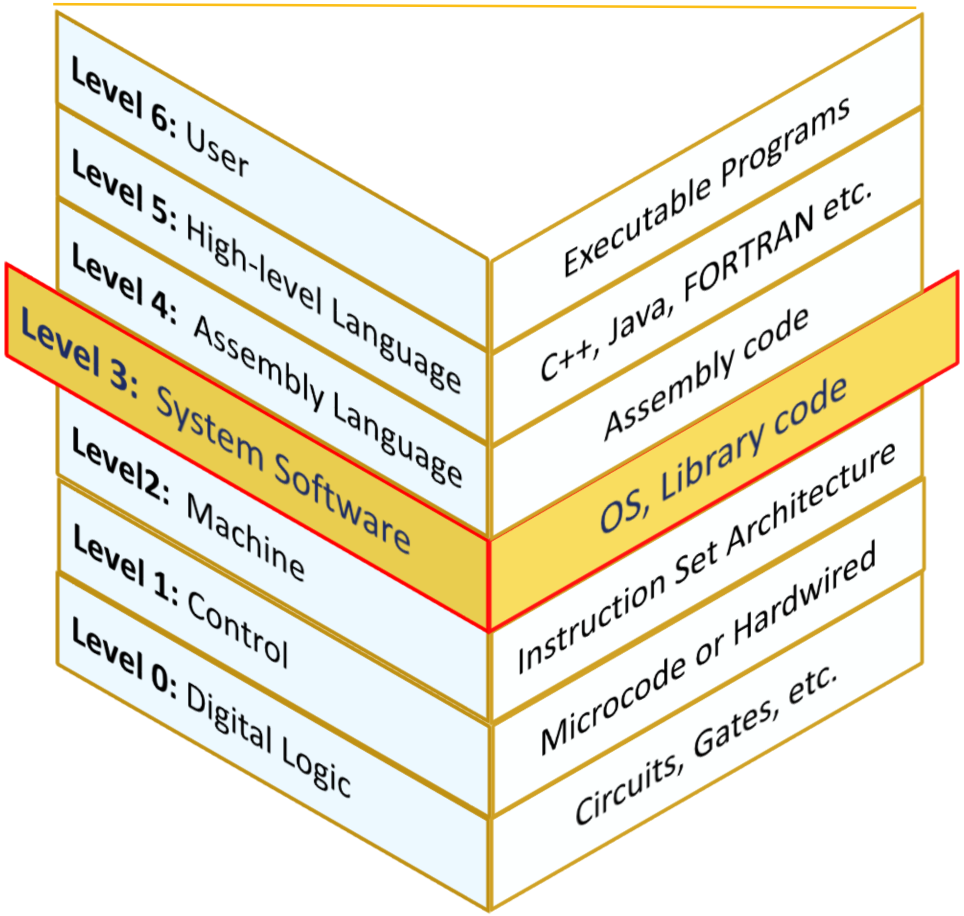
Introduction to Operating Systems

OS Goals,

Computer Systems: Batch, Multiprogramming, Timesharing

Computing Environments

Computer Level Hierarchy



Level 6: User — Executable Programs

Level 5: High-level Language — C++, Java, FORTRAN etc.

Level 4: Assembly Language — Assembly code

Level 3: System Software — OS, Library code

Level2: Machine — Instruction Set Architecture

Level 1: Control — Microcode or Hardwired

Level 0: Digital Logic — Circuits, Gates, etc.

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- OS is a **resource allocator**
  - Manages all resources (CPU, memory, I/O devices)
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Goals

- **Convenience**
  - for the user
  - for the programmer
- **Efficiency**
  - Maximize utilization of the machine’s resources
    - No idling
- Many other goals conceivable
  - **Robustness to errors** (Airplanes)
  - **Security** (Governments)
  - **Scalability** (Google)

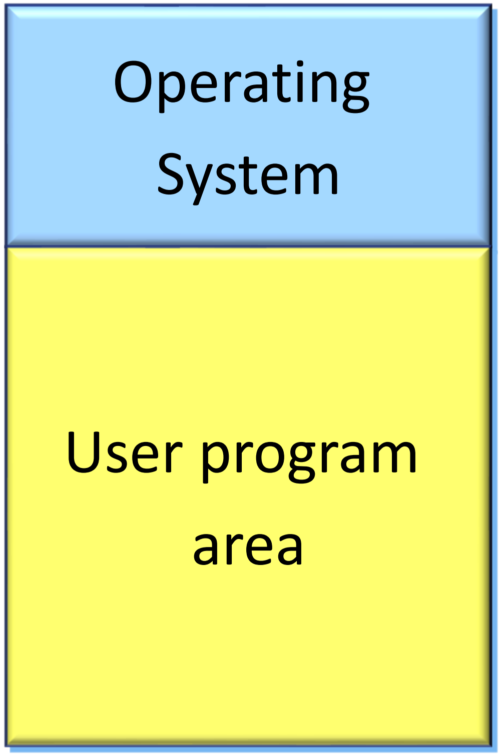
Computer Systems

- Batch systems
- Multiprogrammed systems
- Timesharing systems
- Embedded systems
- Distributed systems

Batch Systems

- Early computers:
  - Input devices: card readers and tape drives
  - Output devices: line printers, tape drives, and card punches
  - No direct interaction between the user and the computer system
  - The major task of the OS was to transfer the control from one job to the next.

Batch System: memory layout



Operating System

User program area

Just one job in memory at any single time!

Characteristics of process execution

- An I/O operation normally takes much more time than a CPU operation
- A process doesn’t need to use the CPU while it is doing I/O.
- Each process spends most of its time using I/O devices.

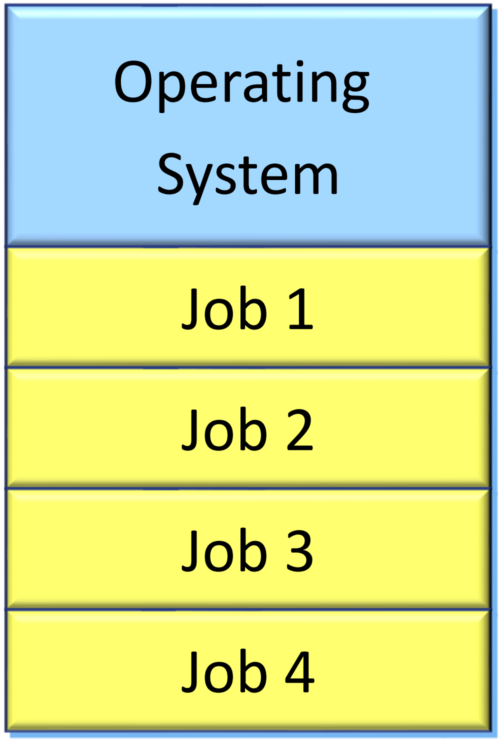
➔Batch Systems:

Wasteful! CPU idle when doing I/O.

Multiprogrammed Systems

- Goal: To keep the CPU as busy as possible
- The OS keeps several jobs in memory simultaneously.
    - **CPU scheduling**
    - OS picks and begins one of the jobs in memory
    - It switches to another job when the current job needs to wait (for an I/O),
    - When a job finishes waiting, it gets the CPU back.
- **Positive: Good resource utilization**
- **Negative: No user interaction**

Multiprogrammed System: memory layout



Operating System

Job 1

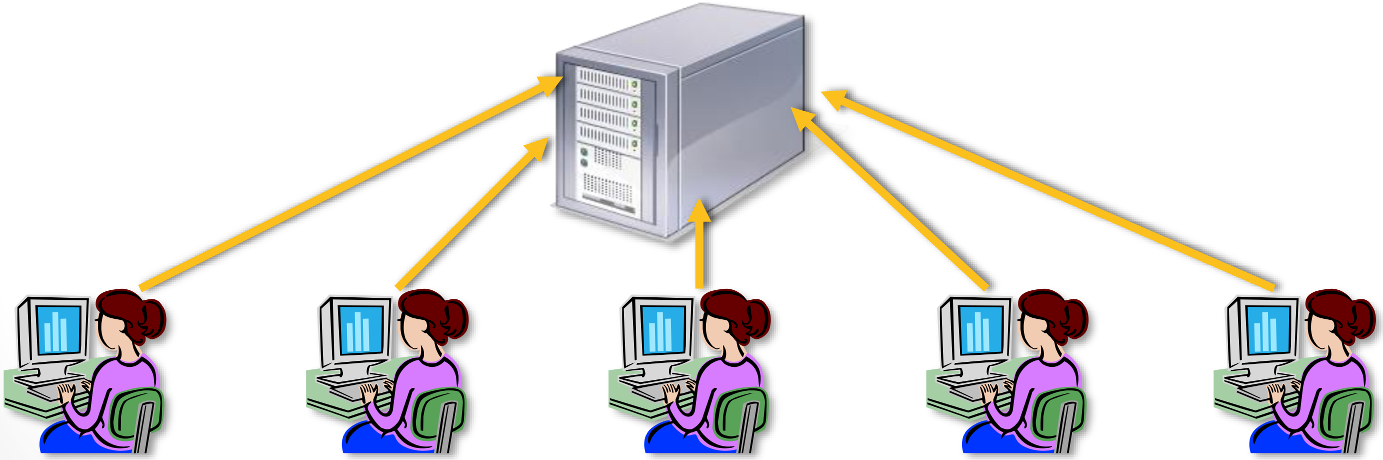
Job 2

Job 3

Job 4

Timesharing Systems

- A logical extension of multiprogramming.
- Also called multitasking
- Its original goal was to enable multiple users to interact with the computer system at the same time.



Timesharing Systems

- Support interactive processes.
  - OS switches very quickly between processes.
  - Interactive processes are given a higher **priority**.
  - Goal: Short response time

    ➔ Illusion of interactivity.

Timesharing Systems

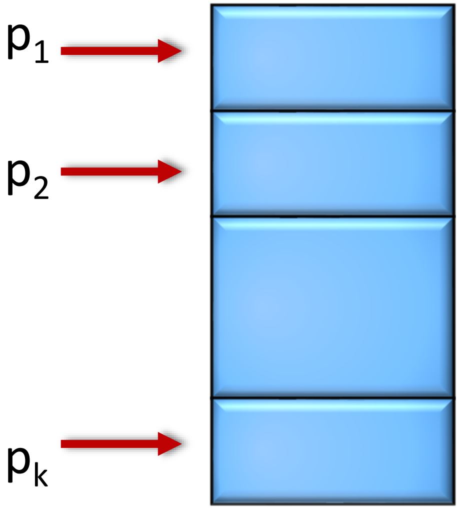
- More complex than multiprogrammed systems:
    - More complex CPU scheduling
        - To achieve short response time
        - What if many processes are ready to execute?
    - Memory management
        - What if process A tries to access process B’s memory?
        - What if not all processes fit in memory?

Timesharing and multiprogramming

- Most systems today use timesharing.
- Timesharing and multiprogramming are central themes of modern systems

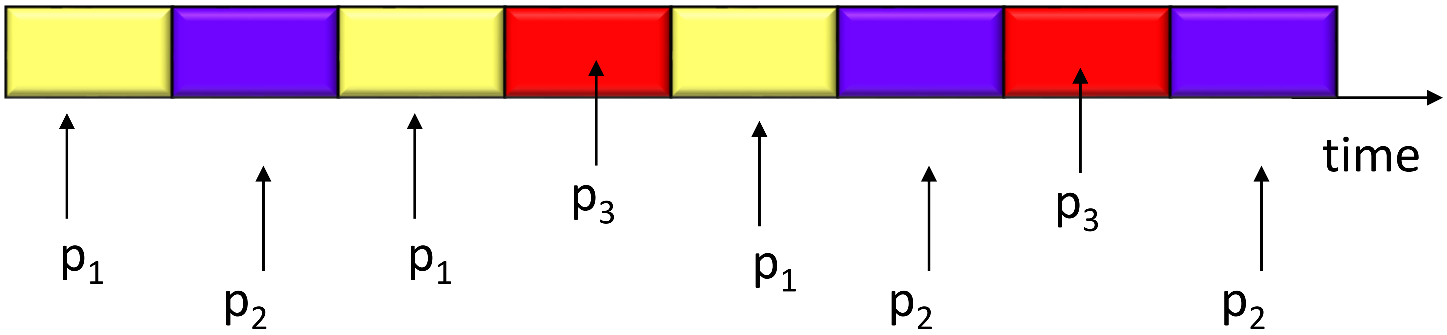
Space-multiplexed Sharing

A resource is divided into two or more distinct units, and then the individual parts are allocated to processes.



Time-multiplexed Sharing

- A process can use the entire resource exclusively for a period of time.



Timesharing

- Space-multiplexing for memory and time-multiplexing for CPU.
- Assume that there are N processes
  - The OS divides the physical memory into N different blocks, and allocates a block to each process.
  - A process that is loaded into its block of memory shares the CPU using time-multiplexed sharing.

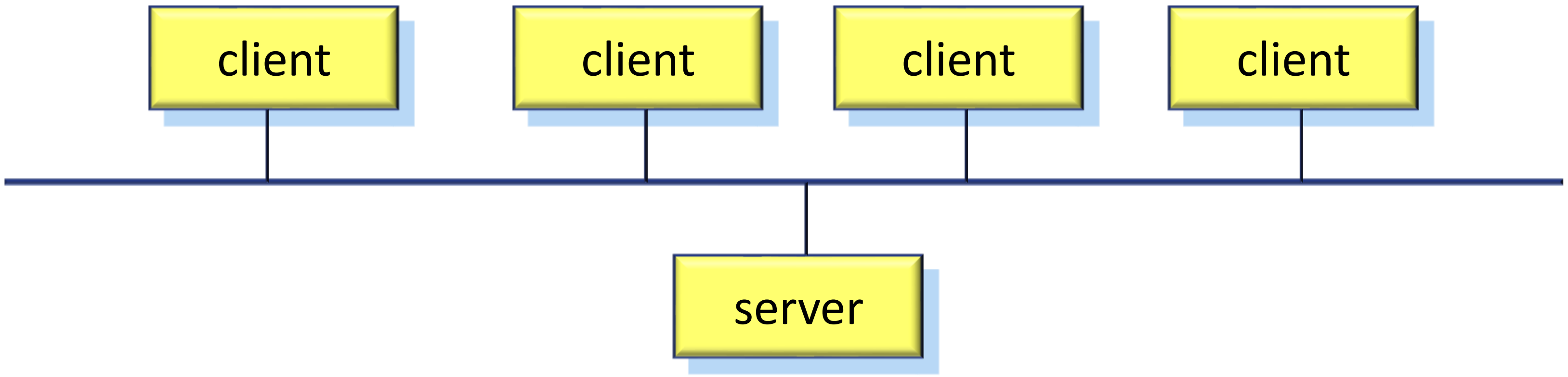
Embedded Systems

- A computer is embedded into a complex system.
- Goals of an embedded system:
  - Minimize the amount of memory and CPU cycles
  - Use as little electrical power as possible.
  - Hard Real-time constraint
    - **Guarantee** a minimum response time

Distributed Systems

- In the mid-1990s, network connectivity became an essential component of a computer system
  - Nearly all modern PCs and workstations are capable of running a web browser
  - OSs such as Windows, Mac OS and UNIX/Linux now include software, such as a **TCP/IP stack**, that enables a computer to access the Internet.
  - Windows even bundled a web browser as part of the OS (got into trouble!)

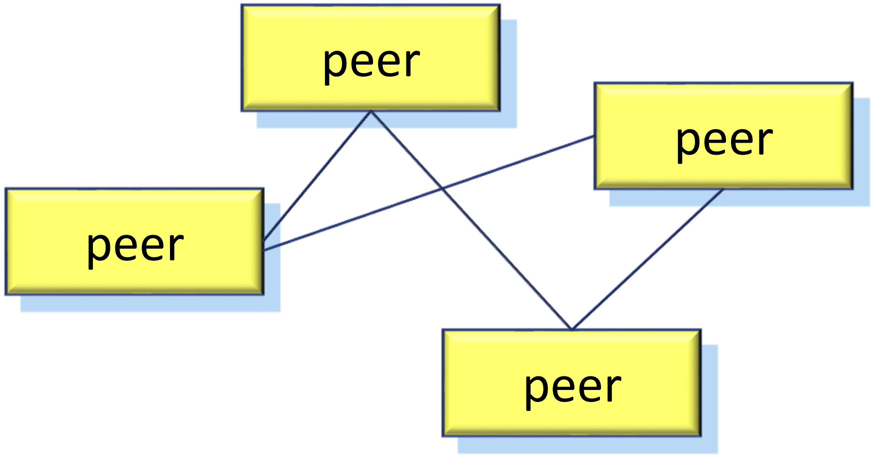
Client-Server Systems



A client sends requests to perform an action; a server executes the action and sends back results to the client.

Peer-to-Peer Systems

- In a client-server system, the server could be a bottleneck.
- In a peer-to-peer system, all nodes within the system are considered peers, and each may act as either a client or a server.



Summary

- Operating System Goals
- Computer Systems
  - Batch systems
  - Multiprogrammed systems
  - Timesharing systems
- Resource sharing
  - Space-multiplexed sharing
  - Time-multiplexed sharing
- Computing Environments
  - Embedded
  - Distributed
  - Client-Server
  - Peer-to-Peer