

# Computer System Architecture

## 1. Introduction

- Computer systems can be categorized based on the **number of general-purpose processors**.
- Main types:
  1. Single Processor System
  2. Multiprocessor System
  3. Clustered System

## 2. Single Processor System

- **One main CPU** executes general-purpose instruction sets (user & OS tasks).
- **Special-purpose processors** may exist (e.g., keyboard microprocessor), but these are device-specific only.
- **Classification is based on general-purpose processors**, so still considered "single processor system".

## 3. Multiprocessor System

- Contains **two or more processors**.
- Also called **Parallel Systems** or **Tightly Coupled Systems**.
- Processors share:
  - Computer bus
  - Sometimes clock, memory, and I/O devices
- Require **close communication & synchronization**.

### Advantages:

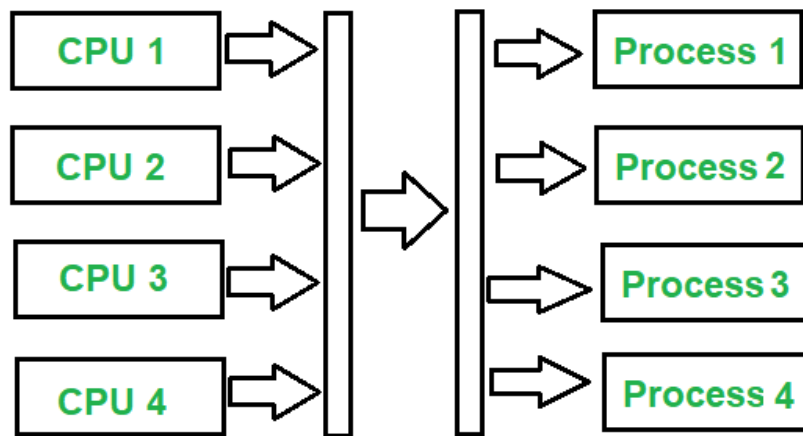
1. **Increased Throughput** – Parallel execution increases performance.
2. **Economy of Scale** – Multiple processors share resources → more cost-effective than multiple single-CPU systems.

3. **Increased Reliability** – If one processor fails, others continue (no total failure).

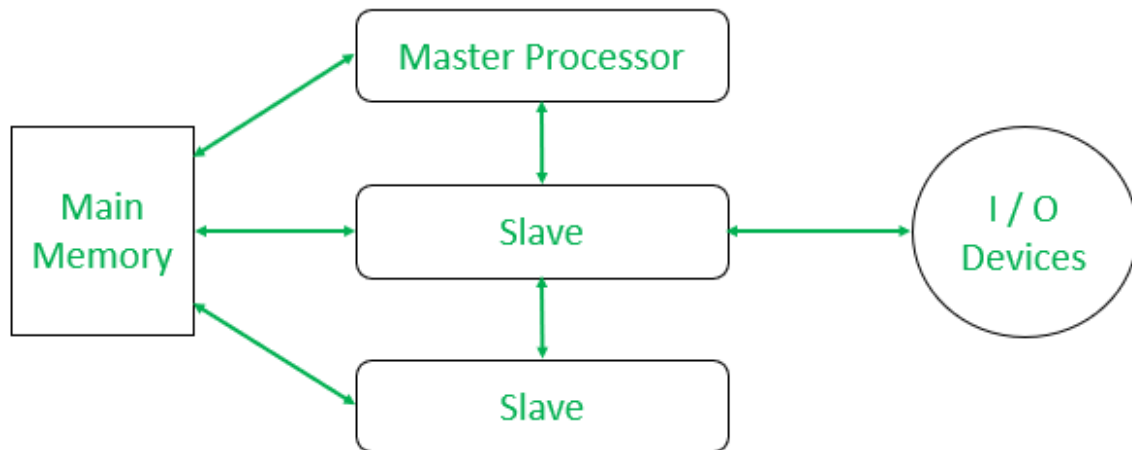
### Types of Multiprocessing:

- **Symmetric Multiprocessing (SMP):**
  - All CPUs are peers (identical).
  - All participate equally in task execution.

### Symmetric Multiprocessing



- **Asymmetric Multiprocessing (AMP):**
  - **Master-Slave approach:** one CPU (master) controls others (slaves).
  - Master assigns tasks & supervises slaves.



#### 4. Clustered Systems

- Composed of **two or more complete systems** (not just processors) **coupled together**.
- Purpose: **High availability & reliability**.
- If one system fails, others continue the work.

##### Structures:

- **Asymmetric Clustering:**
  - One machine in **hot-standby** mode (acts as monitor/master).
  - Other machines run applications.
  - If one fails, standby machine takes over.
- **Symmetric Clustering:**
  - Two or more systems **run applications & monitor each other**.
  - More efficient use of resources compared to asymmetric.

### 1.1 Operating System

- An OS is a program that acts as an intermediary between
  - computer-user and
  - computer-hardware.
- It also provides a basis for application-programs
- Goals of OS:
  - To execute programs.
  - To make solving user-problems easier.
  - To make the computer convenient to use.
- The OS (also called kernel) is the one program running at all times on the computer.
- Different types of OS:
  - Mainframe OS is designed to optimize utilization of hardware.
  - Personal computer (PC) OS supports complex game, business application.
  - Handheld computer OS is designed to provide an environment in which a user can easily interface with the computer to execute programs.

### 1.2 What Operating Systems do?

- Four components of a computer (Figure 1.1):
  - 1) Hardware
  - 2) OS
  - 3) Application programs and
  - 4) Users

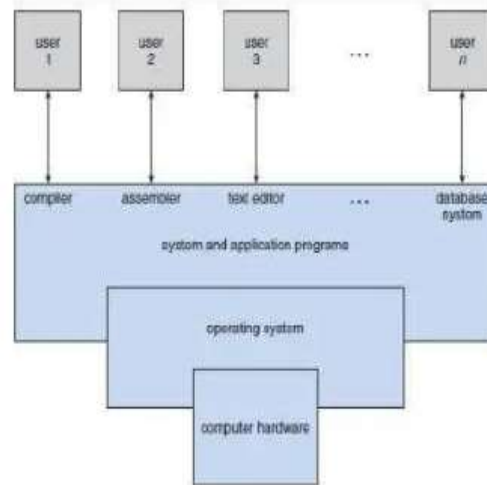


Figure 1.1 Abstract view of the components of a computer system

- Hardware provides basic computing-resources:
  - CPU
  - memory and
  - I/O devices.
- Application-program defines how the resources are used to solve computing-problems of the users.  
Ex: word processors, spread sheets, compilers.
- The OS controls & co-ordinates the use of hardware among various application-program for various users.
- Two views of OS:
  - 1) User and
  - 2) System.

Have the passion, take the action & magic will happen.

---

## 1.3 Computer System Organization

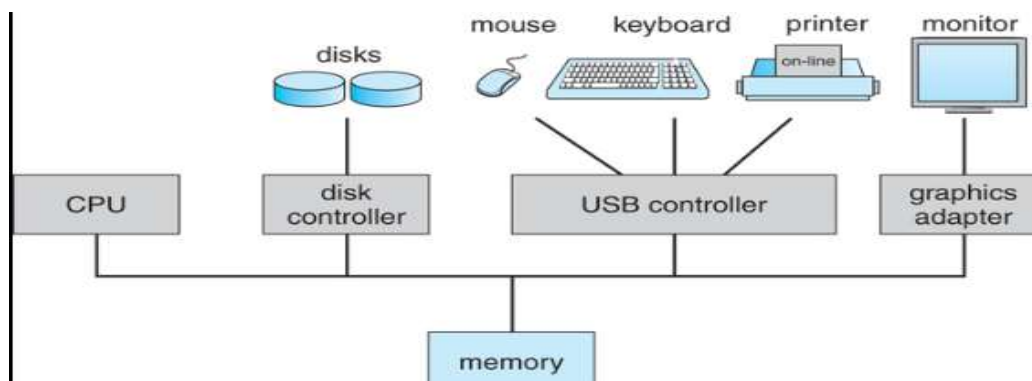
### 1.3.1 Computer System Organization

- A computer consists of
  - one or more CPUs and
  - no. of device-controllers (Figure 1.2).
- Controller is in charge of a specific type of device (for ex: audio devices).
- CPU and controllers can execute concurrently.
- A memory-controller is used to synchronize access to the shared-memory.
- Following events occur for a computer to start running:
  - 1) Bootstrap program is an initial program that runs when computer is powered-up.
  - 2) Bootstrap program
    - initializes all the system from registers to memory-contents and
    - loads OS into memory.
  - 3) Then, OS
    - starts executing the first process (such as "init") and
    - waits for some event to occur.
  - 4) The occurrence of an event is signaled by an interrupt from either the hardware or the software (Figure 1.3).
    - i) Hardware may trigger an interrupt by sending a signal to the CPU.
    - ii) Software may trigger an interrupt by executing a system-call.
  - 5) When CPU is interrupted, the CPU
    - stops current computation and
    - transfers control to ISR (interrupt service routine).
  - 6) Finally, the ISR executes; on completion, the CPU resumes the interrupted computation.

### Common Functions of Interrupts

- Interrupt transfers control to the ISR generally, through the interrupt-vector, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted-instruction.
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- A trap is a software-generated interrupt caused either by an error or a user request.
- A modern OS is interrupt-driven.

## DEPARTMENT OF COMPUTER APPLICATIONS



### 1.4 Computer System Architecture

- 1) Single-Processor Systems
- 2) Multiprocessor Systems
- 3) Clustered Systems

#### 1.4.1 Single Processor Systems

- The system has only one general-purpose CPU.
- The CPU is capable of executing a general-purpose instruction-set.
- These systems range from PDAs through mainframes.
- Almost all systems have following processors:
  - 1) **Special Purpose Processors**
    - Include disk, keyboard, and graphics controllers.
  - 2) **General Purpose Processors**
    - Include I/O processors.
- Special-purpose processors run a limited instruction set and do not run user-processes.

#### 1.4.2 Multi-Processor Systems

- These systems have two or more processors which can share:
  - bus
  - clock
  - memory/peripheral devices
- Advantages:
  - 1) **Increased Throughput**
    - By increasing no. of processors, we expect to get more work done in less time.
  - 2) **Economy of Scale**
    - These systems are cheaper because they can share
      - peripherals
      - mass-storage
      - power-supply.
    - If many programs operate on same data, they will be stored on one disk & all processors can share them.
  - 3) **Increased Reliability**
    - The failure of one processor will not halt the system.
- Two types of multiple-processor systems:
  - 1) Asymmetric multiprocessing (AMP) and
  - 2) Symmetric multiprocessing (SMP)
- 1) **Asymmetric Multiprocessing**
  - This uses master-slave relationship (Figure 1.6).
  - Each processor is assigned a specific task.
  - A master-processor controls the system.
    - The other processors look to the master for instruction.
  - The master-processor schedules and allocates work to the slave-processors.
- 2) **Symmetric Multiprocessing**
  - Each processor runs an identical copy of OS.
  - All processors are peers; no master-slave relationship exists between processors.
  - Advantages:
    - 1) Many processes can run simultaneously.
    - 2) Processes and resources are shared dynamically among the various processors.
  - Disadvantage:
    - 1) Since CPUs are separate, one CPU may be sitting idle while another CPU is overloaded. This results in inefficiencies.

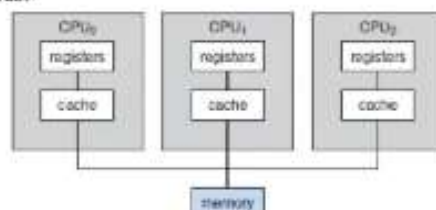


Figure 1.6 Symmetric multiprocessing architecture

You are what you think about all day long.



### 1.4.3 Clustered Systems

- These systems consist of two or more systems coupled together (Figure 1.7).
- These systems share storage & closely linked via LAN.
- Advantage:
  - 1) Used to provide high-availability service.
- High-availability is obtained by adding a level of redundancy in the system.
- Working procedure:
  - A cluster-software runs on the cluster-nodes.
  - Each node can monitor one or more other nodes (over the LAN).
  - If the monitored-node fails, the monitoring-node can
    - take ownership of failed-node's storage and
    - restart the applications running on the failed-node.
  - The users and clients of the applications see only a brief interruption of service.
- Two types are:
  - 1) Asymmetric and
  - 2) Symmetric

#### 1) Asymmetric Clustering

- One node is in hot-standby mode while the other nodes are running the applications.
- The hot-standby node does nothing but monitor the active-server.
- If the server fails, the hot-standby node becomes the active server.

#### 2) Symmetric Clustering

- Two or more nodes are running applications, and are monitoring each other.
- Advantage:
  - 1) This mode is more efficient, as it uses all of the available hardware.
- It does require that more than one application be available to run.

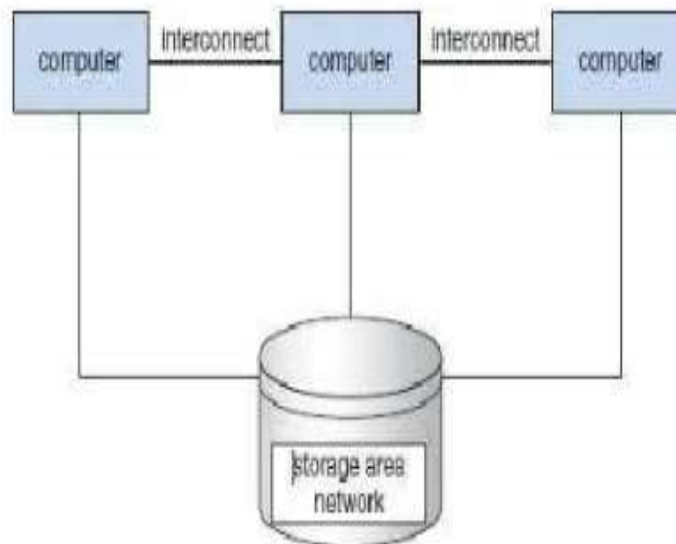


Figure 1.7 General structure of a clustered system

## 1.6 Operating System Operations

- Modern OS is interrupt driven.
- Events are always signaled by the occurrence of an interrupt or a trap.
- A trap is a software generated interrupt caused either by
  - error (for example division by zero) or
  - request from a user-program that an OS service be performed.
- For each type of interrupt, separate segments of code in the OS determine what action should be taken.
- ISR (Interrupt Service Routine) is provided that is responsible for dealing with the interrupt.

### 1.6.1 Dual Mode Operation

- Problem: We must be able to differentiate between the execution of
  - OS code and
  - user-defined code.
- Solution: Most computers provide hardware-support.
- Two modes of operation (Figure 1.9):
  - 1) User mode and
  - 2) Kernel mode
- A mode bit is a bit added to the hardware of the computer to indicate the current mode: i.e. kernel (0) or user (1)

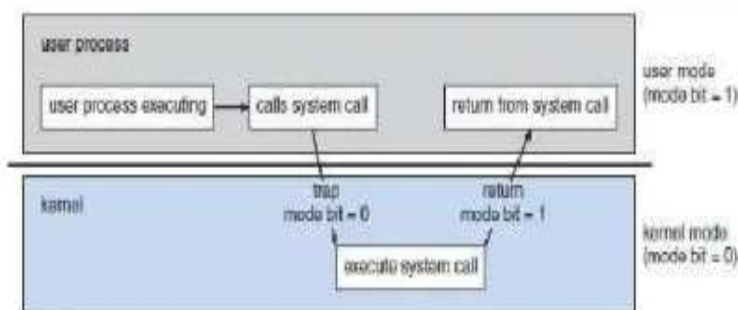


Figure 1.9 Transition from user to kernel mode

- Working principle:
  - 1) At system boot time, the hardware starts in kernel-mode.
  - 2) The OS is then loaded and starts user applications in user-mode.
  - 3) Whenever a trap or interrupt occurs, the hardware switches from user-mode to kernel-mode (that is, changes the state of the mode bit to 0).
  - 4) The system always switches to user-mode (by setting the mode bit to 1) before passing control to a user-program.
- Dual mode protects
  - OS from errant users and
  - errant users from one another.
- Privileged instruction is executed only in kernel-mode.
- If an attempt is made to execute a privileged instruction in user-mode, the hardware treats it as illegal and traps it to the OS.
- A system calls are called by user-program to ask the OS to perform the tasks on behalf of the user program.



### 1.6.2 Timer

- Problem: We cannot allow a user-program to get stuck in an infinite loop and never return control to the OS.

Solution: We can use a timer.

- A timer can be set to interrupt the computer after a specific period.
- The period may be fixed (for ex: 1/60 second) or variable (for ex: from 1ns to 1ms).
- A variable timer is implemented by a fixed-rate clock and a counter.
- Working procedure:
  - 1) The OS sets the counter.
  - 2) Every time the clock ticks, the counter is decremented.
  - 3) When the counter reaches 0, an interrupt occurs.
- The instructions that modify the content of the timer are privileged instructions.

### 1.7 Process Management

- The OS is responsible for the following activities:
  - 1) Creating and deleting both user and system processes
  - 2) Suspending and resuming processes
  - 3) Providing mechanisms for process synchronization
  - 4) Providing mechanisms for process communication
  - 5) Providing mechanisms for deadlock handling
- A process needs following resources to do a task:
  - CPU
  - memory and
  - files.
- The resources are allocated to process
  - when the process is created or
  - while the process is running.
- When the process terminates, the OS reclaims all the reusable resources.
- A program by itself is not a process;
  - 1) A program is a passive entity (such as the contents of a file stored on disk).
  - 2) A process is an active entity.
- Two types of process:
  - 1) **Single-threaded process** has one PC(program counter) which specifies location of the next instruction to be executed.
  - 2) **Multi-threaded process** has one PC per thread which specifies location of next instruction to execute in each thread

### 1.8 Memory Management

- The OS is responsible for the following activities:
  - 1) Keeping track of which parts of memory are currently being used and by whom
  - 2) Deciding which processes are to be loaded into memory when memory space becomes available
  - 3) Allocating and de-allocating memory space as needed.
- Main memory is the array of bytes ranging from hundreds to billions.
- Each byte has its own address.
- The CPU
  - reads instructions from main memory during the instruction-fetch cycle.
  - reads/writes data from/to main-memory during the data-fetch cycle.
- To execute a program:
  - 1) The program will be
    - loaded into memory and
    - mapped to absolute addresses.
  - 2) Then, program accesses instructions & data from memory by generating absolute addresses.
  - 3) Finally, when program terminates, its memory-space is freed.
- To improve CPU utilization, keep several programs will be kept in memory
- Selection of a memory-management scheme depends on hardware-design of the system.

**1.9 Storage Management**

- 1) File-System Management
- 2) Mass-Storage Management
- 3) Caching

**1.9.1 File System Management**

- The OS is responsible for following activities:
  - 1) Creating and deleting files.
  - 2) Creating and deleting directories.
  - 3) Supporting primitives for manipulating files & directories.
  - 4) Mapping files onto secondary storage.
  - 5) Backing up files on stable (non-volatile) storage media.
- Computer stores information on different types of physical media.  
For ex: magnetic disk, optical disk.
- Each medium is controlled by a device (e.g. disk drive).
- The OS
  - maps files onto physical media and
  - accesses the files via the storage devices
- File is a logical collection of related information.
- File consists of both program & data.
- Data files may be numeric, alphabets or binary.
- When multiple users have access to files, access control (read, write) must be specified.

**1.9.2 Mass Storage Management**

- The OS is responsible for following activities:
  - 1) Free-space management
  - 2) Storage allocation and
  - 3) Disk scheduling.
- Usually, disks used to store
  - data that does not fit in main memory or
  - data that must be kept for a "long" period of time.
- Most programs are stored on disk until loaded into memory.
- The programs include
  - compilers
  - word processors and
  - editors.
- The programs use the disk as both the source and destination of their processing.
- Entire speed of computer operation depends on disk and its algorithms.

**1.9.3 Caching**

- Caching is an important principle of computer systems.
- Information is normally kept in some storage system (such as main memory).
- As it is used, it is copied into a faster storage system called as the cache on a temporary basis.
- When we need a particular piece of information:
  - 1) We first check whether the information is in the cache.
  - 2) If information is in cache, we use the information directly from the cache.
  - 3) If information is not in cache, we use the information from the source, putting a copy in the cache under the assumption that we will need it again soon.
- In addition, internal programmable registers, such as index registers, provide high-speed cache for main memory.
- The compiler implements the register-allocation and register-replacement algorithms to decide which information to keep in registers and which to keep in main memory.
- Most systems have an instruction cache to hold the instructions expected to be executed next.
- Most systems have one or more high-speed data caches in the memory hierarchy
- Because caches have limited size, cache management is an important design problem  
Careful selection of cache size & of a replacement policy can result in greatly increased performance

---

**1.9.4 I/O Systems**

- The OS must hide peculiarities of hardware devices from users.
- In UNIX, the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem.
- The I/O subsystem consists of
  - 1) A memory-management component that includes buffering, caching, and spooling.
  - 2) A general device-driver interface.
  - 3) Drivers for specific hardware devices.
- Only the device driver knows the peculiarities of the specific device to which it is assigned.

**1.10 Protection and Security**

- Protection is a mechanism for controlling access of processes or users to resources defined by OS.
- This mechanism must provide
  - means for specification of the controls to be imposed and
  - means for enforcement.
- Protection can improve reliability by detecting latent errors at the interfaces between subsystems.
- Security means defense of the system against internal and external attacks.
- The attacks include
  - viruses and worms
  - DOS(denial-of-service)
  - identity theft.
- Protection and security require the system to be able to distinguish among all its users.
  - 1) User identities (user IDs) include name and associated number, one per user.
    - User IDs are associated with all files (or processes) of that user to determine access control.
  - 2) Group identifier (group ID): can be used to define a group name and the set of users belonging to that group.
    - A user can be in one or more groups, depending on operating-system design decisions.

**1.11 Distributed System**

- This is a collection of physically separate, possibly heterogeneous computer-systems.
- The computer-systems are networked to provide the users with access to the various resources.
- Access to a shared resource increases
  - computation speed
  - functionality
  - data availability and
  - reliability
- A network is a communication path between two or more systems.
- Networks vary by the
  - protocols used
  - distances between nodes and
  - transport media.
- Common network protocol are
  - TCP/IP
  - ATM.
- Networks are characterized based on the distances between their nodes.
  - A local-area network (LAN) connects computers within a building.
  - A wide-area network (WAN) usually links buildings, cities, or countries.
  - A metropolitan-area network (MAN) could link buildings within a city.
- The media to carry networks are equally varied. They include
  - copper wires,
  - fiber strands, and
  - wireless transmissions.



### 1.13 Computing Environments

- 1) Traditional Computing
- 2) Client-Server Computing
- 3) Peer-to-Peer Computing
- 4) Web-Based Computing

#### 1.13.1 Traditional Computing

- Used in office environment:
  - PCs connected to a network, with servers providing file and print services.
- Used in home networks:
  - At home, most users had a single computer with a slow modem.
  - Some homes have firewalls to protect their networks from security breaches.
- Web technologies are stretching the boundaries of traditional computing.
  - Companies establish portals, which provide web accessibility to their internal servers.
  - Network computers are terminals that understand web computing.
  - Handheld PDAs can connect to wireless networks to use company's web portal.
- Systems were either batch or interactive.
  - 1) Batch system processed jobs in bulk, with predetermined input.
  - 2) Interactive systems waited for input from users.

#### 1.13.2 Client-Server Computing

- Servers can be broadly categorized as (Figure 1.10):
  - 1) Compute servers and
  - 2) File servers
- 1) Compute-server** system provides an interface to which a client can send a request to perform an action (for example, read data).
  - In response, the server executes the action and sends back results to the client.
- 2) File-server** system provides a file-system interface where clients can create, read, and delete files.
  - For example: web server that delivers files to clients running web browsers.

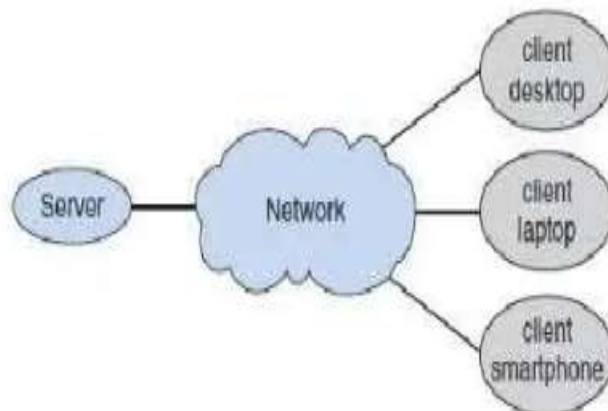


Figure 1.10 General structure of a client-server system.

### 1.13.3 Peer-to-Peer Computing

- All nodes are considered peers, and each may act as either a client or a server (Figure 1.11).
- Advantage:
  - 1) In a client-server system, the server is a bottleneck; but in a peer-to-peer system, services can be provided by several nodes distributed throughout the network.
- A node must first join the network of peers.
- Determining what services are available is done in one of two general ways:
  - 1) When a node joins a network, it registers its service with a centralized lookup service on the network.
    - Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service.
  - 2) A peer broadcasts a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer.

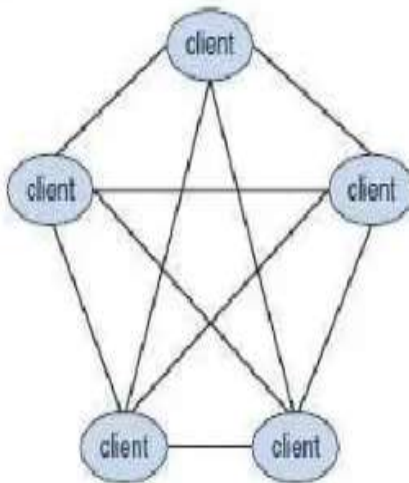


Figure 1.11 Peer-to-peer system with no centralized service.

### 1.13.4 Web-Based Computing

- This includes
  - PC
  - handheld PDA &
  - cell phones
- Load balancer is a new category of devices to manage web traffic among similar servers.
- In load balancing, network connection is distributed among a pool of similar servers.
- More devices becoming networked to allow web access
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers



### 1.14 Operating System Services

- An OS provides an environment for the execution of programs.
- It provides services to
  - programs and
  - users.

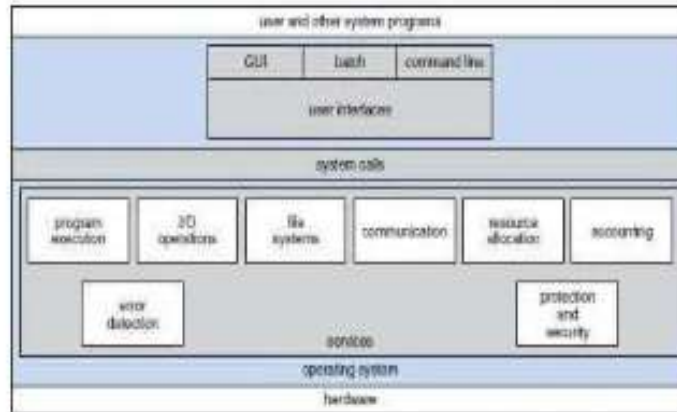


Figure 1.12 A view of OS services

- Common functions helpful to the user are (Figure 1.12):

#### 1) User Interface

- Almost all OS have a user-interface (UI).
- Different interfaces are:

##### i) CLI (Command Line Interface)

- ✧ This uses
  - text commands and
  - method for entering the text commands.

##### ii) Batch Interface

- ✧ Commands & directives to control those commands are entered into files, and those files are executed.

##### iii) GUI (Graphical User Interface)

- ✧ The interface is a window-system with a pointing-device to
  - direct I/O
  - choose from menus and
  - make selections.

#### 2) Program Execution

- The system must be able to
  - load a program into memory and
  - run the program.
- The program must be able to end its execution, either normally or abnormally.

#### 3) I/O Operations

- The OS must provide a means to do I/O operations because users cannot control I/O devices directly.
- For specific devices, special functions may be desired (ex: to blank CRT screen).

#### 4) File-System Manipulation

- Programs need to
  - read & write files (or directories)
  - create & delete files
  - search for a given file and
  - allow or deny access to files.

**DEPARTMENT OF COMPUTER APPLICATIONS**

---

---

### **5) Communications**

- In some situations, one process needs to communicate with another process.
- Communications may be implemented via
  1. Shared memory or
  2. Message passing
- In message passing, packets of information are moved between processes by OS.

### **6) Error Detection**

- Errors may occur in
    - CPU & memory-hardware (ex: power failure)
    - I/O devices (ex: lack of paper in the printer) and
    - user program (ex: arithmetic overflow)
  - For each type of error, OS should take appropriate action to ensure correct & consistent computing.
- Common functions for efficient operation of the system are:
    - 1) Resource Allocation**
      - When multiple users are logged on the system at the same time, resources must be allocated to each of them.
      - The OS manages different types of resources.
      - Some resources (say CPU cycles) may have special allocation code.  
Other resources (say I/O devices) may have general request & release code.
    - 2) Accounting**
      - We want to keep track of
        - which users use how many resources and
        - which kinds of resources.
      - This record keeping may be used for
        - accounting (so that users can be billed) or
        - gathering usage-statistics.
    - 3) Protection**
      - When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the OS itself.
      - Protection involves ensuring that all access to resources is controlled.
      - Security starts with each user having authenticated to the system by means of a password.

### 1.16 System Calls

- These provide an interface to the OS services.
- These are available as routines written in C and C++.
- The programmers design programs according to an API. (API=application programming interface).
- The API
  - defines a set of functions that are available to the programmer (Figure 1.15).
  - includes the parameters passed to functions and the return values.
- The functions that make up an API invoke the actual system-calls on behalf of the programmer.
- Benefits of API:
  - 1) Program portability.
  - 2) Actual system-calls are more detailed (and difficult) to work with than the API available to the programmer.
- Three general methods are used to pass parameters to the OS:
  - 1) via registers.
  - 2) Using a table in memory & the address is passed as a parameter in a register (Figure 1.16).
  - 3) The use of a stack is also possible where parameters are pushed onto a stack and popped off the stack by the OS.

---

### 1.17 Types of System Calls

- 1) Process control
- 2) File management
- 3) Device management
- 4) Information maintenance
- 5) Communications

#### 1.17.1 Process Control

- System calls used:
    - end, abort
    - load, execute
    - create process, terminate process
    - get process attributes, set process attributes
    - wait for time
    - wait event, signal event
    - allocate and free memory
  - A running program needs to be able to halt its execution either normally (**end**) or abnormally (**abort**).
  - If program runs into a problem, error message may be generated and dumped into a file. This file can be examined by a debugger to determine the cause of the problem.
  - The OS must transfer control to the next invoking command interpreter.
    - Command interpreter then reads next command.
    - In interactive system, the command interpreter simply continues with next command.
    - In GUI system, a pop-up window will request action from user.
- How to deal with new process?
- A process executing one program can **load** and **execute** another program.
  - Where to return control when the loaded program terminates?  
The answer depends on the existing program:
    - 1) If control returns to the existing program when the new program terminates, we must save the memory image of the existing program. (Thus, we have effectively created a mechanism for one program to call another program).
    - 2) If both programs continue concurrently, we **created** a new process to be multiprogrammed.
  - We should be able to control the execution of a process. i.e. we should be able to determine and reset the attributes of a process such as:
    - job's priority or
    - maximum execution time
  - We may also want to **terminate** process that we created if we find that it
    - is incorrect or
    - is no longer needed.
  - We may need to **wait** for processes to finish their execution.  
We may want to wait for a specific event to occur.
  - The processes should then signal when that event has occurred.