CS3.304.M22

Advanced Operating Systems

Lecture # 01

Manish Shrivastava

LTRC, IIIT Hyderabad

Course Content

- History of Operating Systems
- Processes and OS Abstractions
- OS APIs, Interrupts and system calls
- Introduction to the Linux Kernel
- Compiling the kernel
- Module programming
- Writing your own system calls
- Overview of kernel startup and initialization
- Kernel Debugging Techniques
- Interrupts - PICs, APICs, exceptions (traps) and hard interrupts, IDTs
- Address Spaces and Loading, Virtual Memory, Memory allocators,
- Overview of memory spaces: logical segmentation, linear virtual, actual physical
- Detecting BIOS-provided physical RAM map, paging, buddy system, setting up page directories (global, upper, middle), tables and PTEs
- (N)UMA, nodes, zone, memory types, Setting up buddy system
- Allocating contiguous pages from buddy system
- Setting up slabs for small memory objects
- CPU Scheduling, Threads, Process - structures, organization, initialization
- Concurrent Programming, Locking, Deadlocks
- Structures: thread union, thread info, stack, task, and thread struct
- Creating kernel threads, using kthread, Kernel process scheduling
- Scheduling processes with red-black tree, process switching, Context switches, Switching to suspended process
- Linux File Systems and Disk Scheduling.

Reference Material

- **PREFERRED TEXT BOOKS:**
  - Thomas Anderson and Michael Dahlin
  - ***Operating Systems: Principles and Practice, 2nd Edition***
  - Recursive books (August 21, 2014),
  - ISBN: 0985673524

  - Daniel P. Bovet & Marco Cesati
  - ***Understanding the Linux Kernel (3rd edition)***
  - O'Reilly & Associates, November 2005.
  - ISBN: 0596005652

- **REFERENCE BOOKS:**
  - Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau
  - ***Operating Systems: Three Easy Pieces***
  - Arpaci-Dusseau Books
  - August, 2018 (Version 1.00)

  - Jonathan Corbet; Alessandro Rubini; Greg Kroah-Hartman
  - ***Linux Device Drivers (3rd edition)***
  - O'Reilly & Associates, February 2005.
  - ISBN-13: 978-0-596-00590-0

  - Robert Love
  - ***Linux Kernel Development*** **(3nd Edition)**
  - Addison-Wesley Professional, 2010.
  - ISBN: 0672329468

  - Ellen Siever, Stephen Figgins, Robert Love, and Arnold Robbins
  - ***Linux in a Nutshell, 6th Edition***
  - O'Reilly & Associates, September 2009.
  - ISBN: 978-0-596-15448-6

Expected Outcome

- Do not revisit but **re-establish** your basics
    - Go deeper
    - Better understand
    - Be more up-to-date
- We will ensure that both ***theoretical*** as well as ***practical*** aspects are learnt simultaneously.
- The project deliverables are expected to working code/prototype.

Grading Scheme

- Relative Grading.

- Detailed Grading Breakup (~ 36% Exams/Quiz + 64% Practical Learning):
  - 34% Course Project
  - 30% Assignments (6×5)

  - 10% Mid-semester Exams
  - 20% Final Exam
  - 06% Quiz/Others

- Zero marks in assignment if a copying is detected
- Cooperation with TA's is expected
- Zero tolerance from institute's attendance policy

Lecture 01: Plan

- System
- Characteristics
- System Concepts
- System Components
- Recap of historical OS efforts

What is an Operating System?

Abstracts the computer hardware

- Hides the messy details of the underlying hardware
- Present users with an easy to use resource abstraction
- Extends or virtualizes the underlying machine

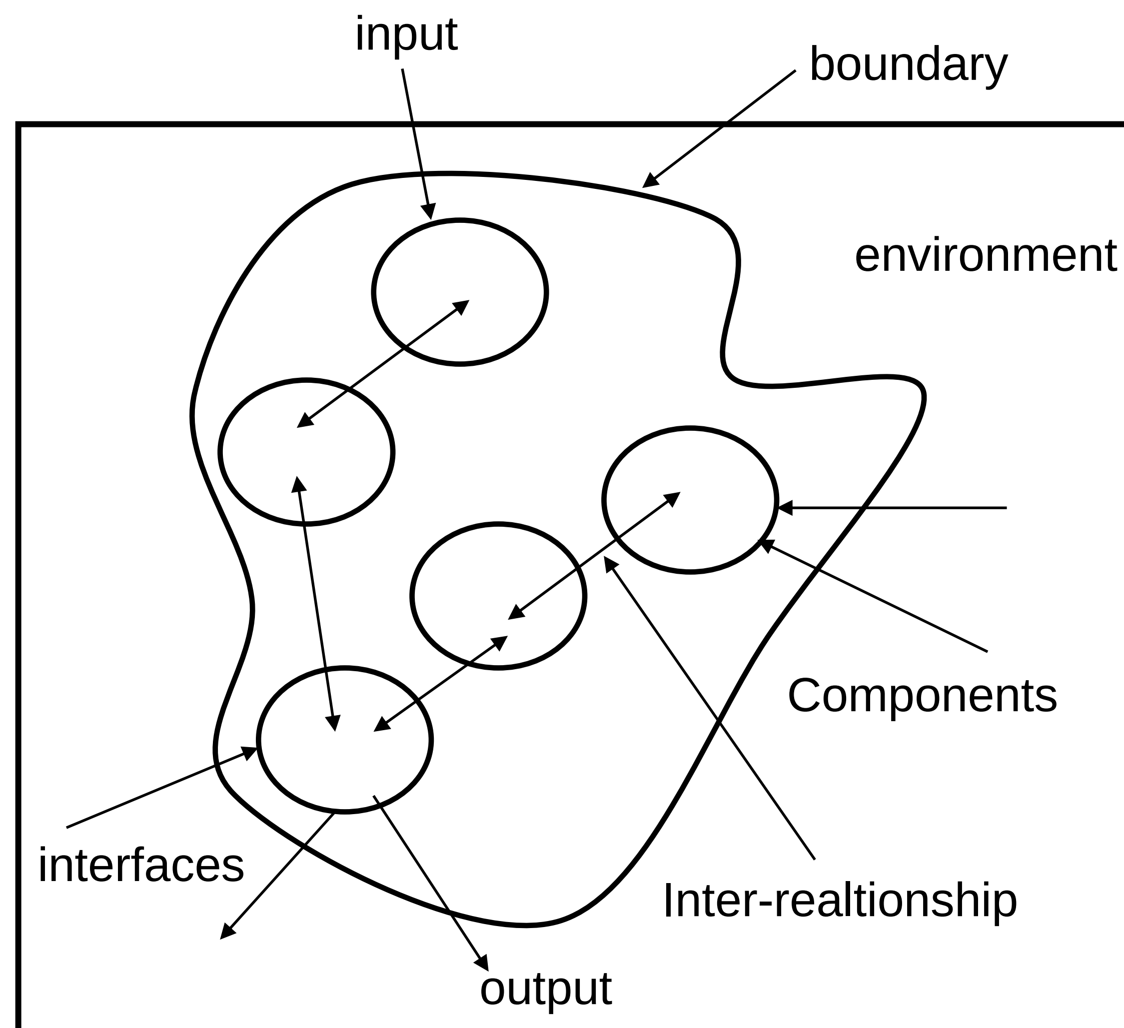
Manages the resources

- Processors, memory, timers, disks, mice, network interfaces, printers, displays, …
- Allows multiple users and programs to share the resources and coordinates the sharing

System

Characteristics of a system:

- Components
- Inter-related components
- A boundary
- A purpose
- An environment
- Interfaces
- Input
- Output
- Constraints



Characteristics

- Components:
  - A system is made up of components.
  - A component is either an irreducible part or aggregation of parts that make-up a system. A component is also called a sub-system.

- Interrelated Components:
  - The components of interrelated
  - Dependence of one subsystem on one or more subsystems.

Characteristics

- Boundary (Scope):
  - A system has a boundary, within which all of its components are contained and which establishes the limits of a separating the system from other systems.

- Purpose
  - The overall goal of function of a system. The system’s reason for existing.

Characteristics

- Environment
  - Everything external to the system that interacts with the system.
- Interface
  - Point of contact where a system meets its environment or subsystems meet each other.
- Constraint:
  - A limit what a system can accomplish: Capacity, speed or capabilities.

Characteristics

- Input
  - Whatever a system takes from its environment in order to fulfill its purpose.

- Output:
  - Whatever a system returns to its environment in order to fulfill its purpose.

Important System Concepts

- Decomposition
- Modularity
- Coupling
- Cohesion

Decomposition

- Divide and Conquer Strategy
- It deals with being able to break down a system into its components.
- Decomposition results in smaller and less complex pieces that are easier to understand than larger, complex pieces.
- Decomposing a system also allows to focus on one particular part of a system, making easier to think of how to modify that part independently of the entire system.

Modularity

- Modularity refers to dividing a system up into chunks or modules of a relatively uniform size.

- You can replace or add any other module (or a component) without effecting the rest of the system.

- It is a design strategy in which system is composed of relatively small and autonomous routines fit together.

Coupling

- Coupling is the extent to which subsystems are dependent on each other.
- Subsystems should be as independent as possible.
- If a subsystem fails and other subsystems are highly dependent on it, the others will either fail themselves or have problems in functioning.

Cohesion

- The extent to which a system or a subsystem performs a single function.

Operating System (OS)

- The operating part of a tool is called as operating system of that tool.
- The purpose of operating system is to facilitate the operation of the underlying machine or tool.
- For a machine, the OS abstracts the machine part in terms of simple services by hiding the details of the machine.
- The OS can provide services to users or other subsystems.
- Examples of typical operating systems:
  - Car operating system, Telephone operating system, TV operating system and so on.

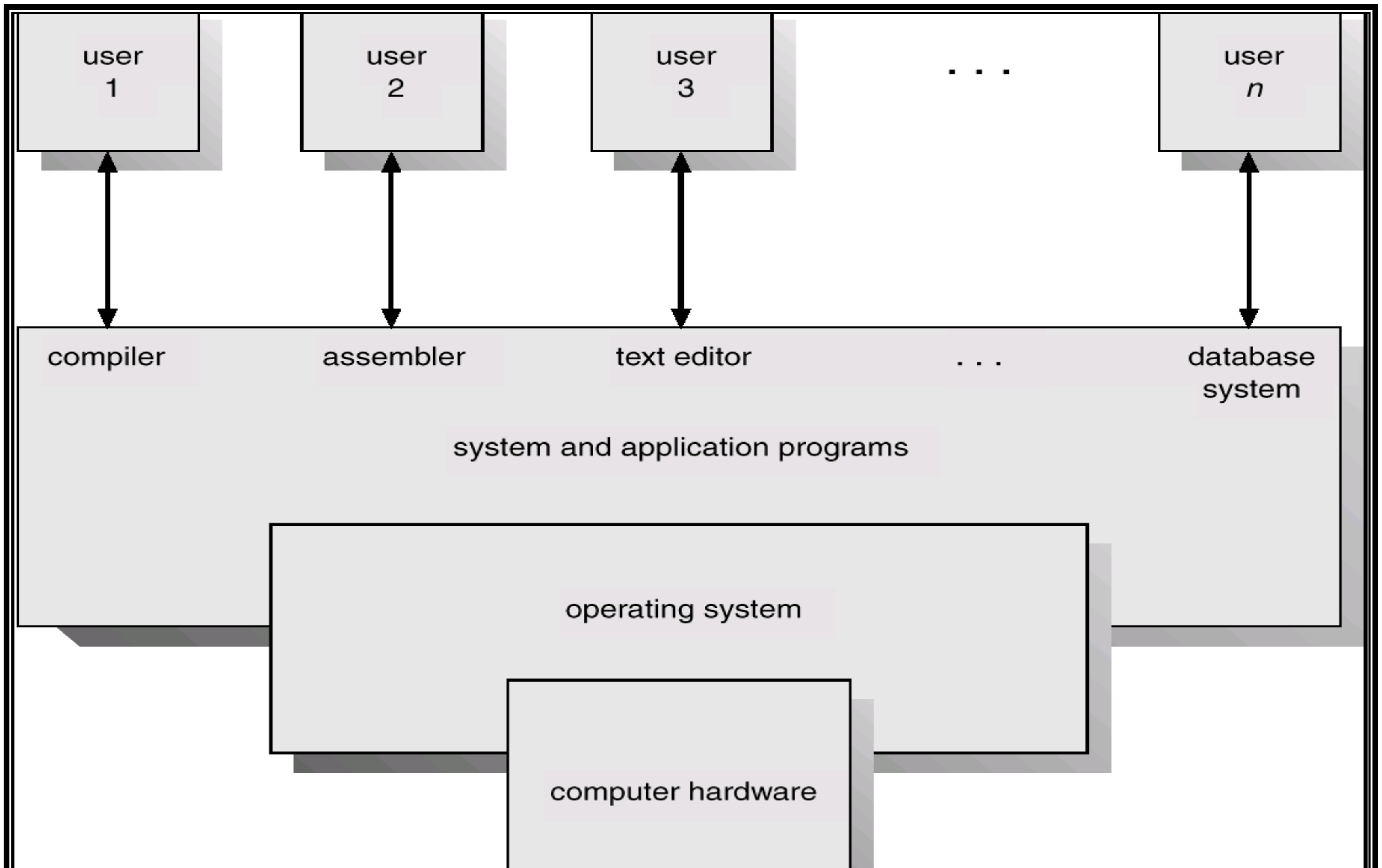
Computer Operating System

- A computer is also a tool that contains machine part and operating part.
- The operating part of a computer is called Computer Operating System.
- For a computer, the operating system abstracts the underlying hardware in terms of simple services by hiding the details of the hardware. The OS can provide services to users or other subsystems.
- Examples of Computer operating systems:
  - WINDOWS, Macintosh, UNIX, SOLARIS, LINUX and so on.
- In the rest of this course, operating system means computer operating system.

Computer OS components

- **Hardware** – provides basic computing resources (CPU, memory, I/O devices).
- **Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users.
- **Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
- **Users** (people, machines, other computers).

Computer OS components



Other OS Definitions

- **Resource allocator** – manages and allocates resources.
  - Resources: CPU time, Memory Space, file storage space, I/O devices and son on.
- **Control program** – controls the execution of user programs and operations of I/O devices .
- **Kernel** – the one program running at all times (all else being application programs).
- The two goals, efficiency and convenience are sometimes contradictory.
- Much of OS theory concentrates on optimal use of resources.

Objectives

- The main objective is to understand the operational part of any computer.
- Understanding the general principles of OS design.
  - Focus on general-purpose, multi-user, uni-processor systems.
  - Emphasis on widely applicable concepts rather than any specific features of any specific OS.
- Understanding problems, solutions and design choices.
- Understanding the structure of specific OSs: UNIX, LINUX, WINDOWS

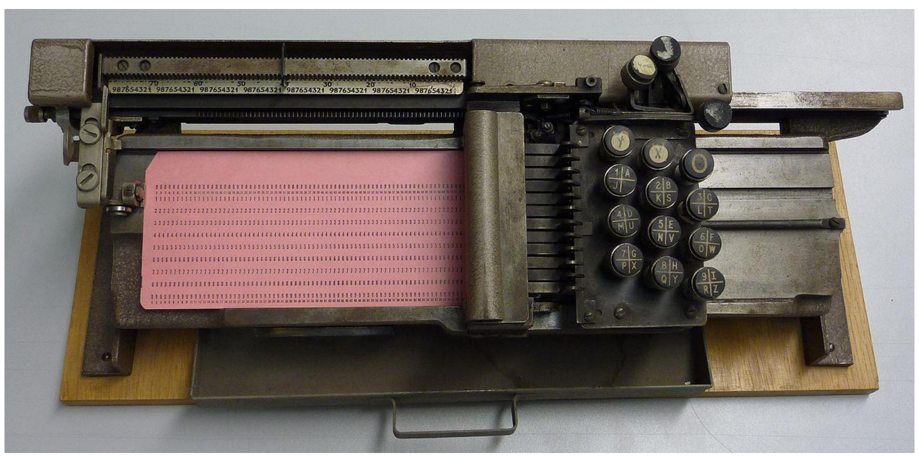
Early Systems (Serial Processing)

- **1940-50:**
  - The programmer interacted directly with the computer hardware.
  - Display light, switches, printer, card reader.
  - No OS. Error is displayed through lights.

- **Problems:**
  - Scheduling → Users spend lots of time at the computer.
    - Signup sheet was used.
  - Job Setup time
    - Loading and compiling
      - Mounting and Un-mounting of tapes
      - Setting up of card desks
  - Libraries of functions, linkers, loaders, debuggers, and I/O driver routines were available for all the users.

Early Systems

- Early computers were (physically) large machines run from a console.
- The programmer would operate the program directly from the console.
  - The program is loaded to the memory from panel of switches, paper tape, and from punched cards.
- As time went on, additional software and hardware were developed.
  - Card readers, line printers, and magnetic tape became common place.
  - Libraries, loaders, and common functions were created.
    - Software reusability.



IBM029

Early Systems

- The setup time was a real problem
- CPU is idle while tapes are being mounted or the programmer was operating the console.
- In the early days, few computers were available and they were expensive (millions of dollars).
  - operational costs: power, cooling, programmers.
- Main question:

How to increase the utilization of CPU ?

Early Systems

- The solution was two fold.
- First, a professional computer operator was hired.
  - Once the program was finished, the operator could start next job.
  - The operator sets up the job, produces the dump, and starts the next job.
  - The set up time was reduced due to operator's experience.
- Second, jobs with similar needs were batched together and run through the computer as a group.
  - For example, if there is a FORTRAN job, COBOL job, and FORTRAN job, two FORTRAN jobs were batched together.
- However, during transition time CPU sat idle.
- Automatic job sequencing to avoid CPU idle time.
  - A first rudimentary OS was created
  - A small program called a **resident monitor** was developed.
  - The resident monitor always resided in memory.

Simple Batch Systems

- In serial systems
  - Machines were very expensive
  - Wasting time was not acceptable.
- To improve usage, the concept of batch OS was developed.
- The main idea is the use of software known as monitor.
  - The user no longer has access to machine.
- The user submits the job (tape) to the operator.
- The operator batches the jobs together sequentially, places entire batch as an input device for use by the computer.

Features of Batch System

- The batch OS is simply a program.
- It relies on the ability of the processor to fetch instructions from various portions of main memory to seize and relinquish control.
- Hardware features:
  - **Memory protection**: While the user program is running, it must not alter the memory area containing the monitor.
  - If such is the case the processor hardware should detect the error and transfer control to monitor.
  - **Timer:** A timer is used to prevent the single job from monopolizing the system

Features of Batch System

- Hardware Features
  - **Privileged instructions**
    - Contains instructions that are only executed by monitor.
    - I/O instructions
    - If a program encounters them the control shifts through monitor.
  - **Interrupts:** It gives OS more flexibility.
    - Relinquishing control and regain control
- With batch OS the machine time alters between execution of user programs and execution of monitor.
- Two overheads
  - Machine time is consumed by the monitor.
  - Memory is consumed by the monitor.
- Still, they improved the performance over serial systems.

Problem with Batch System

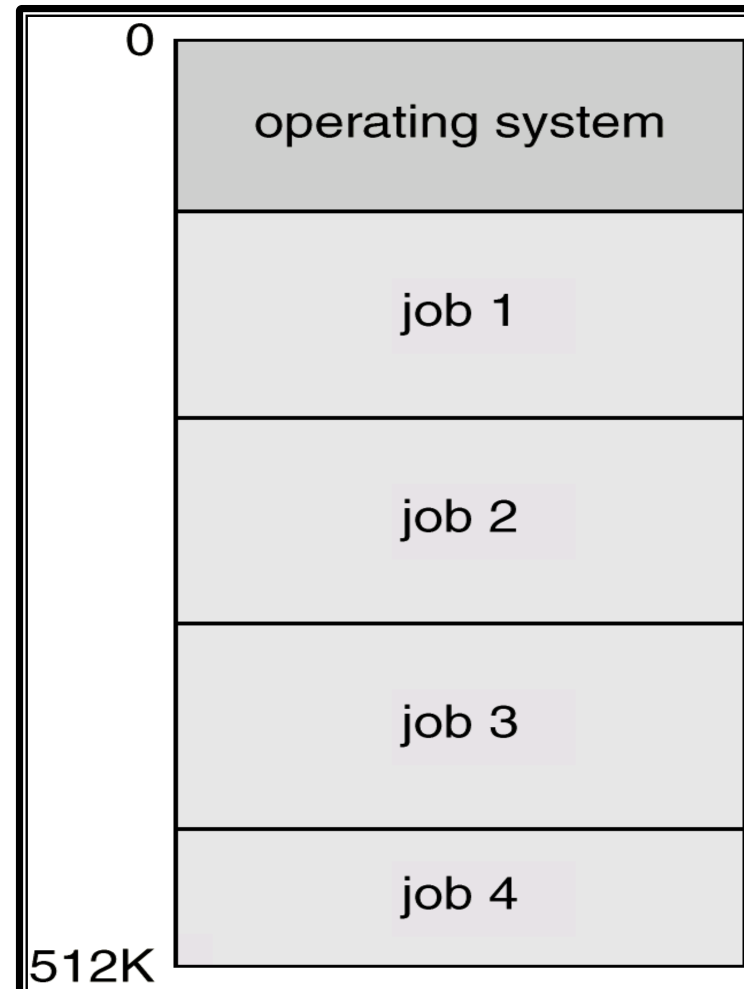
- CPU is idle
- Speed of mechanical devices is very slower than those of electronic devices.
- CPU works in a microsecond range
  - Thousands of instructions/second
- A card reader may read 1200 cards per minute (20 cards per second)
- CPU speed has increased at a faster rate.
- Tape technology improved the performance little-bit.
- Main perceived problem
  - Turn-around time: upto two days
  - CPU often underutilized
    - Most of the time was spent reading and writing from tape.

Multi-programmed Batched Systems (1960s) (or Multi tasking)

- If CPU is executing a job and requires a tape to be mounted
  - In a non multi-programmed system
    - CPU sits idle.
  - In a Multi-programmed system
    - CPU takes up another job.
- **Multiprogramming is the first instance when the OS started taking decisions.**
- Job scheduling is done by OS.
- Having several programs in the memory requires memory management.

Multi-programmed Batched Systems (1960s) (or Multi tasking)

- A single user can not keep either CPU or I/O busy.
- Multiprogramming increases CPU utilization by organizing jobs such that the CPU always has one to execute.
- The OS keeps several jobs in memory at a time and CPU is multiplexed among them



OS Requirements (60s)

- OS Research in 60s
  - MULTICS at MIT
  - Atlas (spooling, demand paging) at Manchester Univ.
  - Multiprogramming
    - Memory allocation and protection
    - I/O operations were responsibility of OS.
  - Interactive systems
    - Scheduling issues
    - Swapping and virtual memory.
  - Users wanted permanent files
    - Hierarchical directory systems.
  - Increased in size and complexity were not well understood
    - IBM: OS/360

UNIX (early 1970s)

- Originally developed at Bell labs for the PDP-7
  - Ken Thomson
  - Dennis Ritchie
- Smaller & Simpler
  - Process spawn and control
    - Each command creates a new process
  - Simple inter-process communication
  - Command interpreter not built in: runs as a user process
  - Files were streams of bytes.
  - Hierarchical file system
- Advantages
  - Written in a high-level language
  - Distributed in source form
  - Powerful OS primitives on an inexpensive platform

Personal Computers (1980s)

- Originally
  - Single user
  - Simplified OSs
    - No memory protection
    - MS-DOS
- Now run sophisticated OSs
  - Windows NT, Linux

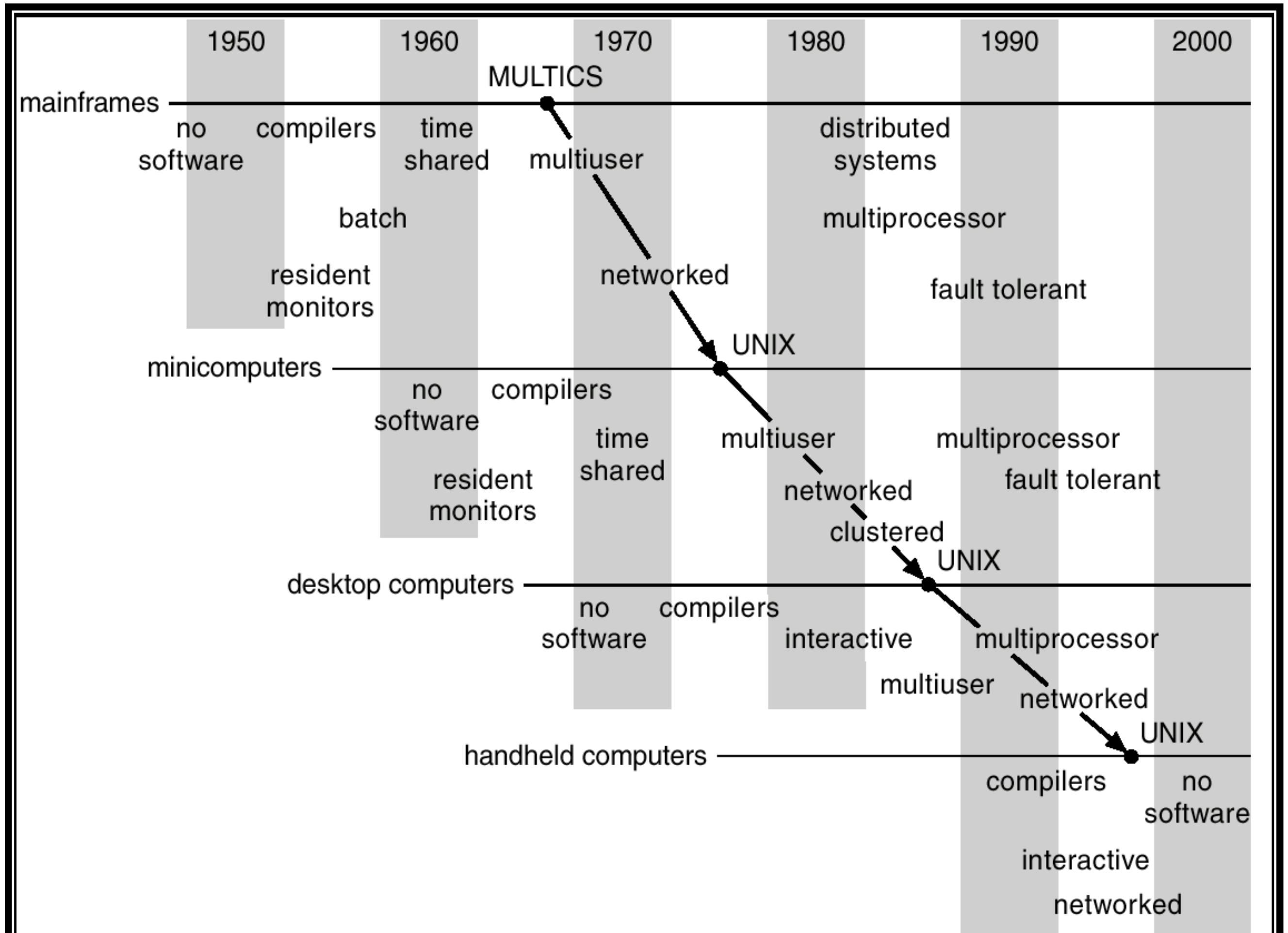
Networks of workstations (1990s)

- High-speed network connections
- Local & world-wide
- Client-server systems
  - File systems
  - Remote windowing systems
- Support a variety of node OSs
  - Unix, Windows XP, OS/2

Future

- Distributed systems
  - Network is invisible
- Micro-kernel and extensible OSs
  - Supports multiple OS flavors
    - E.g., Mach, Amoeba, WINDOWS XP
- Embedded services and network computers
  - Computer runs a very thin OS (Java Virtual machine).

Migration of Operating-System Concepts and Features

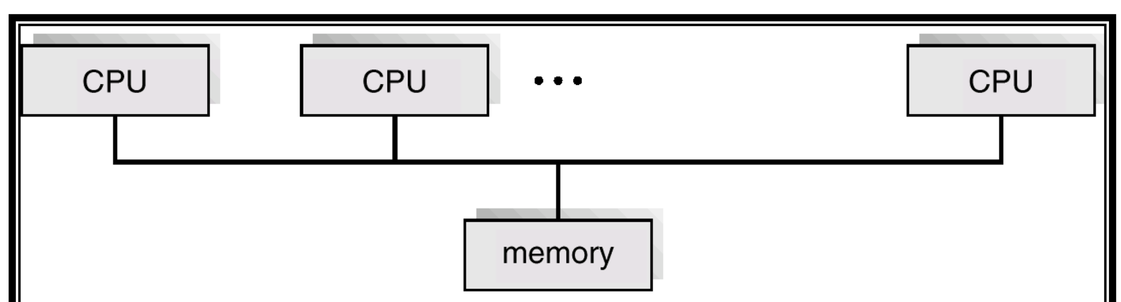


Multiprocessor Systems

- Multiprocessor systems with more than one CPU in close communication.
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of multiprocessor system:
  - Increased *throughput:* more processors more work
  - Economical: more computing in same box
  - Increased reliability
    - Graceful degradation & Fault tolerance: failure of one processor will not halt the system. Service is proportional to the level of surviving hardware.

Parallel Systems

- *Symmetric multiprocessing (SMP)*
  - Each processor runs and identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocated work to slave processors.
  - More common in extremely large systems



SMP Architecture

Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system*
  - each processor has its own local memory;
  - processors communicate with one another through various communications lines such as high-speed buses or telephone lines.
- Advantages of distributed systems
  - Resources Sharing
  - Computation speed up/load sharing
  - Reliability
  - Communications

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.

- Well-defined fixed-time constraints.
  - A process must complete within the defined constraints or system will fail.

- Real-Time systems may be either *hard* or *soft* real-time.

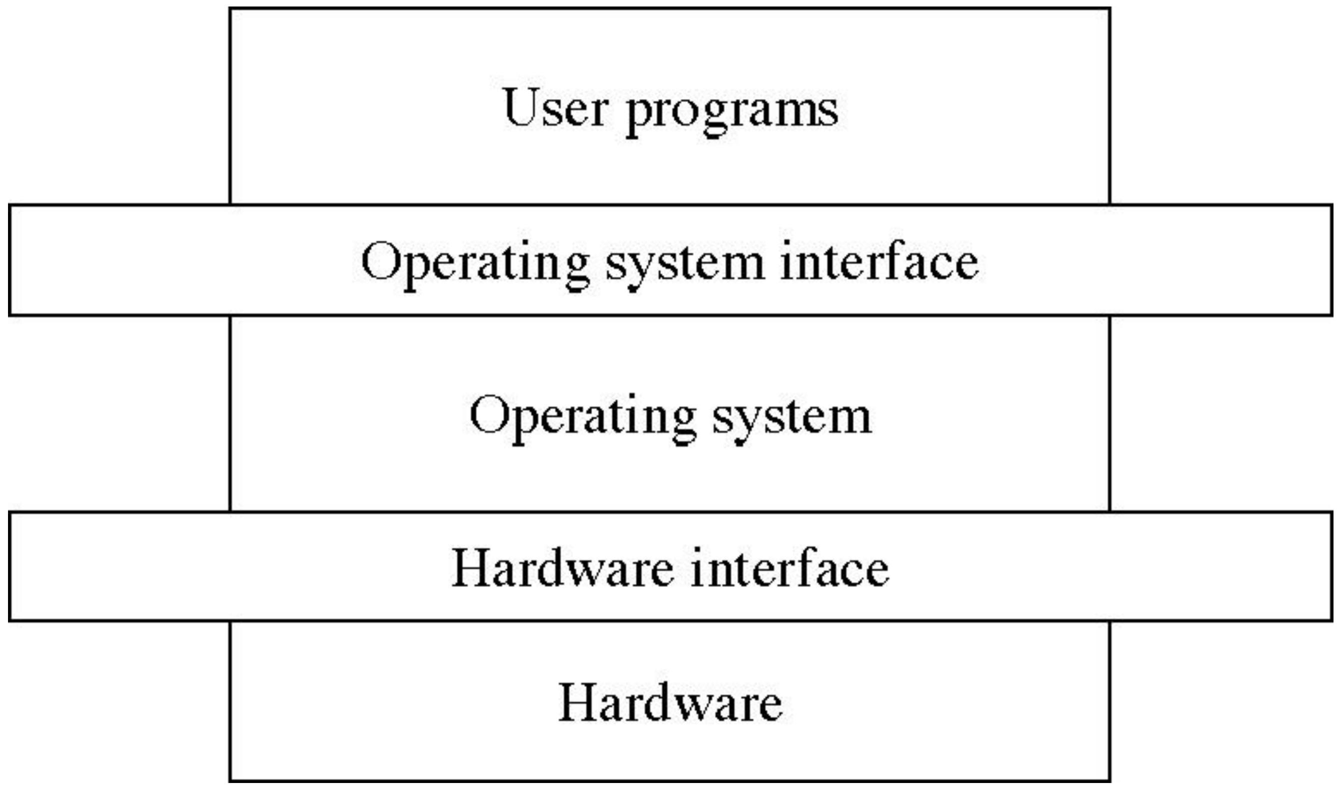
Real-Time Systems

- Hard real-time:
  - Guarantees that critical tasks complete within time.
  - All the delays in the system are bounded.
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time
  - Critical time tasks gets priority over other tasks, and retails that priority until it completes.
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Concept of Virtual Computer

- Multilevel implementation
  - also called layered
- Resources
  - Hardware: provided to the OS
  - Logical (virtual): created by the OS
- Resource management
  - transformation
  - multiplexing
    - time and space

Levels in a computer system



Design: Two-level Implementation

- Two-level implementation
  - Lower level is a problem-specific language
  - Upper level solves the problem at hand
  - Lower level is reusable

- In operating systems
  - *mechanism*: lower level of basic functions, does not change
  - *policy*: upper level policy decisions, easy to change and experiment

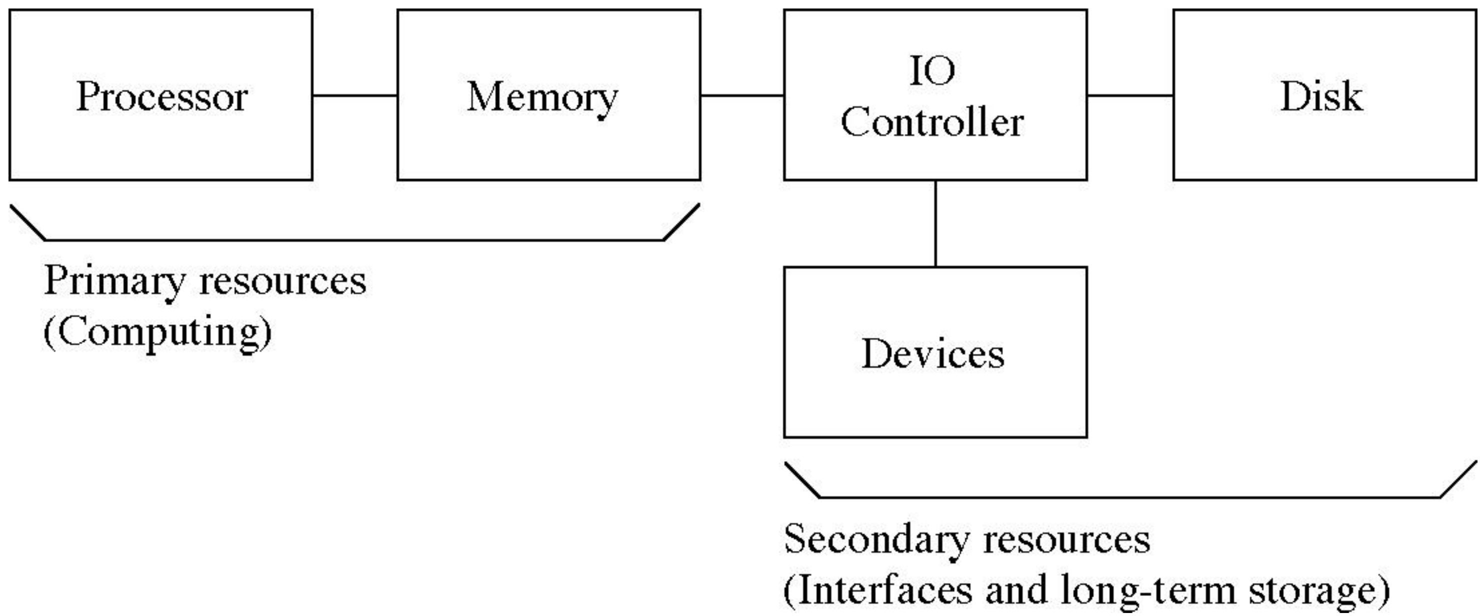
Operating System Functions

- Resource manager
  - manage hardware and software resources
- Virtual machine manager
  - implement a virtual machine for processes to run in
  - a nicer environment than the bare hardware

Hardware Resources

- *Processor*: execute instructions
- *Memory*: store programs and data
- *Input/output (I/O)controllers*: transfer to and from devices
- *Disk devices*: long-term storage
- *Other devices*: conversion between internal and external data representations

Hardware Resources



Resource Management Functions

- *Transforming* physical resources to logical resources
  - Making the resources easier to use
- *Multiplexing* one physical resource to several logical resources
  - Creating multiple, logical copies of resources
- *Scheduling* physical and logical resources
  - Deciding who gets to use the resources

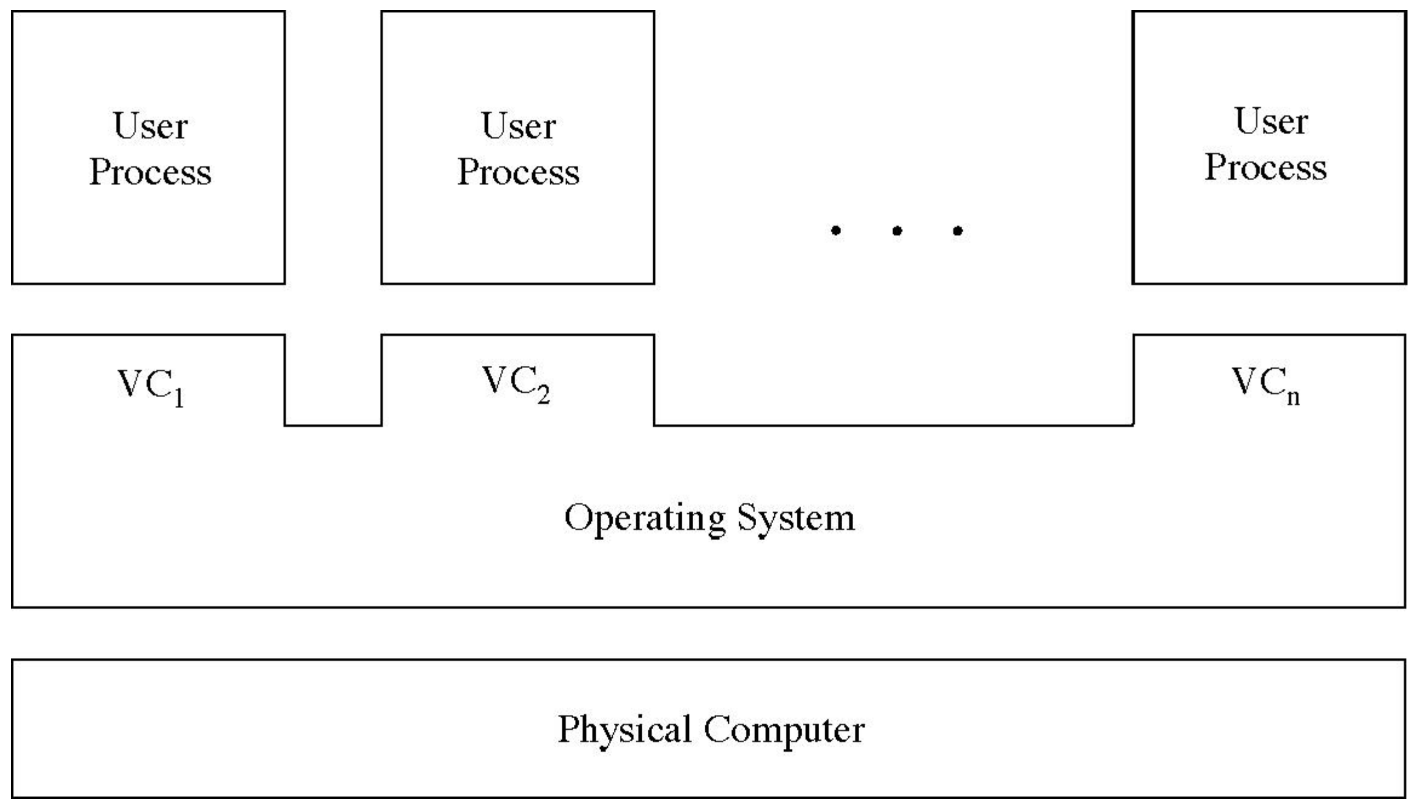
Types of Multiplexing

- Time multiplexing
  - time-sharing
  - scheduling a serially-reusable resource among several users
- Space multiplexing
  - space-sharing
  - dividing a multiple-use resource up among several users

Virtual Computers

- Processor virtualized to processes
  - mainly time-multiplexing
- Memory virtualized to address spaces
  - space and time multiplexing
- Disks virtualized to files
  - space-multiplexing
  - transforming

Multiple Virtual Computers



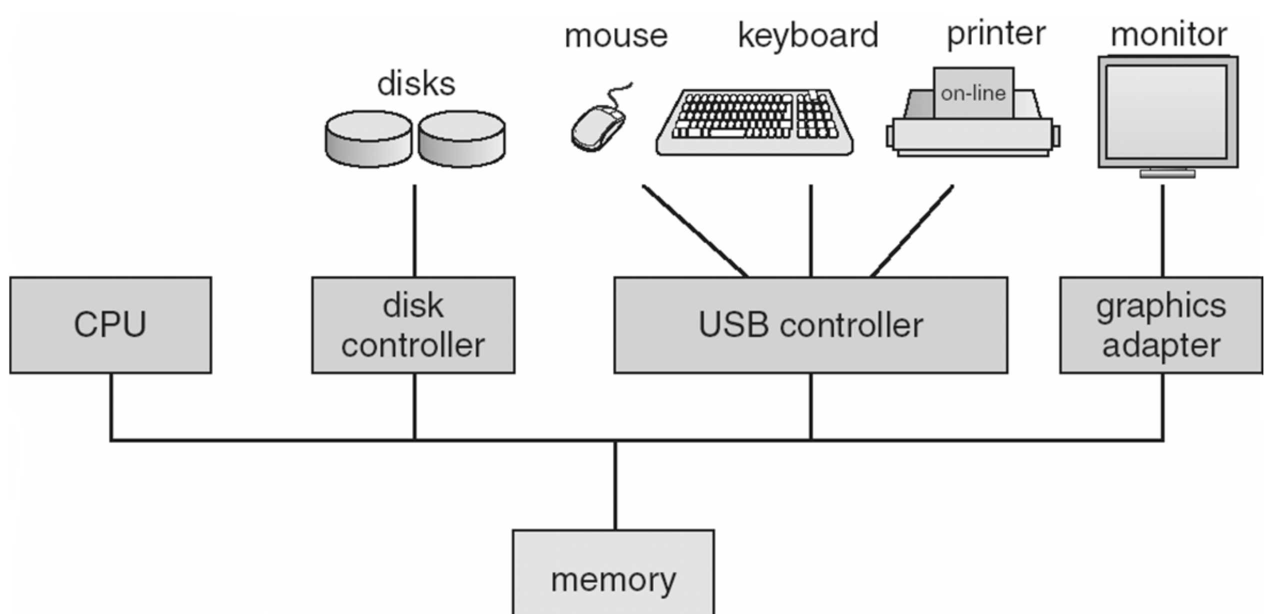
Do we need an OS?

- Not always
  - Some programs run “stand-alone”
- But they are very useful
  - Reusable functions
  - Easier to use than the bare hardware

System Structures I

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Computer-Components

- Computer consists of processor, memory, and I/O components, with one or more modules of each type. These modules are connected through interconnection network.
- I/O devices and the CPU can execute concurrently.
- Each device controller is in-charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- **Device controller informs CPU that it has finished its operation by causing an *interrupt*.**

I/O Hardware

- I/O devices can be categorized as storage, communications, user-interface, and others
- Devices connect with the computer via ***ports***, e.g. a serial or parallel port.
- A common set of wires connecting multiple devices is termed a ***bus.***
  - Buses include rigid protocols for the types of messages that can be sent across the bus and the procedures for resolving contention issues.
  - Bus types commonly found in a modern PC:
    - The ***PCI bus*** connects high-speed high-bandwidth devices to the memory subsystem (and the CPU.)
    - The ***expansion bus*** connects slower low-bandwidth devices, which typically deliver data one character at a time (with buffering.)
    - The ***SCSI bus*** connects a number of SCSI devices to a common SCSI controller.
    - A ***daisy-chain bus,*** is when a string of devices is connected to each other like beads on a chain, and only one of the devices is directly connected to the host.

THANK YOU