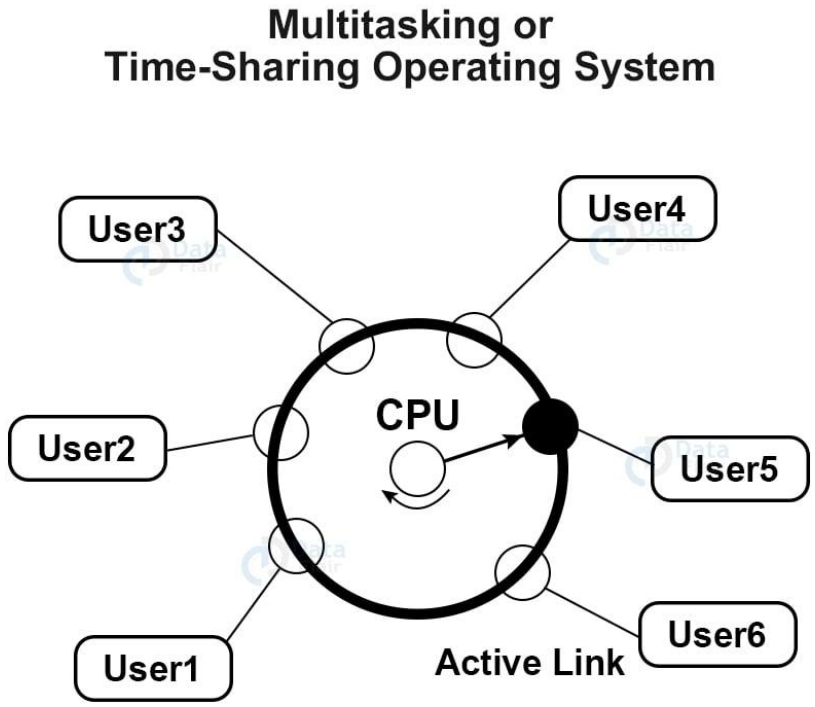
- Does not communicate or interact with the computer directly.
- Combines together similar types of jobs and runs them as a group.
- There is an operator involved.



Source: https://data-flair.training/blogs/types-of-operating-systems/

Multitasking or Time-Sharing OS

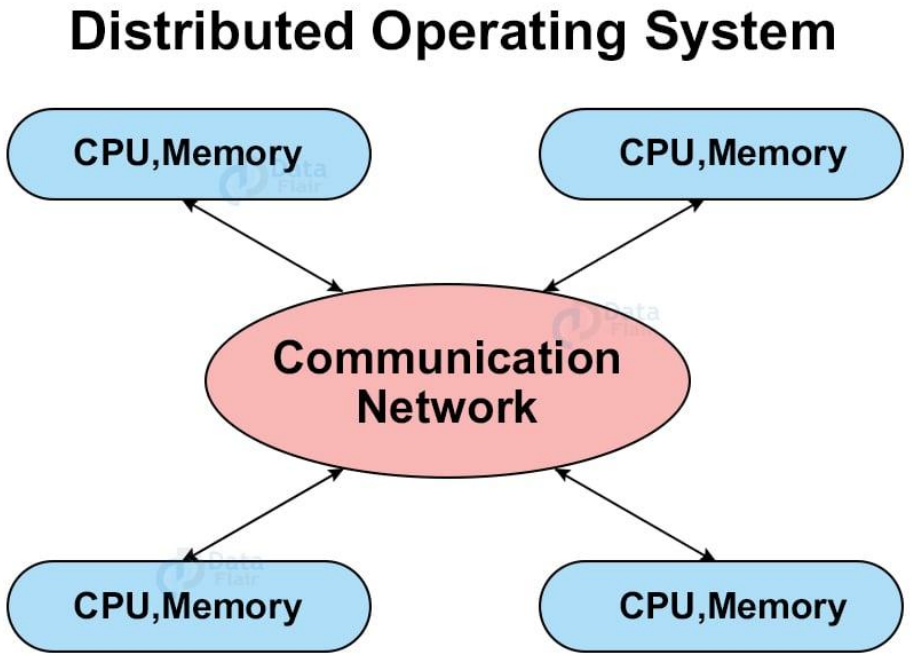
- Enables people who are in two different shells to use the OS at the same time.
- Processing time shared among multiple users
- Each task allotted a quantum of time to execute
- After the allotted quantum, OS switches to next task
- Example: UNIX, Multics



Source: https://data-flair.training/blogs/types-of-operating-systems/

Distributed Operating System

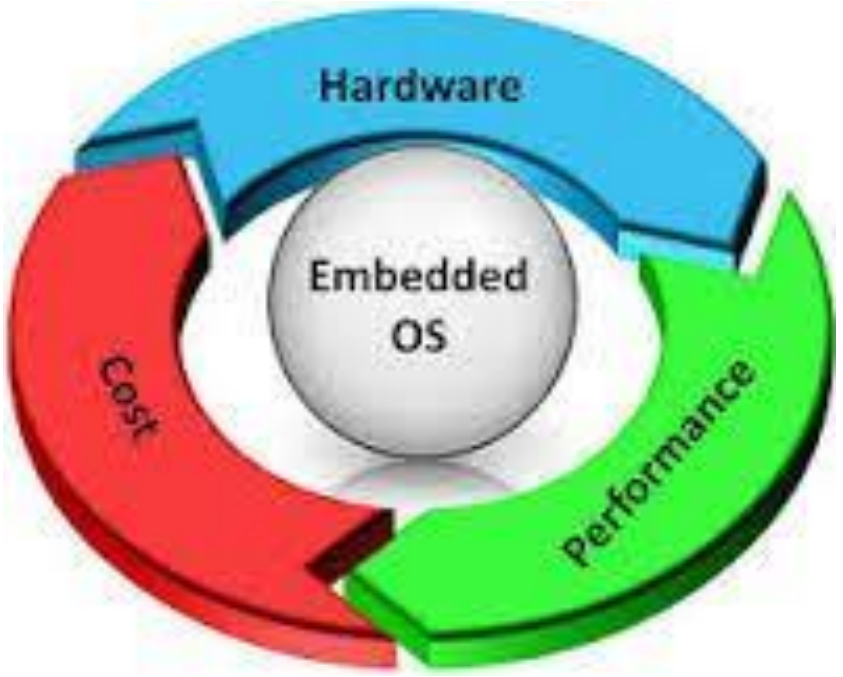
- Communication networks are used to interconnect various computers.
- System software over a collection of communicating and physically separate computation nodes.
- Enables resource sharing and is fault tolerant
- Example: LOCUS (Developed in 1980s at UCLA)



Source: https://data-flair.training/blogs/types-of-operating-systems/

Embedded Operating System

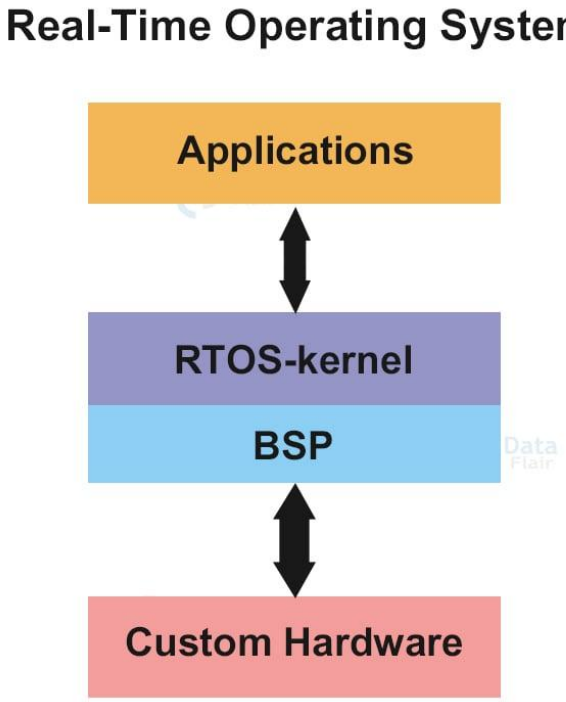
- Specialized OS designed to perform a particular task for a given device
- Designed to be compact, resource-efficient and most reliable



Source: https://itrelease.com/2016/12/advantages-disadvantages-embedded-operating-system/

Real-Time Operating System

- Required for real-time systems, which are subjected to real-time constraints.
- If the operation is not done within the time constraint, the system is most likely to fail.



Source: https://data-flair.training/blogs/types-of-operating-systems/

Thank You!!