

Threads

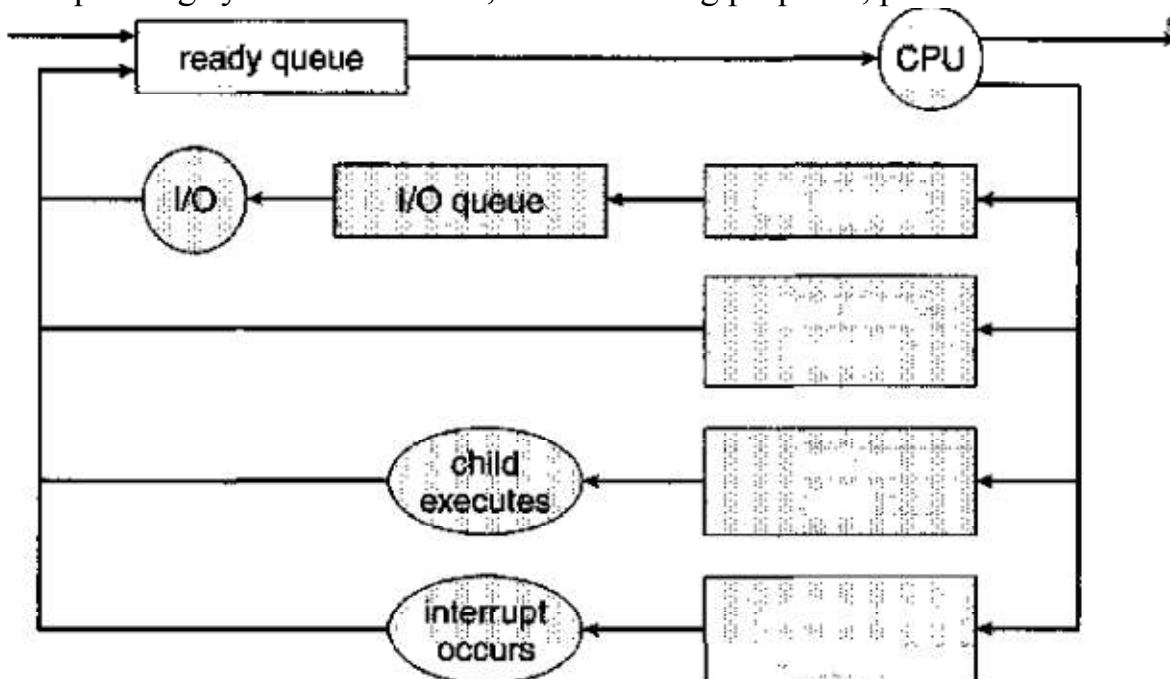
The process model discussed so far has implied that a process is a program that performs a single **thread** of execution. For example, when a process is running a word-processor program, a single thread of instructions is being executed. This single thread of control allows the process to perform only one task at one time. The user cannot simultaneously type in characters and run the spell checker within the same process, for example. Many modern operating systems have extended the process concept to allow a process to have multiple

Process Scheduling

The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet these objectives, the **process scheduler** selects an available process (possibly from a set of several available processes) for program execution on the CPU. For a single-processor system, there will never be more than one running process. If there are more processes, the rest will have to wait until the CPU is free and can be rescheduled

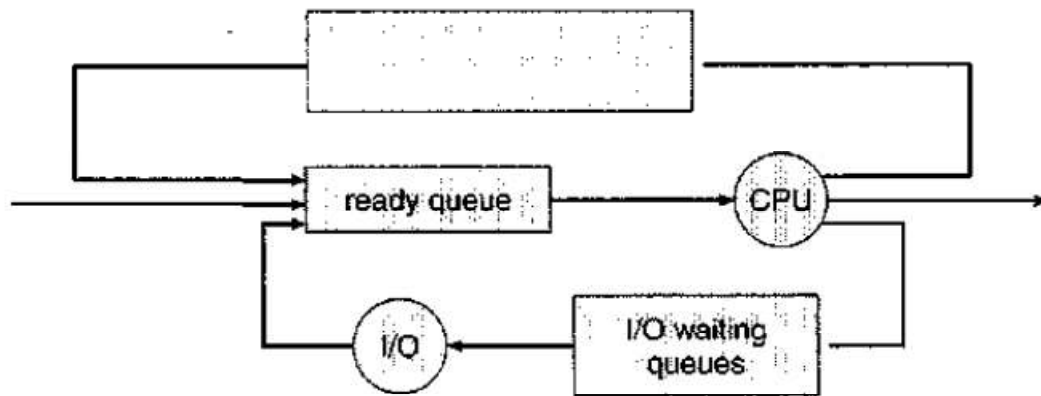
Schedulers

A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes



from these queues in some fashion. The selection process is carried out by the appropriate **scheduler**. Often, in a batch system, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. The **long-term**

scheduler, or **job scheduler**, selects processes from this pool and loads them into memory for execution. The **short-term scheduler**, or **CPU scheduler**, selects from among the processes that are ready to execute and allocates the CPU to one of them.



Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling. This **medium-term scheduler** is diagrammed in Figure . The key idea behind a medium-term scheduler is that sometimes it can be advantageous to remove processes from memory (and from active contention for the CPU) and thus reduce the degree of multiprogramming.

Context Switch

Interrupts cause the operating system to change a CPU from its current task and to run a kernel routine. Such operations happen frequently on general-purpose systems. When an interrupt occurs, the system needs to save the current **context** of the process currently running on the CPU so that it can restore that context when its processing is done, essentially suspending the process and then resuming it. The context is represented in the PCB of the process; it includes the value of the CPU registers, the process state and memory-management information. Generically, we perform a **state save** of the current state of the CPU, be it in kernel or user mode, and then a **state restore** to resume operations. Switching the CPU to another process requires performing a **state save** of the current process and a **state restore** of a different process. This task is known as a **context switch**. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context-switch time is pure overhead, because the system does no useful work while switching. Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions

CPU Scheduling

CPU scheduling is the basis of multi programmed operating systems. The objective of multiprogramming is to have some process running at all times, in order to maximize CPU utilization. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use.

CPU-I/O Burst Cycle

Process execution consists of a **cycle** of CPU execution and I/O wait.

Processes alternate between these two states.

Process execution begins with a **CPU burst**.

That is followed by an I/O **burst**, then another CPU burst, then another I/O burst, and so on.

Eventually, the last CPU burst will end with a system request to terminate execution, rather than with another I/O burst.

CPU Scheduler

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the **short-term scheduler** (or CPU scheduler). The ready queue is not necessarily a first-in, first-out (FIFO) queue. It may be a FIFO queue, a priority queue, a tree, or simply an unordered linked list.

Preemptive Scheduling

CPU scheduling decisions may take place under the following four circumstances:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process terminates

Under 1 & 4 scheduling scheme is non preemptive.

Otherwise the scheduling scheme is preemptive.

Non-preemptive Scheduling

In non preemptive scheduling, once the CPU has been allocated a process, the process keeps the CPU until it releases the CPU either by termination or by switching to the waiting state. This scheduling method is used by the Microsoft windows environment.

Dispatcher

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler.

This function involves:

1. Switching context
2. Switching to user mode
3. Jumping to the proper location in the user program to restart that program

Scheduling Criteria

1. CPU utilization: The CPU should be kept as busy as possible. CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily used system).

2. Throughput: It is the number of processes completed per time unit. For long processes, this rate may be 1 process per hour; for short transactions, throughput might be 10 processes per second.

3. Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

4. Waiting time: Waiting time is the sum of the periods spent waiting in the ready queue.

5. Response time: It is the amount of time it takes to start responding, but not the time that it takes to output that response.

I/O Devices

I/O-device technology exhibits two conflicting trends. On one hand; we see increasing standardization of software and hardware interfaces. This trend helps us to incorporate improved device generations into existing computers and operating systems. On the other hand, we see an increasingly broad variety of I/O devices. Some new devices are so unlike previous devices that it is a challenge to incorporate them into our computers and operating systems. This challenge is met by a combination of hardware and software techniques. The basic I/O hardware elements, such as ports, buses, and device controllers, accommodate a wide variety of I/O devices. To encapsulate the details and oddities of different devices, the kernel of an operating system is structured to use device-driver modules. The device drivers present a uniform device access interface to the I/O subsystem, much as system calls provide a standard interface between the application and the operating system.

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

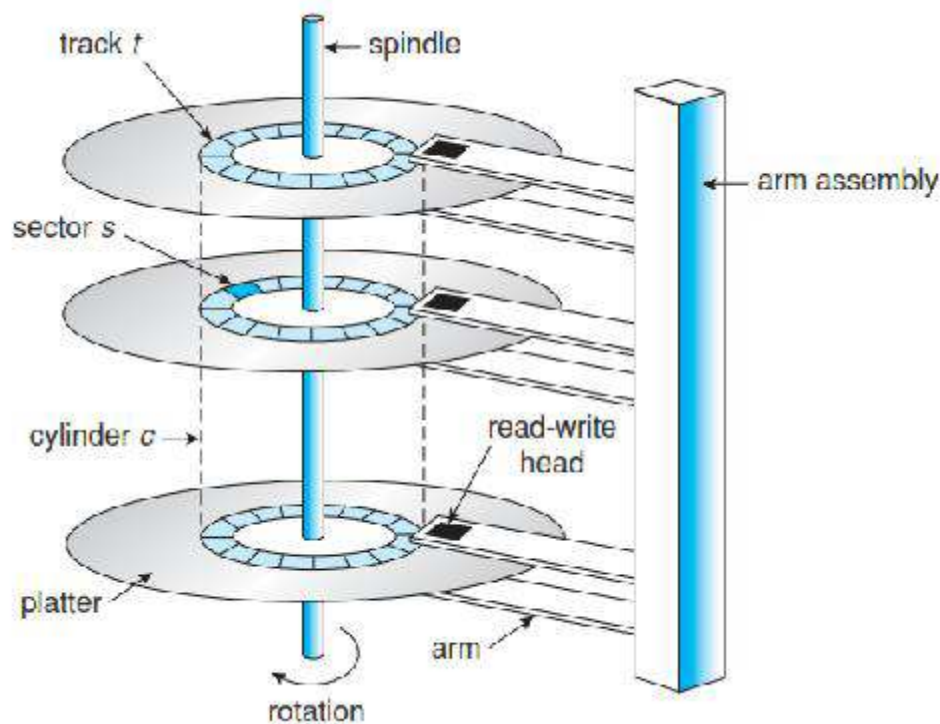
An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories –

- **Block devices** – A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
- **Character devices** – A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sounds cards etc

I/O SYSTEMS

Magnetic Disks

Magnetic disks provide the bulk of secondary storage for modern computer systems. Each disk platter has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 3.5 inches. The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters



A read–write head “flies” just above each surface of every platter. The heads are attached to a **disk arm** that moves all the heads as a unit. The surface of a platter is

logically divided into circular **tracks**, which are subdivided into **sectors**. The set of tracks that are at one arm position makes up a **cylinder**. There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors. The storage capacity of common disk drives is measured in gigabytes.

A disk drive is attached to a computer by a set of wires called an **I/O bus**. Several kinds of buses are available, including **advanced technology attachment (ATA)**, **serial ATA (SATA)**, **eSATA**, **universal serial bus (USB)**, and **fibre channel (FC)**. The data transfers on a bus are carried out by special electronic processors called **controllers**. The **host controller** is the controller at the computer end of the bus. A **disk controller** is built into each disk drive. To perform a disk I/O operation, the computer places a command into the host controller, typically using memory-mapped I/O ports. The host controller then sends the command via messages to the disk controller, and the disk controller operates the disk-drive hardware to carry out the command. Disk controllers usually have a built-in cache. Data transfer at the disk drive happens between the cache and the disk surface, and data transfer to the host, at fast electronic speeds, occurs between the cache and the host controller.

Disk Scheduling and Management

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives,

1. A fast access time and
2. High disk bandwidth.

- The **access time** has two major components;

- . The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.

- The **rotational latency** is the additional time waiting for the disk to rotate the desired sector to the disk head.

- The disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives,

1. A fast access time and
2. High disk bandwidth.

The **access time** has two major components;

The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.

The **rotational latency** is the additional time waiting for the disk to rotate the desired sector to the disk head.

The disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

We can improve both the access time and the bandwidth by disk scheduling. Disk scheduling: Servicing of disk I/O requests in a good order.

RAID

RAID is a technology that is used to increase the performance and/or reliability of data storage. The abbreviation stands for *Redundant Array of Inexpensive Disks*. A RAID system consists of two or more drives working in parallel. These disks can be hard discs, but there is a trend to also use the technology for SSD (solid state drives). There are different RAID levels, each optimized for a specific situation. These are not standardized by an industry group or standardization committee. This explains why companies sometimes come up with their own unique numbers and implementations. This article covers the following RAID levels:

- **RAID 0** – striping
- **RAID 1** – mirroring
- **RAID 5** – striping with parity
- **RAID 6** – striping with double parity
- **RAID 10** – combining mirroring and striping
-

RAID level 0 – Striping

- In a RAID 0 system data are split up into blocks that get written across all the drives in the array. By using multiple disks (at least 2) at the same time, this offers superior I/O performance. This performance can be enhanced further by using multiple controllers, ideally one controller per disk.

Advantages

- RAID 0 offers great performance, both in read and write operations. There is no overhead caused by parity controls.
- All storage capacity is used, there is no overhead.
- The technology is easy to implement.

Disadvantages

- RAID 0 is not fault-tolerant. If one drive fails, all data in the RAID 0 array are lost. It should not be used for mission-critical systems.

RAID level 1 – Mirroring

Data are stored twice by writing them to both the data drive (or set of data drives) and a mirror drive (or set of drives). If a drive fails, the controller uses either the data drive or the mirror drive for data recovery and continues operation

RAID level 5

RAID 5 is the most common secure RAID level. It requires at least 3 drives but can work with up to 16. Data blocks are striped across the drives and on one drive a parity checksum of all the block data is written. The parity data are not written to a fixed drive, they are spread across all drives

RAID level 6 – Striping with double parity

RAID 6 is like RAID 5, but the parity data are written to two drives. That means it requires at least 4 drives and can withstand 2 drives dying simultaneously. The chances that two drives break down at exactly the same moment are of course very small. However, if a drive in a RAID

RAID level 10 – combining RAID 1 & RAID 0

It is possible to combine the advantages (and disadvantages) of RAID 0 and RAID 1 in one single system. This is a nested or hybrid RAID configuration. It provides security by mirroring all data on secondary drives while using striping across each set of drives to speed up data transfers

Disk Management

1. Disk Formatting:

Before a disk can store data, the sector is divided into various partitions. This process is called lowlevel formatting or physical formatting. It fills the disk with a special data structure for each sector. The data structure for a sector consists of

Header,
Data area (usually 512 bytes in size), and
Trailer

File System Storage-File Concepts

File Concept

A file is a named collection of related information that is recorded on secondary storage.

- From a user's perspective, a file is the smallest allotment of logical secondary storage; that is, data cannot be written to secondary storage unless they are within a file.

Examples of files:

- A text file is a sequence of characters organized into lines (and possibly pages). A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements. An object file is a sequence of bytes organized into blocks understandable by the system's linker. An executable file is a series of code sections that the loader can bring into memory and execute.

File Attributes

- **Name:** The symbolic file name is the only information kept in human readable form.
- **Identifier:** This unique tag, usually a number identifies the file within the file system. It is the non-human readable name for the file.

- **Type:** This information is needed for those systems that support different types.

Location: This information is a pointer to a device and to the location of the file on that device.

- **Size:** The current size of the file (in bytes, words or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection:** Access-control information determines who can do reading, writing, executing and so on.
- **Time, date and user identification:** This information may be kept for creation, last modification and last use. These data can be useful for protection, security and usage monitoring.

File Operations

- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file

Access Methods

1. Sequential Access

a. The simplest access method is sequential access. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.

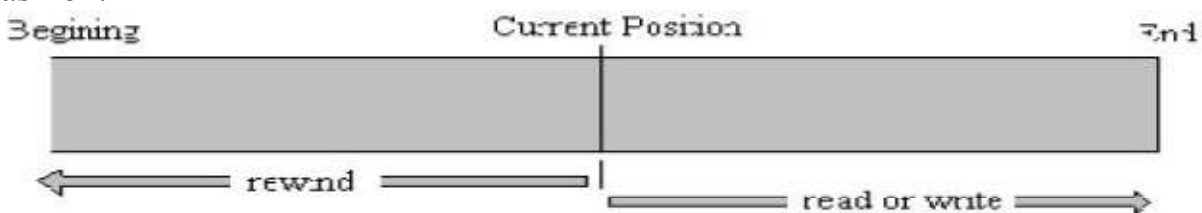


Fig 4.10 Sequential-access file

The bulk of the operations on a file is reads and writes. A read operation reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, a write appends to the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning and, on some systems, a program may be able to skip forward or back ward n records, for some integer n -perhaps only for $n=1$. Sequential access is

based on a tape model of a file, and works as well on sequential-access devices as it does on random – access ones.

2. Direct Access

Another method is direct access (or relative access). A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. The direct- access methods is based on a disk model of a file, since disks allow random access to any file block. For direct access, the file is viewed as a numbered sequence of blocks or records. A direct-access file allows arbitrary blocks to be read or written. Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file. Direct – access files are of great use for immediate access to large amounts of information. Database is often of this type. When a query concerning a particular subject arrives, we compute which block contains the answer, and then read that block directly to provide the desired information.

Directory and Disk Structure

There are five directory structures. They are

1. Single-level directory
2. Two-level directory
3. Tree-Structured directory
4. Acyclic Graph directory
5. General Graph directory

1. Single – Level Directory

- The simplest directory structure is the single- level directory.
- All files are contained in the same directory.

• Disadvantage:

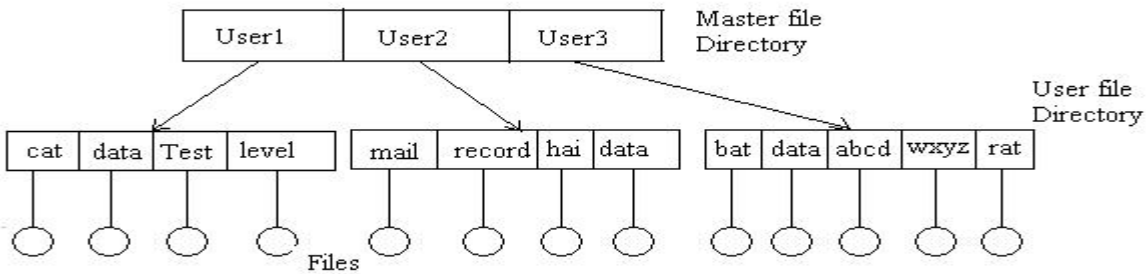
□ □ When the number of files increases or when the system has more than one user, since all files are in the same directory, they must have unique names.

2. Two – Level Directory

- In the two level directory structures, each user has her own user file directory (UFD).
- When a user job starts or a user logs in, the system's master file directory (MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user.
- When a user refers to a particular file, only his own UFD is searched.
- Thus, different users may have files with the same name.
- Although the two – level directory structure solves the name-collision problem

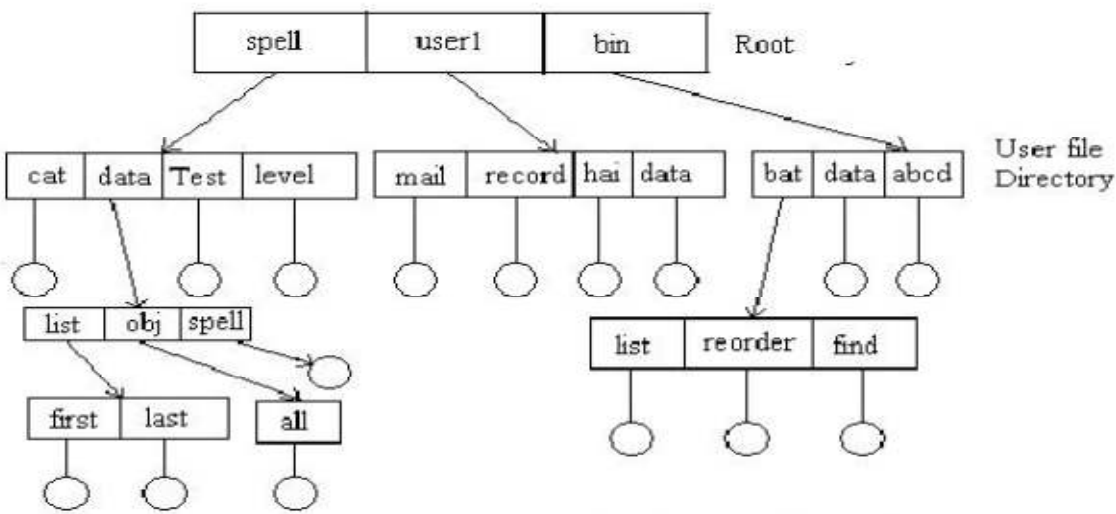
Disadvantage:

Users cannot create their own sub-directories.



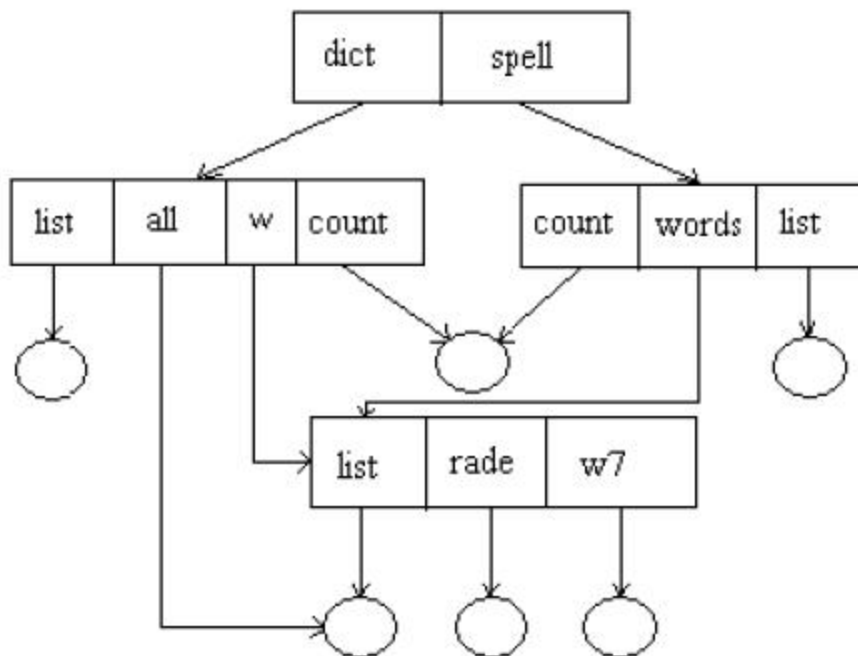
3. Tree – Structured Directory

- A tree is the most common directory structure.
- The tree has a root directory. Every file in the system has a unique path name.
- A **path name** is the path from the root, through all the subdirectories to a specified file.
- A directory (or sub directory) contains a set of files or sub directories.
- A directory is simply another file. But it is treated in a special way.
- All directories have the same internal format.
- One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1).
- Special system calls are used to create and delete directories.
- Path names can be of two types: absolute path names or relative path names.
- An absolute path name begins at the root and follows a path down to the specified file, giving the directory names on the path.
- A relative path name defines a path from the current directory.



4. Acyclic Graph Directory.

- An acyclic graph is a graph with no cycles.
- To implement shared files and subdirectories this directory structure is used.
- An acyclic – graph directory structure is more flexible than is a simple tree structure, but it is also more complex. In a system where sharing is implemented by symbolic link, this situation is somewhat easier to handle. The deletion of a link does not need to affect the original file; only the link is removed.
- Another approach to deletion is to preserve the file until all references to it are deleted. To implement this approach, we must have some mechanism for determining that the last reference to the file has been deleted.



Sharing and Protection

File Sharing

1. Multiple Users:

- When an operating system accommodates multiple users, the issues of file sharing, file naming and file protection become preeminent.
- The system either can allow user to access the file of other users by default, or it may require that a user specifically grant access to the files.
- These are the issues of access control and protection.
- To implementing sharing and protection, the system must maintain more file and directory attributes than a on a single-user system.
- The owner is the user who may change attributes, grand access, and has the most control over the file or directory.

- The group attribute of a file is used to define a subset of users who may share access to the file.
- Most systems implement owner attributes by managing a list of user names and associated user identifiers (user Ids).
- When a user logs in to the system, the authentication stage determines the appropriate user ID for the user. That user ID is associated with all of user's processes and threads. When they need to be user readable, they are translated, back to the user name via the user name list. Likewise, group functionality can be implemented as a system wide list of group names and group identifiers.
- Every user can be in one or more groups, depending upon operating system design decisions. The user's group Ids is also included in every associated process and thread.

2. Remote File System:

- Networks allowed communications between remote computers.
- Networking allows the sharing of resource spread within a campus or even around the world.
- User manually transfer files between machines via programs like **ftp**.
- A **distributed file system** (DFS) in which remote directories is visible from the local machine.
- **TheWorld Wide Web**: A browser is needed to gain access to the remote file and separate operations (essentially a wrapper for ftp) are used to transfer files.

a) The client-server Model:

- Remote file systems allow a computer to mount one or more file systems from one or more remote machines.
- A server can serve multiple clients, and a client can use multiple servers, depending on the implementation details of a given client-server facility.
- Client identification is more difficult. Clients can be specified by their network name or other identifier, such as IP address, but these can be spoofed (or imitate). An unauthorized client can spoof the server into deciding that it is authorized, and the unauthorized client could be allowed access.

b) Distributed Information systems:

- Distributed information systems, also known as distributed naming service, have been devised to provide a unified access to the information needed for remote computing.
- Domain name system (DNS) provides host-name-to-network address translations for their entire Internet (including the World Wide Web).
- Before DNS was invented and became widespread, files containing the same information were sent via e-mail or ftp between all networked hosts.

c) Failure Modes:

- **Redundant arrays of inexpensive disks (RAID)** can prevent the loss of a disk from resulting in the loss of data.
- Remote file system has more failure modes. By nature of the complexity of networking system and the required interactions between remote machines, many more problems can interfere with the proper operation of remote file systems.

d) Consistency Semantics:

- It is characterization of the system that specifies the semantics of multiple users accessing a shared file simultaneously.
- These semantics should specify when modifications of data by one user are observable by other users.
- The semantics are typically implemented as code with the file system.
- A series of file accesses (that is reads and writes) attempted by a user to the same file is always enclosed between the open and close operations.
- The series of access between the open and close operations is a **file session**.

File Protection

(i) Need for file protection.

- When information is kept in a computer system, we want to keep it safe from **physical damage** (reliability) and **improper access** (protection).
- Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically (or through computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed.
- File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism. Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost.
- Protection can be provided in many ways. For a small single-user system, we might provide protection by physically removing the floppy disks and locking them in a desk drawer or file cabinet. In a multi-user system, however, other mechanisms are needed.

(ii) Types of Access

- Complete protection is provided by prohibiting access.
- Free access is provided with no protection.
- Both approaches are too extreme for general use.

What is needed is **controlled access**.

- Protection mechanisms provide controlled access by limiting the types of file access that can be made. Access is permitted or denied depending on several

factors, one of which is the type of access requested. Several different types of operations may be controlled:

1. **Read:** Read from the file.
2. **Write:** Write or rewrite the file.
3. **Execute:** Load the file into memory and execute it.
4. **Append:** Write new information at the end of the file.
5. **Delete:** Delete the file and free its space for possible reuse.
6. **List:** List the name and attributes of the file.

(iii) Access Control

- Associate with each file and directory an access-control list (ACL) specifying the user name and the types of access allowed for each user.
- When a user requests access to a particular file, the operating system checks the access list associated with that file. If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs and the user job is denied access to the file.

Directory Implementation

1. Linear List

- The simplest method of implementing a directory is to use a linear list of file names with pointer to the data blocks.
- A linear list of directory entries requires a linear search to find a particular entry.
- This method is simple to program but time- consuming to execute. To create a new file, we must first search the but time – consuming to execute.
- The real disadvantage of a linear list of directory entries is the linear search to find a file.

2. Hash Table

- In this method, a linear list stores the directory entries, but a hash data structure is also used.
- The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.
- Therefore, it can greatly decrease the directory search time.
- Insertion and deleting are also fairly straight forward, although some provision must be made for collisions
 - situation where two file names hash to the same location.
- The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.

I/O Systems

I/O Hardware

The role of the operating system in computer I/O is to manage and control I/O operations and I/O devices. A device communicates with a computer system by sending signals over a cable or even through the air.

Port: The device communicates with the machine via a connection point (or port), for example, a serial port.

Bus: If one or more devices use a common set of wires, the connection is called a bus.

Daisy chain: Device A' has a cable that plugs into device B', and device B' has a cable that plugs into device C', and device C' plugs into a port on the computer, this arrangement is called a daisy chain. A daisy chain usually operates as a bus.

Interrupts

The CPU hardware has a wire called the interrupt-request line . The basic interrupt mechanism works as follows;

1. Device controller raises an interrupt by asserting a signal on the interrupt request line.
2. The CPU catches the interrupt and dispatches to the interrupt handler and
3. The handler clears the interrupt by servicing the device.

Two interrupt request lines:

1. **Nonmaskable interrupt:** which is reserved for events such as unrecoverable memory errors?
2. **Maskable interrupt:** Used by device controllers to request service