UNIT-V
Secondary-Storage Structure

Storage Structure in Operating Systems:

- Computer Storage contains many computer components that are used to store data. It is traditionally divided into primary storage, secondary storage. Details about these storage types and devices used in them are as follows −

1. Primary Storage

- Primary storage is also known as the main memory and is the memory directly accessible by the CPU. Some primary storage devices are − RAM, ROM, Cache memory.

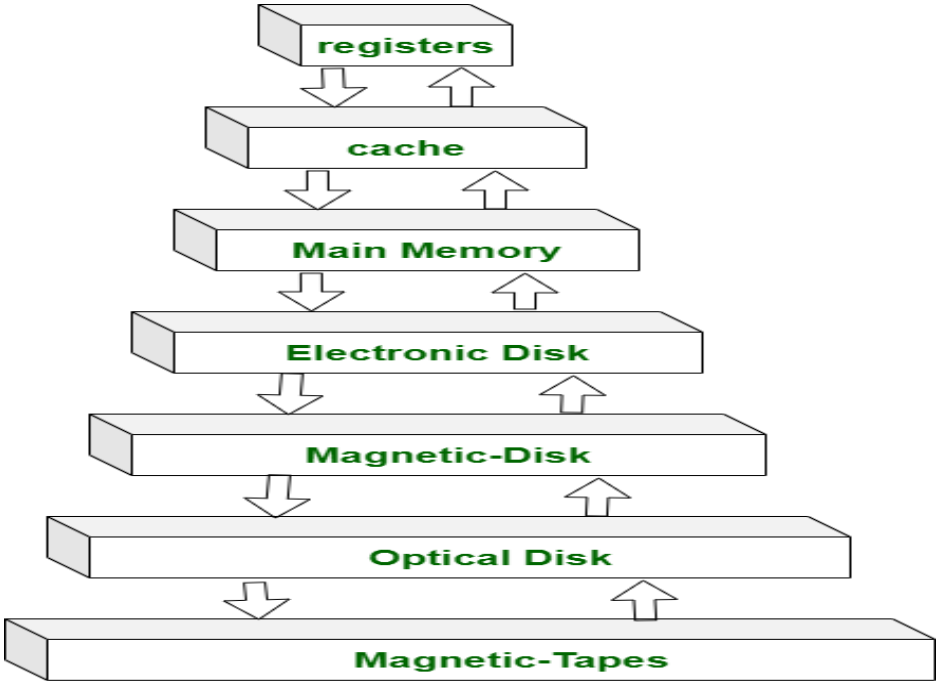
2. Secondary Storage

- Secondary or external storage is not directly accessible by the CPU. The data from secondary storage needs to be brought into the primary storage before the CPU can use it. Secondary storage contains a large amount of data permanently. The different types of secondary storage devices are − Hard disk, floppy disk, Memory cards, CDs etc.

3. Tertiary Storage

- This provides a third level of storage. Most of the rarely used data is archived in tertiary storage as it is even slower than primary storage. Tertiary storage stores a large amount of data that is handled and retrieved by machines, not humans. The different tertiary storage devices are −

- Tape Libraries, Optical Jukeboxes



Storage Device Hierarchy

Secondary Storage Structure:

Secondary storage devices are those devices whose memory is non volatile, the stored data will be intact even if the system is turned off. Here are a few things worth noting about secondary storage.

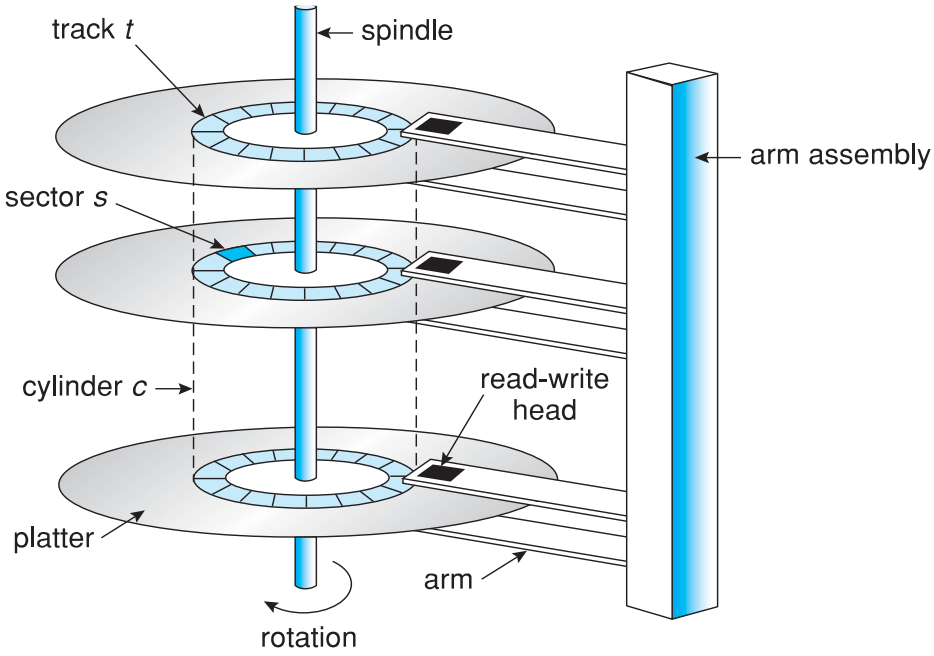
- Secondary storage is also called auxiliary storage.
- Secondary storage is less expensive when compared to primary memory like RAMs.
- The speed of the secondary storage is also lesser than that of primary storage.
- Hence, the data which is less frequently accessed is kept in the secondary storage.
- A few examples are magnetic disks, magnetic tapes, removable thumb drives etc.

Overview of Magnetic Disk structure:

- In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.
- A magnetic disk contains several **platters**. Each platter is divided into circular shaped **tracks**. The length of the tracks near the centre is less than the length of the tracks farther from the centre. Each track is further divided into **sectors**, as shown in the figure.
- Tracks of the same distance from centre form a cylinder. A read-write head is used to read data from a sector of the magnetic disk.

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate:** This is the rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**

Structure of a magnetic disk:



- The speed of the disk is measured as two parts:
  - Transfer rate: This is the rate at which the data moves from disk to the computer.
  - Random access time: It is the sum of the seek time and rotational latency.
- Seek time is the time taken by the arm to move to the required track.
- Rotational latency is defined as the time taken by the arm to reach the required sector in the track.
- Disks can be removable
- Drive attached to computer via **I/O bus**

Disk Scheduling:

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- **Disk Scheduling**: The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.
- Access time has two major components
- It will try to Minimize seek time
- Seek time ≈ seek distance
- Seek Time is defined as the time required by the read/write head to move from one track to another.
- The seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector.
- 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
- Several algorithms exist to schedule the servicing of disk I/O requests .
- There are many sources of disk I/O request
    - OS
    - System processes
    - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
    - Optimization algorithms only make sense when a queue exists

Some important terms related to disk scheduling.

- **Seek Time :** Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

- **Rotational Latency**: It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

- **Transfer Time** :It is the time taken to transfer the data.

- **Disk Access Time** :Disk access time is given as,
  Disk Access Time = Rotational Latency + Seek Time + Transfer Time

- **Disk Response Time**
  It is the average of time spent by each request waiting for the IO operation.

Purpose of Disk Scheduling:

- The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm:

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

- The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.
    - FCFS scheduling algorithm
    - SSTF (shortest seek time first) algorithm
    - SCAN scheduling
    - C-SCAN scheduling
    - LOOK Scheduling
    - C-LOOK scheduling

Note that drive controllers have small buffers and can manage a queue of I/O requests

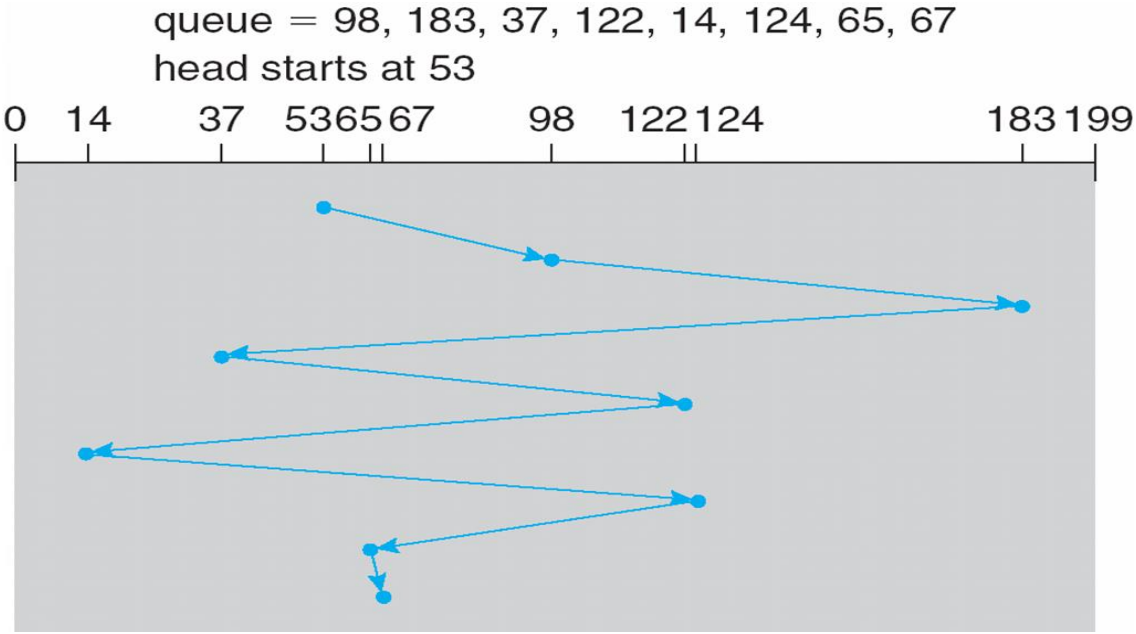
- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate scheduling algorithms with a request queue (0-199)
  98, 183, 37, 122, 14, 124, 65, 67
  Head pointer 53

1)FCFS Disk Scheduling Algorithm:

- Simplest form of disk scheduling - first-come, first-served (F C F S) algorithm.
- Fair, but it generally does not provide the fastest service.
- In FCFS, the request which comes fist will be served first.
- In FCFS, the requests are addressed in the order they arrive in the disk queue.

Example:

A disk queue with requests for I / O to blocks on cylinders in that order.
98, 183, 37, 122, 14, 124, 65, 67,
queue~ 98, 183, 37, 122, 14, 124, 65, 67 head starts at 53



In FCFS, the scheduling is done as follows:

- If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65, and finally to 67, for a total head movement of 640 cylinders.
- The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule.
- If the requests for cylinders 37 and 14 could be serviced together, before or after the requests for 122 and 124, the total head movement could be decreased substantially, and performance could be thereby improved.

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

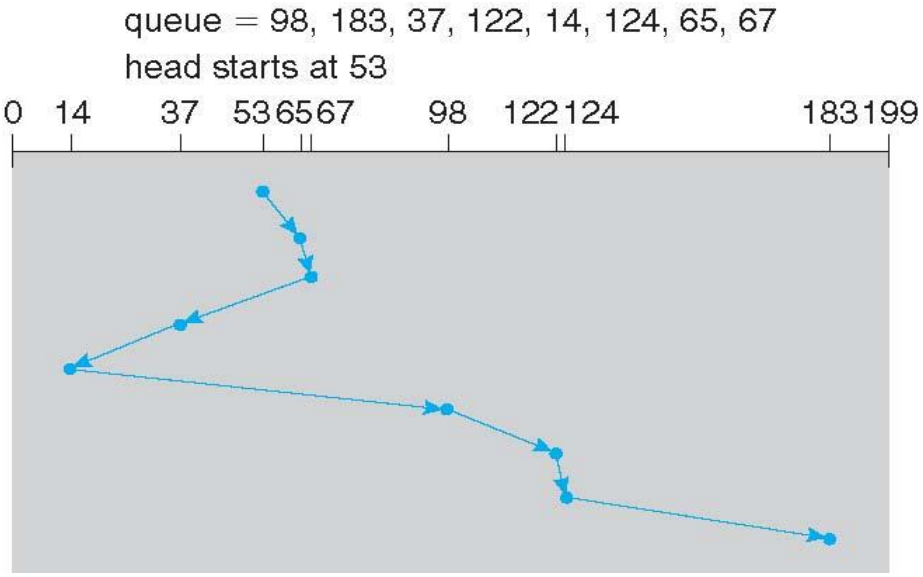
- Does not try to optimize seek time
- May not provide the best possible service

SSTF Scheduling:

- Shortest Seek Time First(SSTF) selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system. Let us understand this with the help of an example.

- It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests. • This assumption is the basis for the shortest-seek-time-first (SSTF) algorithm.
- SSTF chooses the pending request closest to the current head position

Example:
98, 183, 37, 122, 14, 124, 65, 67,
queue~ 98, 183, 37, 122, 14, 124, 65, 67 head starts at 53
Consider the above request queue, the closest request to the initial head position (53) is at cylinder 65.



- Example: Consider the above request queue, the closest request to the initial head position(53) is at cylinder 65.
- Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. In this way nearest request are served. Continuing, it services the request at cylinder 14, then 98, 122, 124, and finally 183. This scheduling method results in a total head movement of only 236 cylinders—little more than one-third of the distance needed for FCFS scheduling of this request queue.

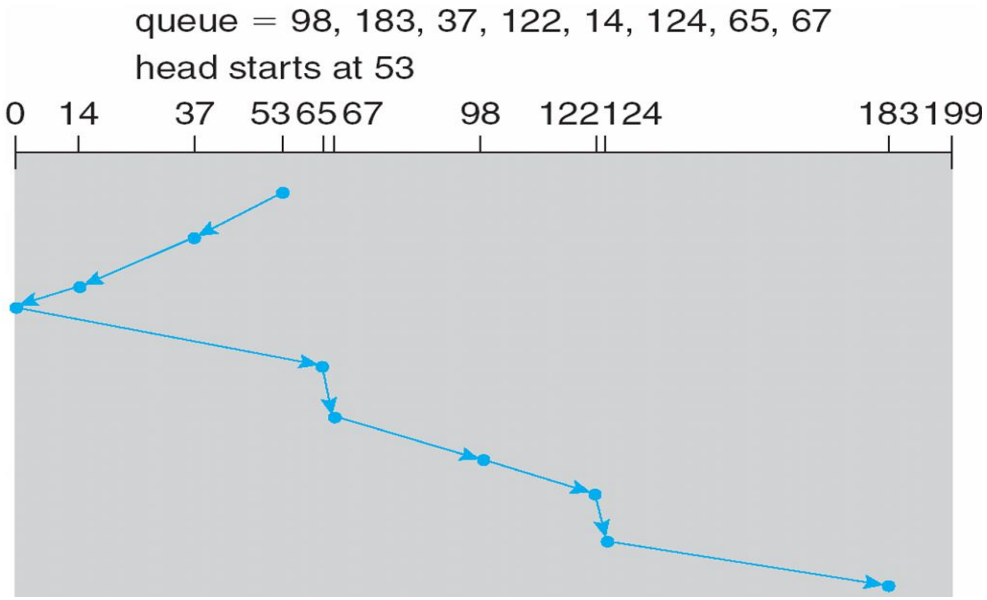
Advantages:
- Average Response Time decreases
- Throughput increases

Disadvantages:
- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests.

SCAN Scheduling Algorithm:

- In this, The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk. After reaching the other end of disk, it reverses its direction and again services the request arriving in its path
- The head continuously scans back and forth across the disk.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Example:
98, 183, 37, 122, 14, 124, 65, 67,
queue~ 98, 183, 37, 122, 14, 124, 65, 67 head starts at 53.
Apply Scan Algorithm to above example



Assuming that the disk arm is moving toward 0 and that the initial head position is again 53, the head will next service 37 and then 14. After reaching end, it reverses direction and services all requests in that path.

At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183. • If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

C-SCAN Scheduling Algorithm:

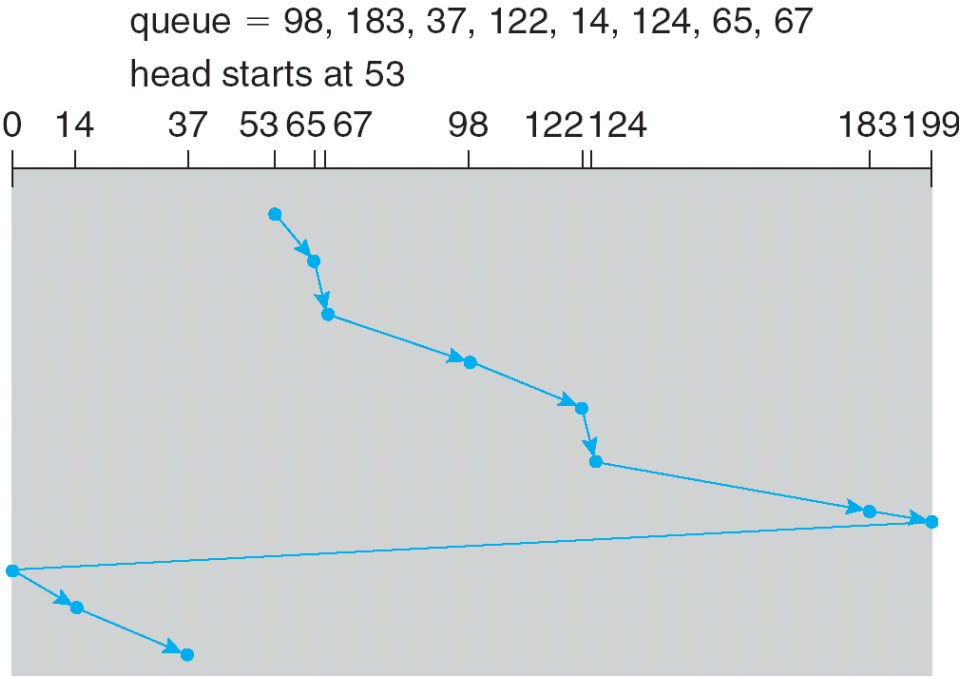
- Circular S C A N (C - S C AN) scheduling is a variant of SCAN designed to provide a more uniform wait time than SCAN
- In this algorithm, the head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- The C - S C A N scheduling algorithm essentially, treats the cylinders as a circular list that wraps around from the last cylinder to the first one

Advantages:

- Provides more uniform wait time compared to SCAN

Example:

- 98, 183, 37, 122, 14, 124, 65, 67,
- queue~ 98, 183, 37, 122, 14, 124, 65, 67 head starts at 53. Apply C-Scan Algorithm to above example



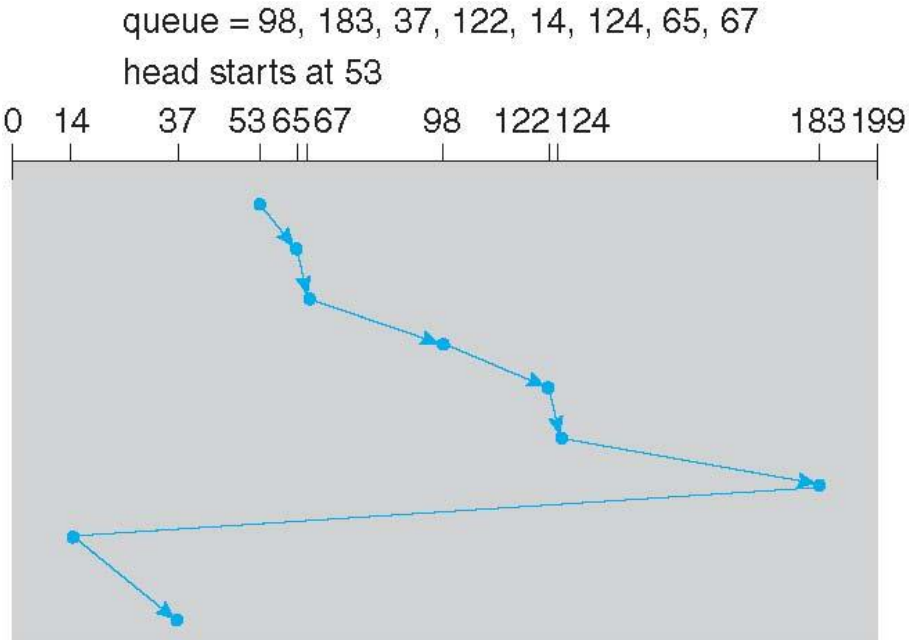
LOOK and C-LOOK Scheduling:

- Both SCAN and C-SCAN move the disk arm across the full width of the disk(to the end of disks). In practice, neither algorithm is often implemented this way.
- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- In these algorithms, the arm only goes as far as the last request in each direction, then it reverses direction immediately, without first going all the way to the end of the disk

- Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

- 98, 183, 37, 122, 14, 124, 65, 67,
- queue~ 98, 183, 37, 122, 14, 124, 65, 67 head starts at 53. Apply C-LOOK Algorithm to above example



Selecting a Disk-Scheduling Algorithm:

- With any scheduling algorithm, performance depends heavily on the number and types of requests.
- Requests for disk service can be greatly influenced by the file-allocation method. A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement. A linked or indexed file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement.
- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm

- The location of directories and index blocks is also important. Since every file must be opened to be used, and opening a file requires searching the directory structure, the directories will be accessed frequently.
- Caching the directories and index blocks in main memory can also help to reduce disk-arm movement, particularly for read requests.

- The scheduling algorithms described here consider only the seek distances. For modern disks, the rotational latency can be nearly as large as the average seek time.
- It is difficult for the operating system to schedule for improved rotational latency, though, because modern disks do not disclose the physical location of logical bloc

RAID Structure:

- RAID, or "Redundant Arrays of Independent Disks" is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both. The term was coined by David Patterson, Garth A. Gibson, and Randy Katz at the University of California, Berkeley in 1987.
- Why data redundancy?
- Data redundancy, although taking up extra space, adds to disk reliability. This means, in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation. On the other hand, if the data is spread across just multiple disks without the RAID technique, the loss of a single disk can affect the entire data.

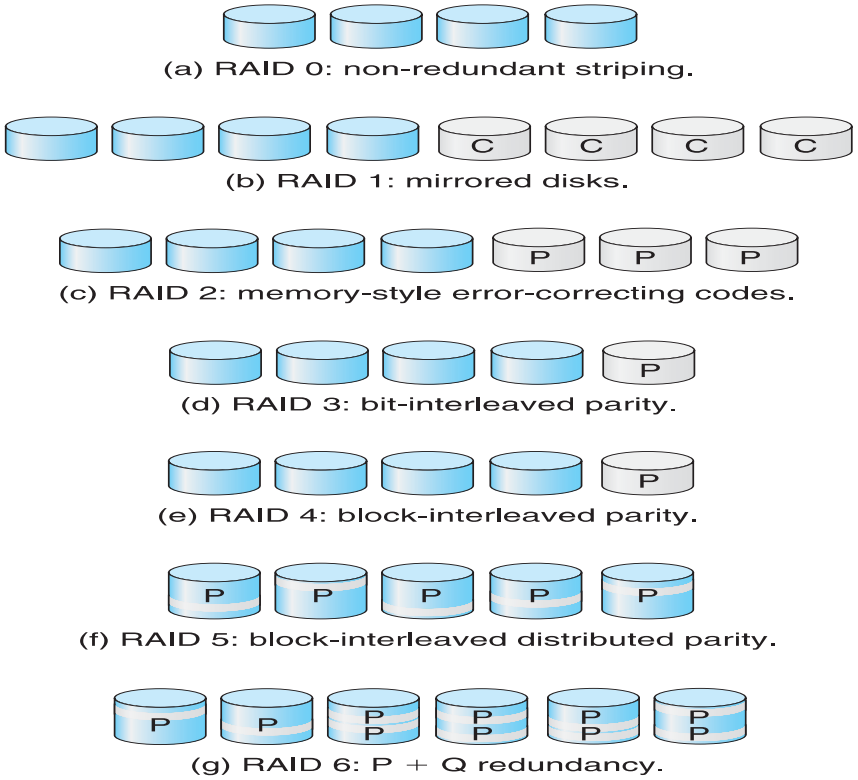
How RAID Works

- RAID works by placing data on multiple disks and allowing input/output operations to overlap in a balanced way, improving performance. Because various disks increase the mean time between failures (MTBF), storing data redundantly also increases fault tolerance.
- RAID arrays appear to the operating system as a single logical drive. RAID employs the techniques of disk mirroring or disk striping.
- Disk Mirroring will copy identical data onto more than one drive.
- Disk Striping partitions help spread data over multiple disk drives. Each drive's storage space is divided into units ranging from 512 bytes up to several megabytes. The stripes of all the disks are interleaved and addressed in order.
- Disk mirroring and disk striping can also be combined in a RAID array.

RAID Levels:

- RAID is arranged into six different levels
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common
- Disk **striping** uses a group of disks as one storage unit

- Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them
-



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

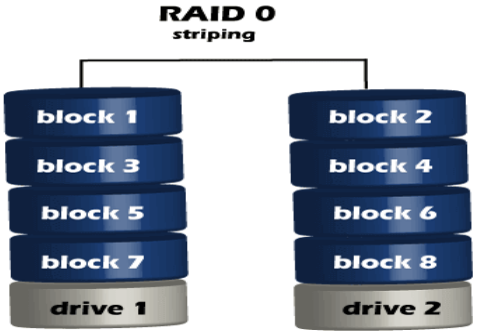
(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

RAID Level 0: Non redundant:

- Data striping is used in level 0 for increase performance but no redundant information is maintained. Striping is done at block level but without any redundancy. Writing performance is best in this level because due to absence of redundant information there is no need to update redundant information.

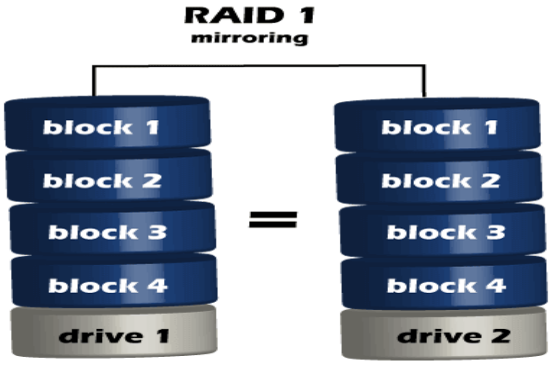


RAID Level 1: Mirrored Disks:

- In this level same data is copied on two different disks. This type of redundancy is called as mirroring. It is the most expensive system. Because two copies of same data are available in two different disks, it allows parallel read. If we lost one disk, you can still stay up and running off the other disks.
- It duplicates data across two disks in the array, providing full redundancy. Both disks are store exactly the same data, at the same time, and at all times. Data is not lost as

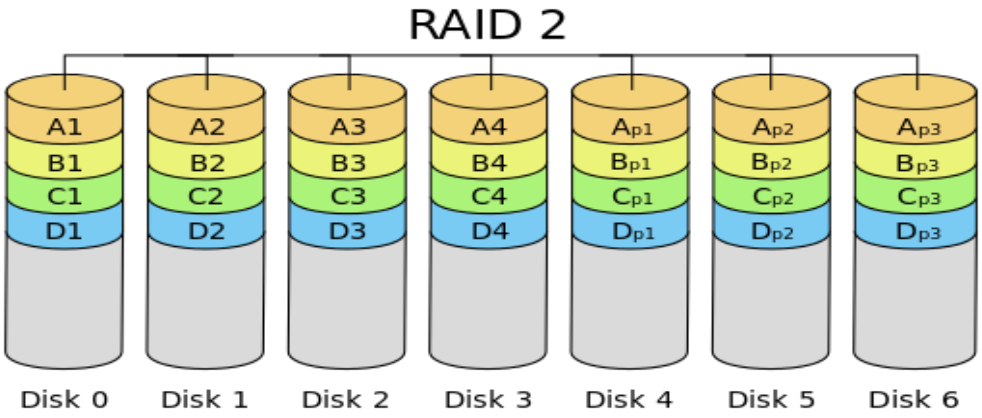
long as one disk survives. The total capacity of the array equals the capacity of the smallest disk in the array. At any given instant, the contents of both disks in the array are identical .

- RAID 1 is capable of a much more complicated configuration. The point of RAID 1 is primarily for redundancy. If you completely lose a drive, you can still stay up and running off the other drive.



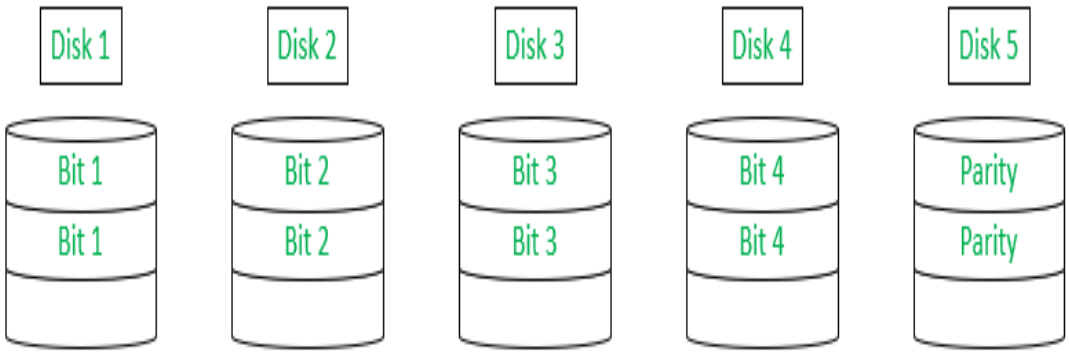
RAID Level 2: Error correcting codes.

This level uses bit-level data stripping in place of block level. It is used with drives with no built in error detection technique. Error-correcting schemes (ECC) store two or more extra bits and it is used for reconstruction of the data if a single bit is damaged.



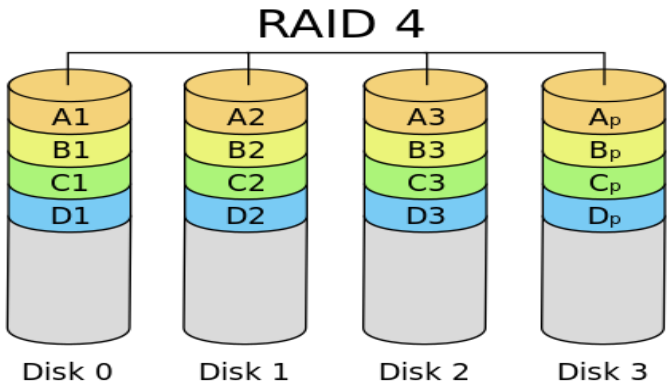
Level 3: Bit-Interleaved parity

- In this level a single parity bit is used for error correction as well as for detection. Systems have disk controller that detects which disk has failed. Check disks does not contains the information about failed disks. RAID level 3 has a single check disk with parity bit.



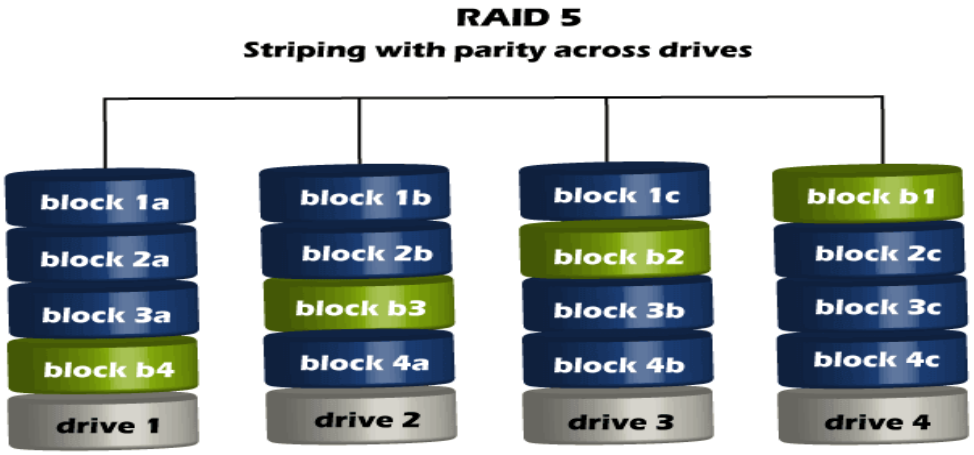
Level 4: Block-Interleaved parity:

- RAID level 4 is similar as RAID level 3 but it has Block-Interleaved parity instead of bit parity. You can access the data independently so read performance is high.



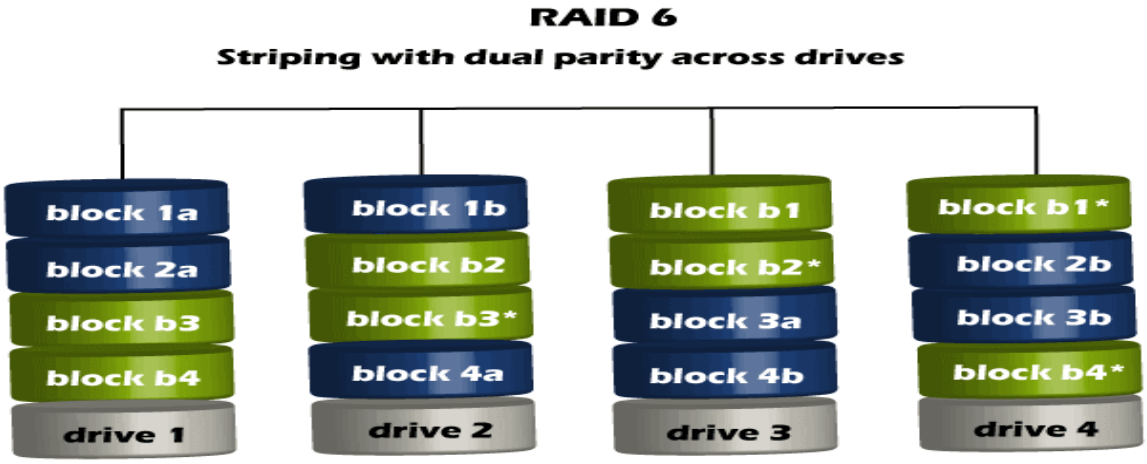
Level 5: Block-Interleaved distributed parity

- RAID level 5 distributes the parity block and data on all disks. For each block, one of the disks stores the parity and the others store data. RAID level 5 gives best performance for large read and write.



Level 6: P+Q Redundancy

- What happen if more than one disk fails at a time? This level stores extra redundant information to save the data against multiple disk failures. It does not use parity but uses reed-solomon codes for data recovery.
- RAID 6 is similar to RAID 5, but the parity data are written to two drives. The use of additional parity enables the array to continue to function even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 has a slower write performance than RAID 5.



Nested RAID levels

Some RAID levels are referred to as nested RAID because they are based on a combination of RAID levels, such as:

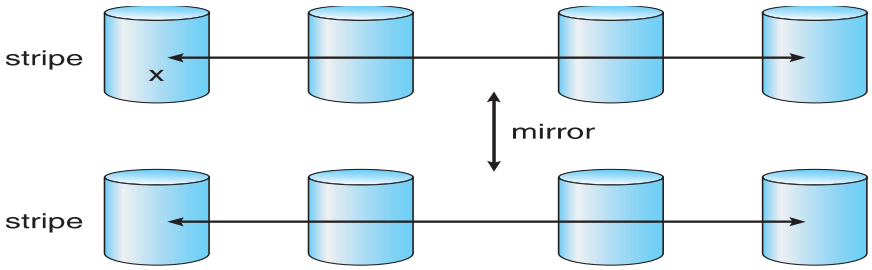
RAID (0 + 1) and (1 + 0):

RAID Level 1+0:

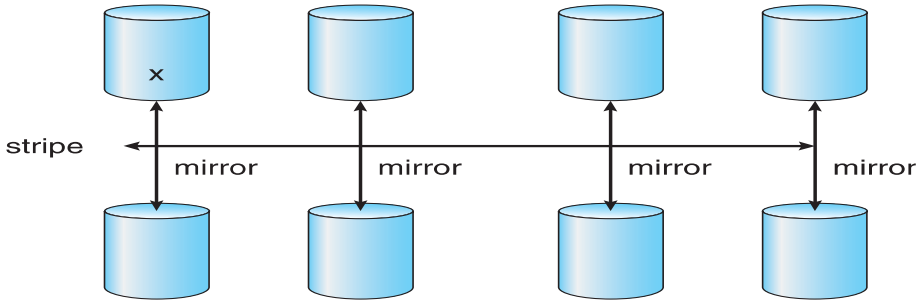
- Striping and Mirroring
- This level is combination of level 0 and level 1, sometimes called as level 10.

RAID Level 0+ 1:

- Mirroring and Striping
- RAID 0+1 is similar to RAID 1+0, except the data organization method is slightly different. Rather than creating a mirror and then striping the mirror, RAID 0+1 creates a stripe set and then mirrors the stripe set.



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

Drawbacks of RAID

- Nested RAID levels are more expensive to implement than traditional RAID levels because they require many disks.
- The cost per gigabyte of storage devices is higher for nested RAID because many of the drives are used for redundancy.
- Some RAID levels, such as RAID 1 and 5, can only sustain a single drive failure.
- RAID arrays are in a vulnerable state until a failed drive is replaced and the new disk is populated with data.
- When RAID was implemented, it takes a lot longer to rebuild failed drives because drives have much greater capacity.

Disk Management:

- The operating system is also responsible for several other aspects of disk management. Disk Formatting
- A new magnetic disk is a blank slate.
- It is just a platter of a magnetic recording material.
- Before a disk can store data, it must be divided into sectors that the disk controller can read and write.
- This process is called low-level formatting, or physical formatting
- Low-level formatting fills the disk with a special data structure for each sector.
- The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
- The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC)

- When the controller writes a sector of data during normal I / O , the E C C is updated with a value calculated from all the bytes in the data area. When the sector is read, the E C C is recalculated and compared with the stored value.
- If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad
- The ECC is an error-correcting code because it contains enough information, if only a few bits of data have been corrupted, to enable the controller to identify which bits have changed and calculate what their correct values should be.
- It then reports a recoverable soft error.
- The operating system still needs to record its own data structures on the disk. • It does so in two steps.
- The first step is to partition the disk into one or more groups of cylinders.
- The operating system can treat each partition as though it were a separate disk. For instance, one partition can hold a copy of the operating system's executable code, while another holds user files.
- The second step is logical formatting, or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk.

Stable-Storage Implementation :

To achieve such storage, we need to replicate the required information on multiple storage devices with independent failure modes. The writing of an update should be coordinate in such a way that it would not delete all the copies of the state and that, when we are recovering from a failure, we can force all the copies to a consistent and correct value, even if another failure occurs during the recovery.
The disk write operation results to one of the following outcome:

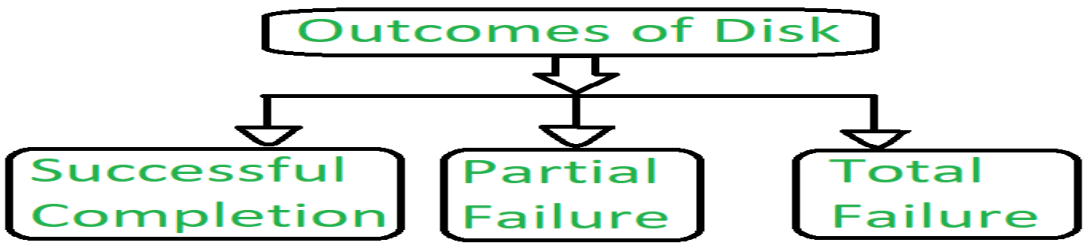
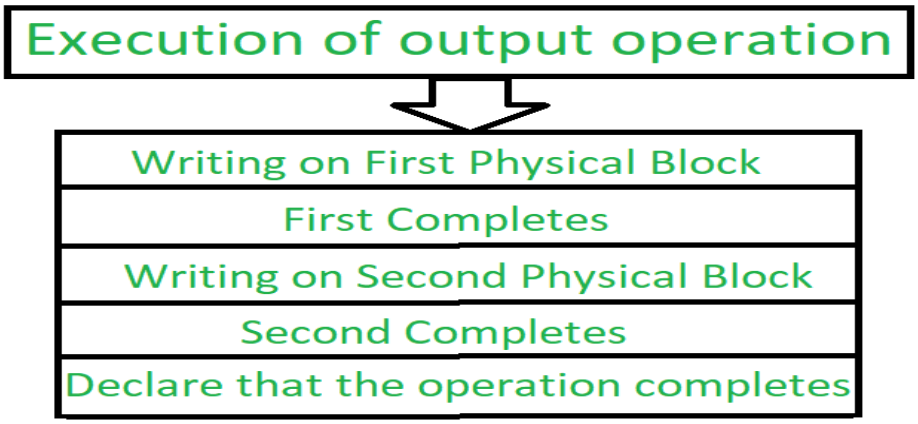


Figure – Outcomes of Disk

1. **Successful completion –**The data will written correctly on disk.
2. **Partial Failure –**In this case, failure is occurred in the middle of the data transfer, so that only some sectors were written with the new data, and the sector which is written during the failure may have been corrupted.
3. **Total Failure –**The failure occurred before the disk write started, so the previous data values on the disk remains intact.

During writing a block somehow if failure occurs, the system's first work is to detect the failure and then invoke a recovery process to restore the consistent state. To do that, the system must contain two physical block for each logical block.
An output operation is executed as follows:



1. Write the information onto the first physical block.
2. When the first write completes successfully, write the same operation onto the second physical block.
3. When both the operation declares successfully, declare the operation as complete.

During the recovery from a failure each of the physical block is examined. If both are the same and no detectable error exists, then no further action is necessary. If one block contains detectable errors then we replace its content with the value of the other block. If neither block contains the detectable error, but the block differ in content, then we replace the content of first block with the content of the second block.

With the usage of large number of copies, the chances of the failure reduces. Generally, it is usually reasonable to simulate stable storage with only two copies. The data present in the stable storage is safe unless a failure destroys all the copies. The data that is present in the stable storage is guaranteed to be safe unless a failure destroys all the copies.

Since the memory is no-volatile it can be trusted to store the data enroute to the disks. In this way it is considered as a part of the stable storage. Writing to the stable storage is much faster than to disk, so performance is greatly improved.

Protection:

- A mechanism that controls the access of programs, processes, or users to the resources defined by a computer system is referred to as protection. You may utilize protection as a tool for multi-programming operating systems, allowing multiple users to safely share a common logical namespace, including a directory or files.

- It needs the protection of computer resources like the software, memory, processor, etc. Users should take protective measures as a helper to multiprogramming OS so that multiple users may safely use a common logical namespace like a directory or data. Protection may be achieved by maintaining confidentiality, honesty and availability in the OS. It is critical to secure the device from unauthorized access, viruses, worms, and other malware.

Need of Protection in Operating System

- Various needs of protection in the operating system are as follows:

1. To prevent the access of unauthorized users and

There may be security risks like unauthorized reading, writing, modification, or preventing the system from working effectively for authorized users.

1. It helps to ensure data security, process security, and program security against unauthorized user access or program access.
2. It is important to ensure no access rights' breaches, no viruses, no unauthorized access to the existing data.
3. Its purpose is to ensure that only the systems' policies access programs, resources, and data.

Goals of Protection:

Operating system consists of a collection of objects, hardware or software. Each object has a unique name and can be accessed through a well-defined set of operations.

Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

1. The policies define how processes access the computer system's resources, such as the CPU, memory, software, and even the operating system. It is the responsibility of both the operating system designer and the app programmer. Although, these policies are modified at any time.
2. Protection is a technique for protecting data and processes from harmful or intentional infiltration. It contains protection policies either established by itself, set by management or imposed individually by programmers to ensure that their programs are protected to the greatest extent possible.
3. It also provides a multiprogramming OS with the security that its users expect when sharing common space such as files or directories.

The most obvious is the need to prevent the mischievous, intentional violation of an access restriction by a user. Of more general importance, however, is the need to ensure that each program component active in a system uses system resources only in ways consistent with stated policies. This requirement is an absolute one for a reliable system.

Principles of Protection:

- Guiding principle – principle of least privilege

- It dictates that programs, users, and even systems should be given just enough privileges to perform their tasks
- An operating system following the principle of least privilege implements its features, programs, system calls, and data structures so that failure or compromise of a component does the minimum damage and allows the minimum damage to be done.

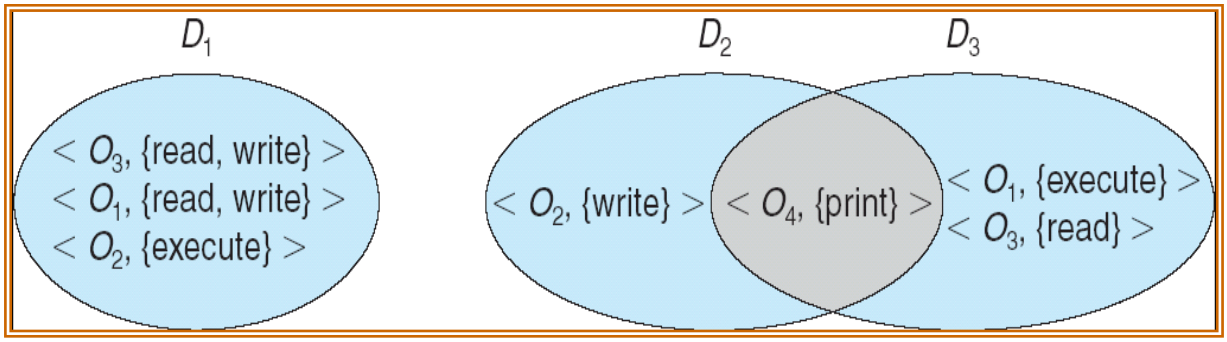
Domain of Protection:

- To facilitate the protection, a process operates within a protection domain, which specifies the resources that the process may access.
- A protection domain specifies the resources that a process may access.
- Each domain defines a set of objects and the types of operations that may be invoked on each object. The ability to execute an operation on an object is an **access right**.
- A domain is a collection of access rights, each of which is an ordered pair <object-name, rights-set>.
- Each domain comprises a collection of objects and the operations that may be implemented on them. A domain could be made up of only one process, procedure, or user. If a domain is linked with a procedure, changing the domain would mean changing the procedure ID. Objects may share one or more common operations.

For example, if domain D has the access right <file F, {read, write}>, then a process executing in domain D can both read and write file F; it cannot, however, perform any other operation on that object.

Domain Structure:

- Access-right = <*object-name*, *rights-set*>
  where *rights-set* is a subset of all valid operations that can be performed on the object.
- Domain = set of access-rights
- Example: Figure shows- System with three protection domains



- In this example: we have three domains: D1, D2, and D3 . The access right < 0_4, {print}> is shared by D2 and D3, implying that a process executing in either of these two domains can print object 0_4. Note that a process must be executing in domain D1 to read and write object 0_1, while only processes in domain D3 may execute object 0_1.

Association between Process and Domain

- When processes have the necessary access rights, they can switch from one domain to another. It could be of two types, as shown below.

1. Fixed or Static

- In a fixed association, all access rights could be given to processes at the start. However, the results in a large number of access rights for domain switching. As a result, a technique of changing the domain's contents is found dynamically.

2. Changing or dynamic

- In dynamic association ,a process can switch dynamically, creating a new domain in the process, if need be.

Domain Implementation (UNIX):

- System consists of 2 domains:
  - User
  - Supervisor
- UNIX
  - Domain = user-id
  - Domain switch accomplished via file system.
    - Each file has associated with it a domain bit (setuid bit).
    - When file is executed and setuid = on, then user-id is set to owner of the file being executed. When execution completes user-id is reset.

Access Matrix:

- Access Matrix is a security model of protection state in computer system. It is represented as a matrix.
- Access matrix is used to define the rights of each process executing in the domain with respect to each object.
- The rows of matrix represent domains and columns represent objects. Each cell of matrix represents set of access rights which are given to the processes of domain means each entry(i, j) defines the set of operations that a process executing in domain Di can invoke on object Oj.
- View protection as a matrix (*access matrix*)
- Rows represent domains
- Columns represent objects

- *Access(i, j)* is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

According to the above matrix: there are four domains and four objects- three files(F1, F2, F3) and one printer. A process executing in D1 can read files F1 and F3. A process executing in domain D4 has same rights as D1 but it can also write on files. Printer can be accessed by only one process executing in domain D2. The mechanism of access matrix consists of many policies and semantic properties. Specifically, We must ensure that a process executing in domain Di can access only those objects that are specified in row i.

Switching between Domains.:

- Processes should be able to switch from one domain to another. Switching from domain D_i to domain D_j is allowed if and only if the access right switch E access(i,j). Thus, in Figure, a process executing in domain D2 can switch to domain D3 or to domain D4 . A process in domain D4 can switch to D1, and one in domain D1 can switch to D2-
- When we switch a process from one domain to another, we execute a switch operation on an object(the domain). We can control domain switching by including domains among the objects of the access matrix. Processes should be able to switch from one domain (Di) to another domain (Dj) if and only is a switch right is given to access(i, j).

Access Matrix of Figure A With Domains as Objects:

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

According to the matrix: A process executing in domain D2 can switch to domain D3 and D4. A process executing in domain D4 can switch to domain D1 and process executing in domain D1 can switch to domain D2.

Use of Access Matrix:

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix.
- Can be expanded to dynamic protection.
  - Operations to add, delete access rights.
  - Special access rights:
    - *owner of* O_i
    - *copy op from* O_i *to* O_j
    - *control –* D_i *can modify* D_j *access rights*
    - *transfer – switch from domain* D_i *to* D_j

- Access matrix design separates mechanism from policy.
  - Mechanism
    - Operating system provides access-matrix + rules.
    - If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.
  - Policy
    - User dictates policy.
    - Who can access what object and in what mode.

Implementation of Access Matrix:

- Each column = Access-control list for one object
  Defines who can perform what operation.
  Domain 1 = Read, Write
  Domain 2 = Read

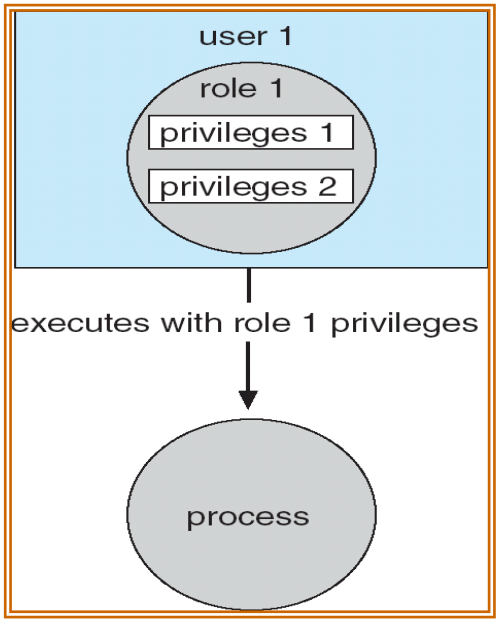
Domain 3 = Read
⋮

- Each Row = Capability List (like a key)
  Fore each domain, what operations allowed on what objects.
  Object 1 – Read
  Object 4 – Read, Write, Execute
  Object 5 – Read, Write, Delete, Copy

Access Control:

- Access control for an operating system determines how the operating system implements accesses to system resources by satisfying the security objectives of integrity, availability, and secrecy. Such a mechanism authorizes subjects (e.g., processes and users) to perform certain operations (e.g., read, write) on objects and resources of the OS (e.g., files, sockets).
- Operating system is the main software between the users and the computing system resources that manage tasks and programs. Resources may include files, sockets, CPU, memory, disks, timers, network, etc. System administrators, developers, regular users, guests, and even hackers could be the users of the computing system. Information security is an important issue for an operating system
- Protection can be applied to non-file resources
- Solaris 10 provides **role-based access control** to implement principle of least privilege
  - Privilege is right to execute system call or use an option within a system call
  - Privileges Can be assigned to processes with limited access.
  - Privileges and programs can also be assigned to roles of users.

Role-based Access Control in Solaris 10:



Revocation of Access Rights:

- In a dynamic protection system, we may sometimes need to revoke access rights to objects shared by different users.
- Since the capabilities are distributed throughout the system, we must find them before we can revoke them.
- *Access List* – Delete access rights from access list.
  - Simple
  - Immediate
- *Capability List* – Scheme required to locate capability in the system before capability can be revoked.
  - Reacquisition
  - Back-pointers
  - Indirection
  - Keys

Schemes that implement revocation for capabilities include the following:

• **Reacquisition**: Periodically, all capabilities are deleted from each domain. If a process wants to use a capability, it may find that that capability has been deleted. The process may then try to reacquire the capability. If access has been revoked, the process will not be able to Reacquire the capability.

• **Back-pointers:** A list of pointers is maintained with each object, pointing to all capabilities associated with that object. When revocation is required, we can follow thesepointers, these pointers, changing the capabilities as necessary

• **Indirection**: The capabilities point indirectly to the objects. Each capability points to a unique entry in a global table, which in turn points to the object. We implement revocation by searching the global table for the desired entry and deleting it. Then, when an access is attempted, the capability is found to point to an illegal table entry.

• **Keys:** A key is a unique bit pattern that can be associated with a capability. This key is defined when the capability is created, and it can be neither modified nor inspected by the process owning the capability. A master key is associated with each object; it can be defined or replaced with the set-key operation.

When a capability is created, the current value of the master key is associated with the capability. When the capability is exercised, its key is compared with the master key. If the keys match, the operation is allowed to continue; otherwise, an exception condition is raised.

In key-based schemes, the operations of defining keys, inserting them into lists, and deleting them from lists should not be available to all users.

Operating System Security:

- The process of ensuring OS availability, confidentiality, integrity is known as operating system security.
- OS security refers to the processes or measures taken to protect the operating system from dangers, including viruses, worms, malware, and remote hacker intrusions.
- Operating system security comprises all preventive-control procedures that protect any system assets that could be stolen, modified, or deleted if OS security is breached.
- Security refers to providing safety for computer system resources like software, CPU, memory, disks, etc. It can protect against all threats, including viruses and unauthorized access. It can be enforced by assuring the operating system's integrity, confidentiality, and availability. If an illegal user runs a computer application, the computer or data stored may be seriously damaged.

The Security Problem:

- Security must consider external environment of the system, and protect the system resources
- Intruders (crackers) attempt to breach security
- **Threat** is potential security violation
- **Attack** is attempt to breach security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse

- System security may be threatened through two violations, and these are as follows:
- **1. Threat**

A program that has the potential to harm the system seriously.

- **2. Attack**

A breach of security that allows unauthorized access to a resource. An attack is the attempt to break security.

Security Violations:

There are two types of security breaches or violations that can harm the system: malicious and accidental. Malicious threats are a type of destructive computer code or web script that is designed to cause system vulnerabilities that lead to back doors and security breaches. On the other hand, Accidental Threats are comparatively easier to protect against.

Security may be compromised through the breaches. Some of the breaches are as follows:

1. Breach of integrity

This violation has unauthorized data modification.

2. Theft of service

It involves the unauthorized use of resources.

3. Breach of confidentiality

It involves the unauthorized reading of data.

4. Breach of availability

It involves the unauthorized destruction of data.

5. Denial of service

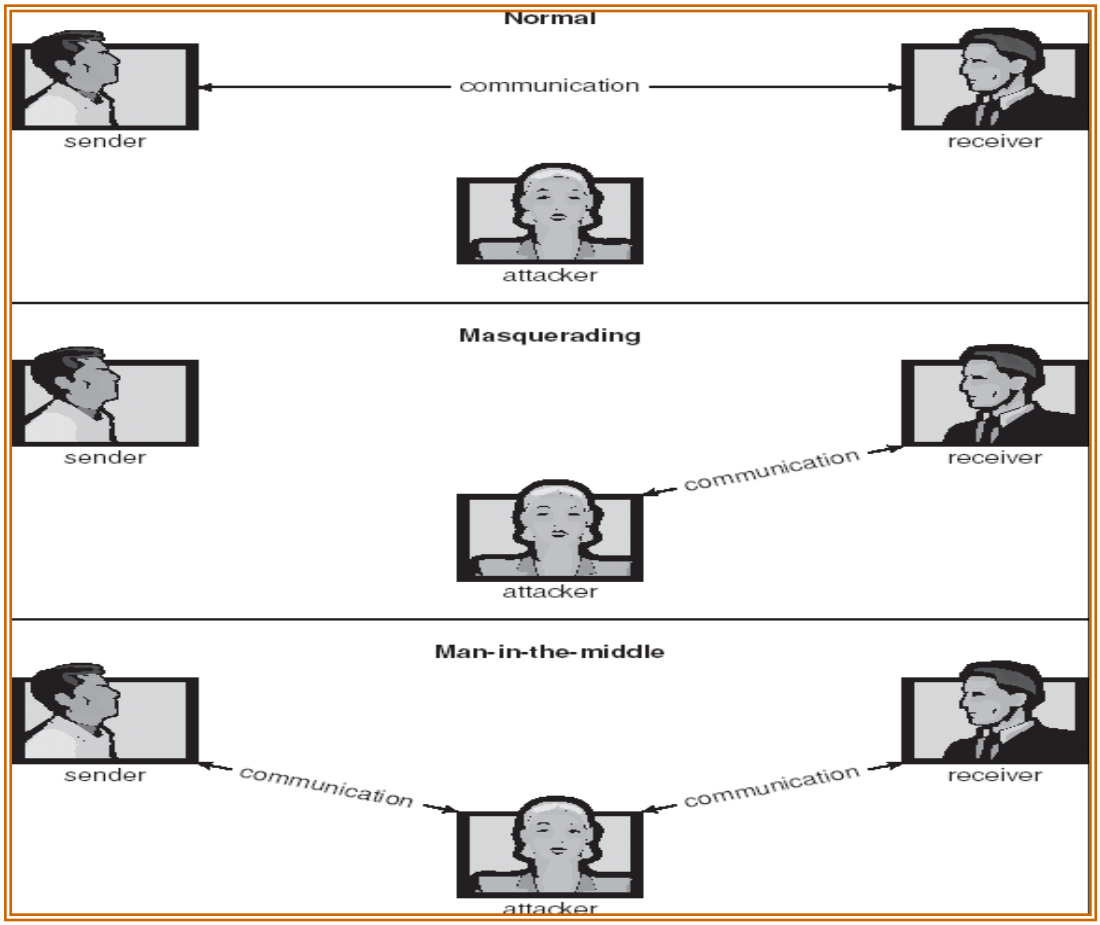
It includes preventing legitimate use of the system. Some attacks may be accidental.

Methods:

Some attacks methods are

- Masquerading (breach authentication)
- Replay attack
  - Message modification
- Man-in-the-middle attack
- Session hijacking

Standard Security Attacks:



The goals of Security System:

There are several goals of system security. Some of them are as follows:

1. Integrity

Unauthorized users must not be allowed to access the system's objects, and users with insufficient rights should not modify the system's critical files and resources.

2. Secrecy

The system's objects must only be available to a small number of authorized users. The system files should not be accessible to everyone.

3. Availability

All system resources must be accessible to all authorized users, i.e., no single user/process should be able to consume all system resources. If such a situation arises, service denial may occur. In this case, malware may restrict system resources and preventing legitimate processes from accessing them.

Security Measure Levels:

- Security must occur at four levels to be effective: To protect a system, we must take security measures at four levels:
  - Physical
  - Human
    - Avoid **social engineering, phishing, dumpster diving**
  - Operating System
  - Network
- Security is as week as the weakest chain

Program Threats:

- The operating system's processes and kernel carry out the specified task as directed. Program Threats occur when a user program causes these processes to do malicious operations. The common example of a program threat is that when a program is installed on a computer, it could store and transfer user credentials to a hacker. There are various program threats. Some of them are as follows:
- **Trojan Horse**
  - This is a Code segment that misuses its environment
  - This type of application captures user login credentials. It stores them to transfer them to a malicious user who can then log in to the computer and access system resources.
  - Exploits mechanisms for allowing programs written by users to be executed by other users.
  - **Spyware, pop-up browser windows, covert channels**
- **Trap Door**
  - A trap door is when a program that is supposed to work as expected has a security weakness in its code that allows it to do illegal actions without the user's knowledge.

  - It is a Specific user identifier or password that circumvents normal security procedures
  - It could be included in a compiler
- **Logic Bomb**
  - A logic bomb is a situation in which software only misbehaves when particular criteria are met; otherwise, it functions normally.
  - It is a Program that initiates a security incident under certain circumstances
- **Stack and Buffer Overflow**
  - Exploits a bug in a program (overflow either the stack or memory buffers)
  - The stack- or buffer-overflow attack is the most common way for an attacker outside the system, on a network or dial-up connection, to gain unauthorized access to the target system. An authorized user of the system may also use this exploit for privilege escalation.

Program Threats :

Viruses

- Another form of program threat is a virus.
- A virus is a fragment of code embedded in a legitimate program.
- Viruses are self-replicating and are designed to "infect" other programs. They can wreak havoc in a system by modifying or destroying files and causing system crashes and program malfunctions.
- Viruses are very specific to CPU architectures, operating systems, and applications.
- Viruses are a particular problem for users of PCs. UNIX and other multiuser operating systems generally are not susceptible to viruses because the executable programs are protected from writing by the operating system.
- Viruses are usually borne via e-mail, with spam the most common vector. They can also spread when users download viral programs Internet file-sharing services or exchange infected disks.
- Usually borne via email or as a macro

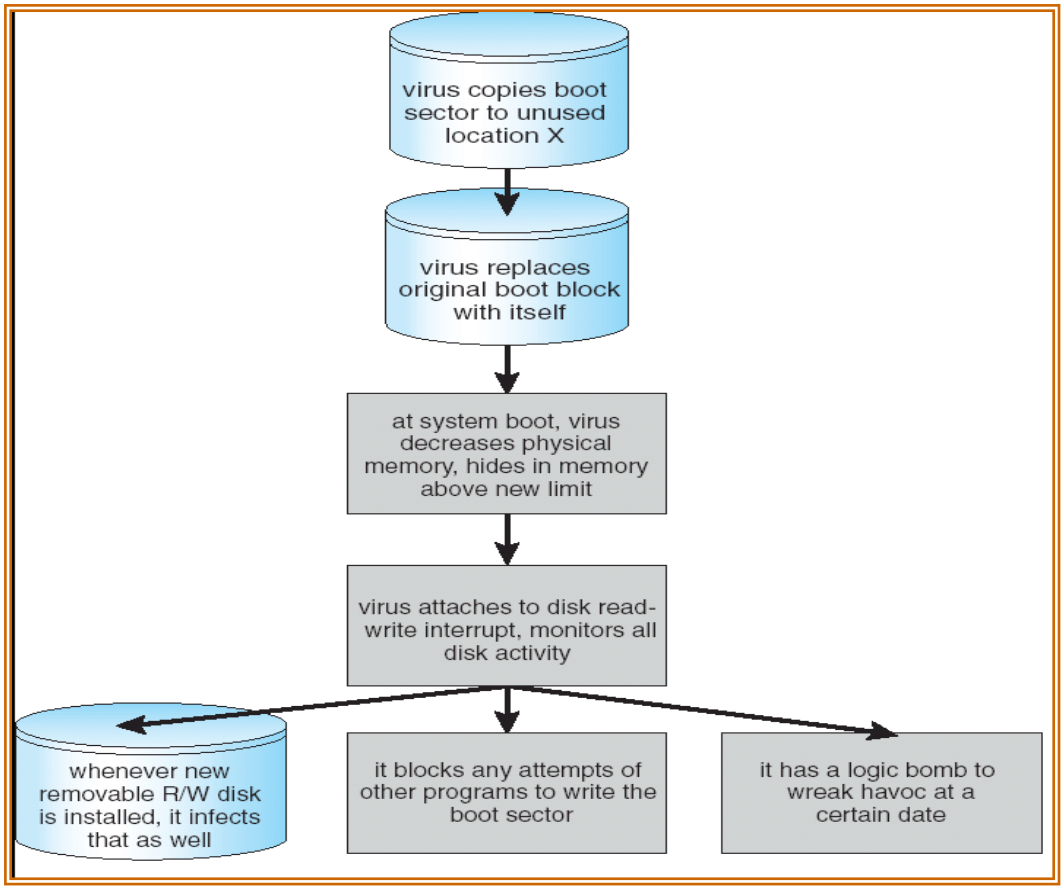
How do viruses work?

- Once a virus reaches a target machine, a program known as a virus dropper inserts the virus into the system.
- The virus dropper is usually a Trojan horse, executed for other reasons but installing the virus as its core activity. Once installed, the virus may do any one of a number of things. There are literally thousands of viruses, but they fall into several main categories. Note that many viruses belong to more than one category.

- **Virus dropper** inserts virus onto the system
- Many categories of viruses, literally many thousands of viruses. Some types of viruses are:

- File virus- A standard file virus infects a system by appending itself to a file. It changes the start of the program so that execution jumps to its code. After it executes, it returns control to the program so that its execution is not noticed.
- Boot virus- A boot virus infects the boot sector of the system, executing every time the system is booted and before the operating system is loaded. It watches for other boatable media (that is, floppy disks) and infects them. These viruses are also known as memory viruses, because they do not appear in the file system.
- Macro virus- Most viruses are written in a low-level language, such as assembly or C. Macro viruses are written in a high-level language, such as Visual Basic. These viruses are triggered when a program capable of executing the macro is run. For example, a macro virus could be contained in a spreadsheet file.
- Source code virus- A source code virus looks for source code and modifies it to include the virus and to help spread the virus.
- Polymorphic- A polymorphic virus changes each time it is installed to avoid detection by antivirus software.
- Encrypted
- Stealth
- Tunneling
- Multipartite
- Armored

A Boot-sector Computer Virus:

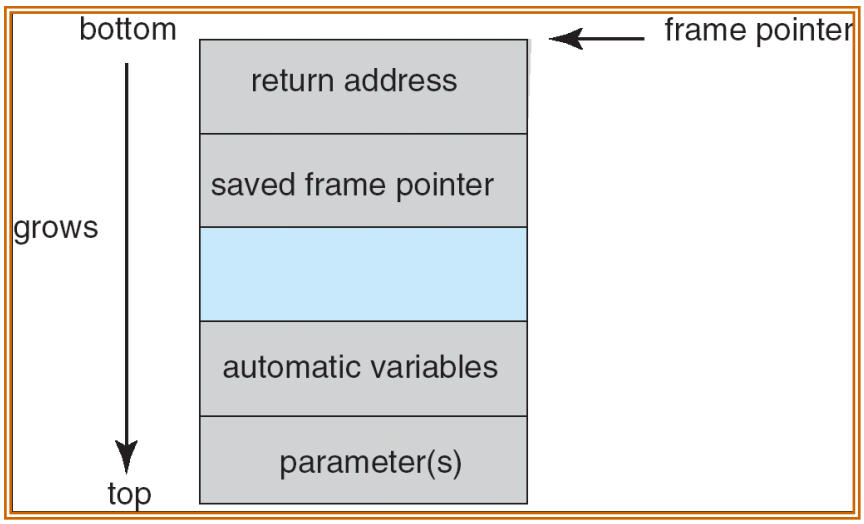


C Program with Buffer-overflow Condition:

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
        char buffer[BUFFER SIZE];
        if (argc < 2)
                return -1;
        else {
                strcpy(buffer,argv[1]);
                return 0;
                }
}
```

This program creates a character array of size BUFFER_SIZE and copies the contents of the parameter provided on the command line-argv [1]. As long as the size of this parameter is less than BUFFER_SIZE (we need one byte to store the null terminator), this program works properly. But consider what happens if the parameter provided on the command line is longer than BUFFER_SIZE. In this scenario, the strcpy () function will begin copying from argv [1] until it encounters a null terminator (\0) or until the program crashes. Thus, this program suffers from a potential buffer overflow problem in which copied data overflow the buffer array.

Layout of Typical Stack Frame:



System and Network Threats:

- System threats are described as the misuse of system services and network connections to cause user problems. These threats may be used to trigger the program threats over an entire network, known as program attacks. System threats make an environment in which OS resources and user files may be misused. There are various system threats. Some of them are as follows:

1. Worm

- The worm is a process that can choke a system's performance by exhausting all system resources. A Worm process makes several clones, each consuming system resources and preventing all other processes from getting essential resources. Worm processes can even bring a network to a halt.
- Internet worm
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - **Grappling hook** program uploaded main worm program

2. Port scanning

- It is a method by which the cracker determines the system's vulnerabilities for an attack. It is a fully automated process that includes connecting to a specific port via TCP/IP. To protect the attacker's identity, port scanning attacks are launched through Zombie Systems, which previously independent systems now serve their owners while being utilized for such terrible purposes.
- It is an Automated attempt to connect to a range of ports on one or a range of IP addresses

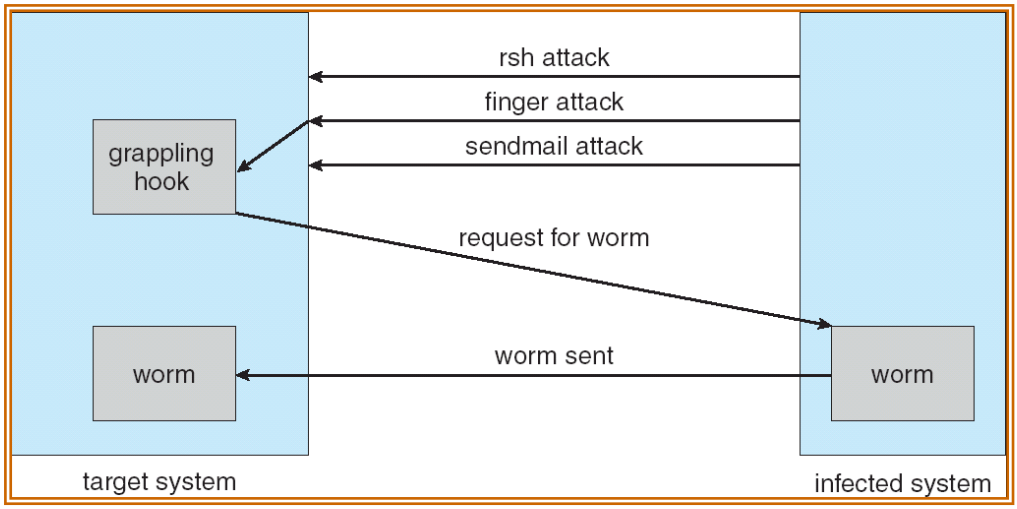
3. Denial of Service

- Denial of service attacks usually prevents users from legitimately using the system. For example, if a denial-of-service attack is executed against the browser's content settings, a user may be unable to access the internet.
- Overload the targeted computer preventing it from doing any useful work
- Distributed denial-of-service (**DDOS**) come from multiple sites at once

Some of types of attacks are:

- **1.Malicious threat:** Malicious threat include Computer viruses, Trojan, worm and spyware.
- **2.DOS attack:** A Denial-of-Service (DOS) attack is an attack intended to close down a machine or network, making it unavailable to its intended users.
- **3.Phishing:** Phishing is the process to gain sensitive information like usernames, passwords and credit card information, frequently for malicious reasons, by taking on .the appearance of a dependable element

The Morris Internet Worm:



Cryptography as a Security Tool:

- Cryptography is a broadest security tool available
  - Source and destination of messages cannot be trusted without cryptography
  - Means to constrain potential senders (*sources*) and / or receivers (*destinations*) of *messages*
- It is based on secrets (**keys**)