Unit 2
Secondary Storage Management, Protection and Security

Structure of the Unit

1. Unit Outcomes
2. Secondary storage management
3. Protection-Goals, Principles, Domain
4. Access Matrix
5. Access Control
6. Self-Assessment Questions
7. Self-Assessment Activities
8. Multiple Choice Questions
9. Keys to Multiple Choice Questions
10. Summary of the Unit
11. Keywords
12. Recommended Resources for Further Reading

2.1 Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Define secondary storage and its role in an operating system.
2. Describe the objectives of protection mechanisms in an operating system.
3. Evaluate the different principles and strategies used to implement protection mechanisms.
4. Analyze the advantages and disadvantages of different access control models.

2.2 Secondary storage management

2.2.1 Disk structure

Magnetic disks are a type of secondary storage that offers a significant amount of storage space. These disks are available in different sizes and speeds, and they store data using magnetism. Each disk platter has a flat, circular shape, similar to a CD, with two surfaces coated in magnetic material. The platter surface is logically divided into circular tracks, which are further divided into sectors - the basic unit of storage. The collection of tracks that are positioned at one arm location makes up a cylinder. The number of cylinders in a disk drive is equal to the number of tracks on each platter, and there may be thousands of concentric cylinders in a disk drive, with each track containing hundreds of sectors. The storage capacity of disk drives is measured in gigabytes. When the disk drive is in use, the head moves from the first track of the disk to the last track, with the disks rotating at a constant speed. To read or write data, the head must be positioned at the desired track and the beginning of the desired sector on that track.

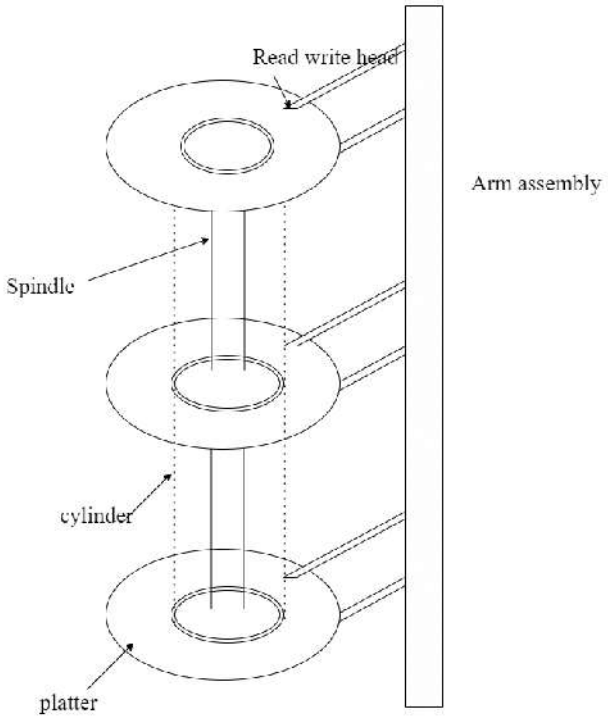

Figure 2.1: Moving head disk mechanism

Figure 2.1 above shows the typical disk structure with the moving head mechanism. The following parameters are defined concerning disk access:

• Seek time refers to the duration required for moving the disk arm to the desired track.

• Rotational latency, on the other hand, represents the time taken by the disk to rotate until the required sector is beneath the read/write head.

• The sum of seek time and rotational latency is known as positioning time or random-access time.

• Meanwhile, disk bandwidth is computed by dividing the total bytes transferred by the time that elapses between the initial service request and the completion of the last transfer.

• Lastly, transfer rate pertains to the speed at which data flows between the drive and the computer.

It is important to note that, since the disk head is sustained by a paper-thin air cushion, it might come into contact with the disk surface, even if it is coated with a thin layer of protection, causing damage to the magnetic surface. This type of accident is referred to as a head crash.

Magnetic tape is a storage medium that is considered a permanent memory and has the ability to hold large amounts of data. However, the access time for retrieving data from magnetic tape is much slower than that of a magnetic disk because data is accessed sequentially. This means that when a particular piece of data needs to be accessed, the tape must first start moving from the beginning and move to the specific location of the data, making it a time-consuming process. Therefore, magnetic tapes are mainly used for backing up infrequently used information.

A disk is divided into tracks and sectors, with sectors being the basic unit for read or write operations. Modern disk drives are addressed as a large one-dimensional array where logical blocks are mapped onto the sectors of the disk sequentially. The disk structure can be of two types: Constant Linear Velocity

(CLV) and Constant Angular Velocity (CAV).

In CLV architecture, the density of bits per track is uniform, but the length of the track increases the farther it is from the center of the disk, allowing it to hold more sectors. As we move from outer zones to inner zones, the number of sectors per track decreases. This architecture is commonly used in CD-ROM and DVD-ROM.

In CAV architecture, there is an equal number of sectors in each track, and the sectors are densely packed in the inner tracks. The density of bits decreases from inner tracks to outer tracks to maintain a constant data rate.

2.2.2 Disk Scheduling Methods

In an operating system, it is essential to use the hardware efficiently, which means reducing the access time and increasing the disk bandwidth. Access time is the time it takes for the disk arm to move to the correct cylinder (seek time) and for the disk to rotate to the correct sector (rotational latency). Disk bandwidth is the amount of data transferred per unit of time.

By managing the order of disk I/O requests, we can improve both the access time and bandwidth. Whenever a process needs to read or write data from the disk, it makes a system call to the operating system, specifying the type of operation (read or write), the disk address, the memory address for the transfer, and the number of sectors to be transferred. If the disk is free, the request is serviced immediately; otherwise, it is added to a queue.

The operating system chooses one of the requests from the queue and serves it when the disk becomes free. Several disk-scheduling algorithms are available to determine which request to serve next. We shall look at them.

2.2.2.1 First-Come, First-Served (FCFS):

The requests are served in the order they arrive in the queue. This method is simple but may result in longer access times for requests at the end of the queue. To understand the algorithm let us consider the following disk queue with requests for I/0 to blocks on cylinders

- 98, 183, 37, 122, 14, 124, 65, 67

We can calculate the number of head movements for each request in the queue using this approach. The total number of head movements is the sum of all the distances between adjacent requests in the queue.

Assume that the head is initially at cylinder 53. Head movements are shown in Figure 2.2. We will calculate the number of head movements.

To calculate the number of head movements, we need to find the distance between each request and the previous one in the queue. The distance is measured in the number of cylinders that the head needs to move. For example, to move from cylinder 53 to cylinder 98, the head needs to move 45 cylinders. Similarly, to move from cylinder 98 to cylinder 183, the head needs to move 85 cylinders.

- 53 to 98 = 45 cylinders
- 98 to 183 = 85 cylinders
- 183 to 37 = 146 cylinders
- 37 to 122 = 85 cylinders

- 122 to 14 = 108 cylinders
- 14 to 124 = 110 cylinders
- 124 to 65 = 59 cylinders
- 65 to 67 = 02 cylinders

In all 45 + 85 + 146 + 85 + 108 + 110 + 59 + 02 = 640 Cylinders.
In this example, the total number of head movements is 640 cylinders. The head moves from cylinder 53 to cylinder 98 (45 cylinders), then to cylinder 183 (85 cylinders), then to cylinder 37 (146 cylinders), and so on until it reaches cylinder 67 (2 cylinders).

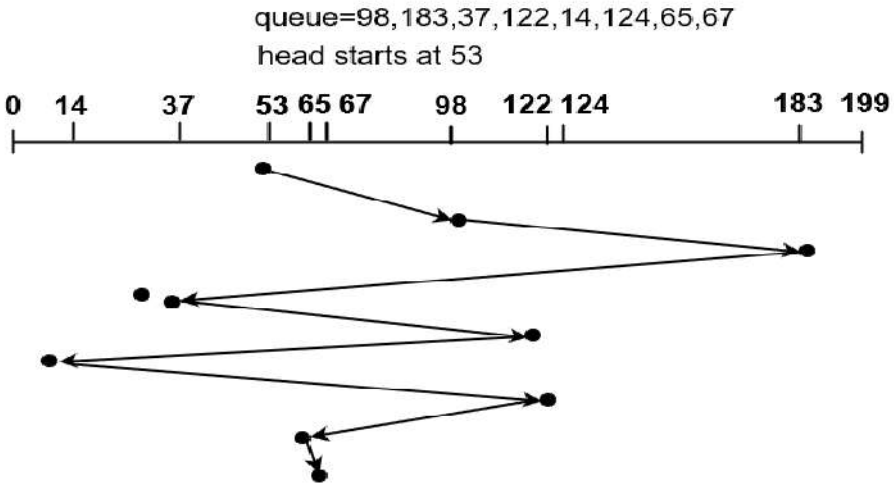

Figure 2.2: FCFS Scheduling

One drawback of the FCFS disk scheduling algorithm is that it may result in longer access times for requests at the end of the queue, as in this example. The disk head moves back and forth between cylinders 122, 14, and 124, increasing the total head movement. To reduce the head movement, we can consider other disk scheduling algorithms, such as Shortest Seek Time First (SSTF) or SCAN.

2.2.2.2 Shortest Seek Time First (SSTF)

Shortest Seek Time First (SSTF) is a disk scheduling algorithm that minimizes the average seek time by servicing the request that is closest to the current position of the disk arm. The idea is to select the request that requires the least amount of movement of the disk arm, which is the head, to reduce the seek time.
The operating system serves the request that requires the least disk arm movement. This method reduces the average access time but may result in starvation for requests that are far from the current arm position. It serves all the requests that are close to the current head position. Thus, it reduces the seek time. For example, in the reference string given in section 2.2.2.1, this scheduling algorithm results in a total head movement of 236 cylinders. This algorithm is expected to give the least number of cylinder moves.
The algorithm services the requests in the order of their distance from the current head position, starting from the closest one. In other words, it always chooses the next request that is closest to the current head position to be serviced next.

The advantage of the SSTF algorithm is that it reduces the seek time, which is the time required for the disk arm to move to the desired cylinder. However, this algorithm may result in starvation for requests that are far from the current head position. That is, requests that are far from the current head position may have to wait for a long time to be serviced, causing delays in the processing of those requests.

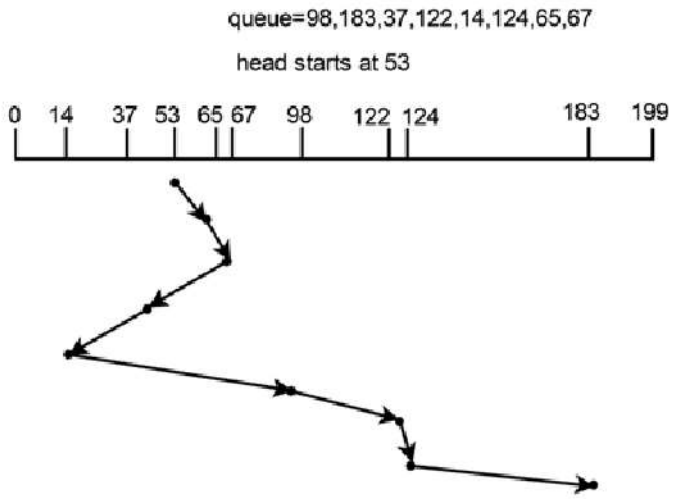

Figure 2.3: SSTF Scheduling

SSTF disk scheduling is shown in Figure 2.3. We will calculate the number of head movements.
To calculate the number of head movements in the given example, we can simply add up the distances that the disk arm needs to move to service each request. As mentioned earlier, the algorithm always chooses the next request that is closest to the current head position. So, we can calculate the distance between each consecutive request and add them up to get the total number of head movements.

- 53 to 65 = 12
- 65 to 67 = 02
- 67 to 37 = 30
- 37 to 14 = 23
- 14 to 98 = 84
- 98 to 122 = 24
- 122 to 124 = 02
- 124 to 183 = 59

In all 12 + 02 + 30 + 23 + 84 + 24 + 02 + 59 = 236 Cylinders.
In this example, the total number of head movements is 236 cylinders, which is the sum of the distances between each consecutive request. By using the SSTF algorithm, the total number of head movements is expected to be the least among all possible scheduling algorithms.
In conclusion, disk scheduling algorithms help to reduce access time and increase disk bandwidth by managing the order of disk I/O requests. The choice of algorithm depends on the specific system requirements and workload.

2.2.2.3 SCAN

The SCAN disk scheduling algorithm, also known as the elevator algorithm, is a method used by the operating system to optimize disk arm movement and efficiently handle I/O requests on a disk. When executing the SCAN algorithm, the disk arm starts moving towards one end of the disk, serving the requests it encounters along the way, until it reaches the opposite end. Once it reaches the other end, the direction of the head movement is reversed, and servicing continues in the opposite direction.

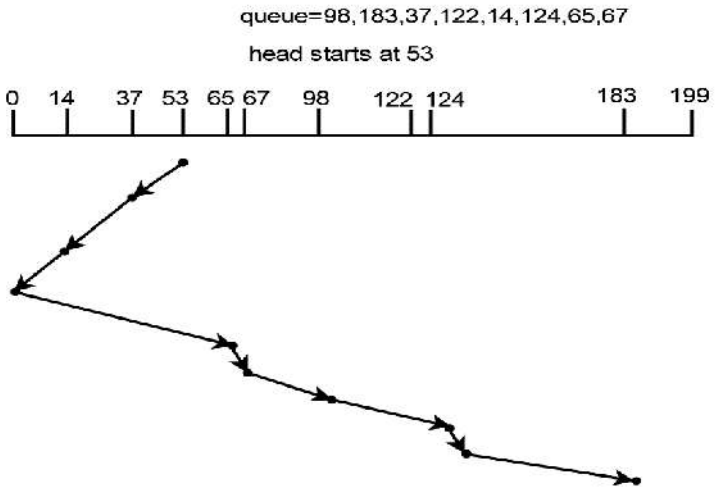

Figure 2.4: SCAN Scheduling

Let's illustrate the SCAN algorithm using an example disk queue with requests for I/O to blocks on cylinders: 98, 183, 37, 122, 14, 124, 65, and 67. Suppose the disk head is initially at cylinder 53, and the head is moving towards the last track. If the head moves towards the last track, it services the requests in the following order: 65, 67, 98, 122, 124, and 183. The head continues its movement until it reaches cylinder 199, where it reverses its direction.

After the reversal, the head moves towards the other end of the disk and services the remaining requests: 37 and then 14.

Alternatively, if the disk head is initially at cylinder 53, and it is moving towards the 0th track, the servicing order would be different:

The head first services request 37 and then 14.

When the head reaches cylinder 0, it reverses its direction.

After the reversal, the head moves towards the other end of the disk and services the remaining requests in the order: 65, 67, 98, 122, 124, and 183 as shown in figure 2.4 above.

The SCAN algorithm ensures that requests are served efficiently in a sequential manner along the path of the disk arm's movement, minimizing unnecessary head movement and reducing the seek time. This algorithm is referred to as the "elevator algorithm" because the head moves up and down the disk similar to an elevator in a building, stopping at each floor (cylinder) to service the requests. By optimizing the disk arm movement, the SCAN algorithm improves the overall disk I/O performance.

2.2.2.4 C-SCAN

The C-SCAN disk scheduling algorithm is a variation of the SCAN algorithm that aims to provide a more uniform wait time for I/O requests. Like the SCAN algorithm, C-SCAN also moves the disk head from one end of the disk to the other, servicing requests along the way. However, when the head reaches the other end of the disk, it immediately returns to the beginning of the disk without servicing any requests on the return path.

Let's elaborate on the C-SCAN algorithm using an example disk queue with I/O requests for blocks on cylinders: 98, 183, 37, 122, 14, 124, 65, and 67. Suppose the disk head is initially at cylinder 53, and it is moving towards the last track.

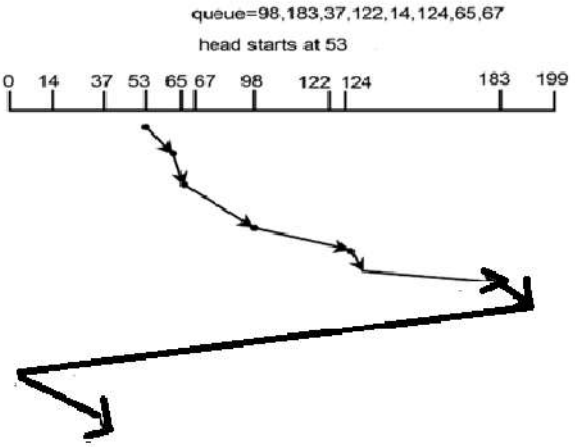

Figure 2.4: C-SCAN Scheduling

If the head moves towards the last track, it services the requests in the following order: 65, 67, 98, 122, 124, and 183. The head continues its movement until it reaches cylinder 199.

At cylinder 199, the arm immediately reverses direction and moves towards the other end of the disk, without servicing any request in this direction. Once the head reaches the beginning of the disk, it changes direction again and starts servicing the remaining requests: 14 and then 37, as shown in figure 2.4.

Please note that the C-SCAN algorithm ensures that the head always returns to the starting point after reaching the other end, which helps in providing a more consistent waiting time for I/O requests, especially for processes that are waiting for their turn in the disk queue.

Regarding the scenario when the disk head is initially at cylinder 53 and moving towards track 0, the servicing order would be different:

The head first services request 37 and then 14.

When the head reaches cylinder 0, the arm immediately reverses direction and moves towards the other end of the disk, without servicing any request in this direction.

Once the head reaches the other end of the disk, it changes direction and starts servicing the remaining requests: 65, 67, 98, 122, 124, and 183.

The C-SCAN algorithm ensures a more balanced and predictable servicing pattern, thereby reducing the variation in wait times experienced by different I/O requests, ultimately improving the overall efficiency of disk I/O operations.

2.2.2.5 LOOK

The LOOK disk scheduling algorithm, along with its variant C-LOOK, is an alternative to the SCAN and C-SCAN algorithms. LOOK and C-LOOK aim to optimize disk arm movement and reduce unnecessary head movement by only going as far as the last request in each direction, without reaching the end of the disk. Unlike SCAN and C-SCAN, which move the head to the extreme ends of the disk regardless of whether there are any requests, LOOK and C-LOOK check for requests before continuing in a given direction.

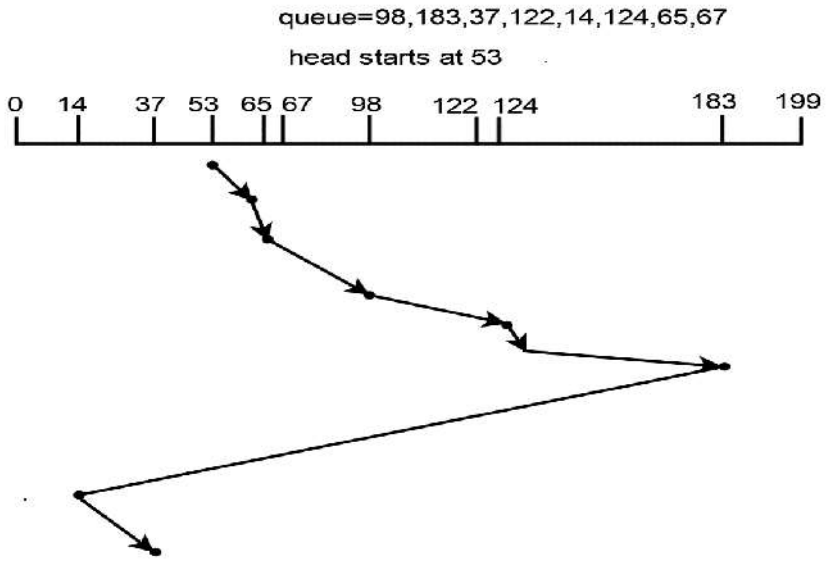

Figure 2.5: C-LOOK Scheduling

Let's elaborate on the LOOK algorithm using an example disk queue with I/O requests for blocks on cylinders: 98, 183, 37, 122, 14, 124, 65, and 67. Suppose the disk head is initially at cylinder 53, and it is moving towards the last track.

If the head moves towards the last track, it services the requests in the following order: 65, 67, 98, 122, 124, and 183. The head reaches the final request at cylinder 183, and at this point, it reverses its direction without going all the way to the end of the disk. After reversing, the head moves towards the first request in the opposite direction, which is 14. It then services request 14. Next, the head continues to move towards the last track and services the remaining request, which is 37.

Unlike SCAN, which would have moved the head to the extreme last track and then reversed, LOOK optimizes the movement by checking for requests before reversing, which can significantly reduce the seek time.

The C-LOOK algorithm, as a variant of C-SCAN, follows a similar principle. It also moves the head only as far as the last request in each direction without going to the end of the disk. When the head reaches the last request in one direction, it immediately reverses without servicing any request on the return path. The LOOK and C-LOOK algorithms are designed to minimize head movement, improve disk I/O performance, and provide faster response times for accessing data on the disk. By avoiding unnecessary travel to the extreme ends of the disk and considering requests more efficiently, LOOK and

C-LOOK can lead to a more balanced and optimized disk scheduling approach.

2.2.2.6 Selection of a Disk-Scheduling Algorithm

Disk scheduling algorithms in operating systems play a crucial role in optimizing the order in which I/O requests are serviced from the disk, aiming to improve performance and reduce seek time. Several algorithms are commonly used, each with its advantages and suitability for specific system configurations.

SSTF is a widely used disk scheduling algorithm that selects the request with the shortest seek time (i.e., the shortest distance between the current head position and the requested cylinder). SSTF can significantly reduce seek time compared to First-Come-First-Serve (FCFS), leading to improved performance. SCAN and C-SCAN are preferred when the disk experiences heavy loads with numerous read and write operations. SCAN moves the head from one end of the disk to the other, serving requests along the way, and then reverses direction without servicing any requests on the return path. C-SCAN is a variation of C-SCAN that ensures the head always returns to the starting point after reaching the other end. Both algorithms have fewer starvation problems, providing more equitable access to requests.

The choice of disk scheduling algorithm can be influenced by the file allocation method. For contiguous file allocation, where files are stored in adjacent blocks, FCFS may be the most suitable as it minimizes head movements. In contrast, linked or indexed file allocation may lead to scattered blocks, necessitating better performance from algorithms like SCAN and C-SCAN.

The location of directories and index blocks is important, as they are accessed frequently. Placing directory entries closer to the middle cylinders reduces head movement when opening files. Caching directories and index blocks in main memory can further reduce disk-arm movement, especially for read requests. Due to the complexity and importance of disk scheduling, it is typically implemented as a separate module within the operating system. This allows for easier maintenance, updates, and adaptability to different system configurations and workloads.

In conclusion, selecting an appropriate disk scheduling algorithm is crucial for optimizing disk I/O performance. SSTF is commonly used due to its reduced seek time, while SCAN and C-SCAN are preferred for systems with heavy disk loads. The choice of algorithm can be influenced by the file allocation method and the location of directories and index blocks. By treating the disk-scheduling algorithm as a separate module, the operating system can better manage disk access and improve overall efficiency.

2.3 Protection

In an operating system, it is essential to protect processes from interfering with each other's operations. To achieve this, various mechanisms can be employed to ensure that only authorized processes, with proper authorization from the operating system, can access resources such as files, memory segments, and the CPU.

Protection involves controlling the access of programs, processes, or users to the resources defined within the computer system. This mechanism should allow for specifying the necessary controls and have an

enforcement mechanism to ensure their implementation. It is essential to distinguish between protection and security, as security relates to confidence in preserving the integrity of the system and its data.

2.3.1 Goals of Protection

As computer systems have become more complex and widely used, there is a greater need to protect them and keep them safe. At first, protection was created to extend the capabilities of multiprogramming operating systems, so that even users who might not be fully trusted could share things like files and memory without causing harm. Over time, these protection ideas have improved and changed to make sure that any complicated system which uses shared resources remains dependable and reliable.

Protection serves several critical purposes. Firstly, it prevents intentional violations of access restrictions by users, ensuring system security. Additionally, it guarantees that each program component operates within the defined policies and guidelines, contributing to system reliability. A protection-oriented system can also detect errors at interfaces between subsystems early on, preventing potential issues from spreading to healthy subsystems.

Protection methods make sure that rules about using resources are followed. These rules can be set when the system is made, decided by the people who manage the system, or chosen by users to keep their files and programs safe. It's important for protection systems to be flexible because the rules can be different depending on what the computer is being used for and how it changes over time. That's why both the people who design the operating system and the people who create programs need to use protection methods. This helps them keep resources safe and follow the right rules. It's essential to distinguish between mechanisms (how something will be done) and policies (what will be done). This separation allows for adaptability, as policies may change frequently while the underlying mechanisms remain stable. General mechanisms enable the system to accommodate policy changes without requiring an overhaul of the entire structure.

In summary, protection mechanisms in computer systems play a vital role in preserving integrity, security, and reliability. They enforce access restrictions and ensure adherence to established policies, benefitting both system designers and application programmers in creating secure and adaptable software.

2.3.2 Principles of Protection

An operating system that believes in the "least privilege" rule sets up its parts, programs, system calls, and how it keeps its information to make sure things don't get too messed up if something goes wrong or if someone tries to mess with it. For instance, if there's too much stuff overflowing in a part of the system, it should just make that part stop working and not let anyone get full control of the whole system without permission.

This type of OS provides, fine-grained access controls through system calls and services, enabling privileges when needed and disabling them when not required. Additionally, it creates audit trails for all privileged function access, allowing monitoring and tracing of protection and security activities.

2.3.3 Domain of Protection

The domain of protection encompasses the entire computer system, which comprises processes and various objects. Objects can be stuff related to the computer's hardware, such as the brain of the computer (CPU), parts of its memory, things that print stuff (printers), where it keeps its information (disks), and even tapes that store data, or objects can be related to the software, like files you save, programs you use, semaphores etc. Each object has a unique name so that it can be identified.

Different objects may support distinct operations based on their nature. For example, CPUs can only do their job, memory parts can be read from and written to, CD or DVD drives can only read things, and tape drives can do things like read, write, and go back to the start. Similarly, files that store information and programs that do things can be created, opened, read from, written to, run, and even deleted.

To make sure objects stay safe, a process (which is like a task the computer does) should only be allowed to use of objects it's been given permission for and only when it really needs them. This makes sure that processes don't use objects they shouldn't and that are beyond their limits.

2.3.3.1 Domain Structure

A domain in the context of protection mechanisms refers to a set of objects and the types of access rights that processes executing within that domain possess. Each domain is represented as an ordered pair <object-name, rights-set>, where the object-name identifies a specific resource, and the rights-set denotes the set of access permissions or rights associated with that object.

For example, consider domain D with the access right <file F, {read, write}>. In this case, any process executing within domain D has the privilege to both read and write the file F. However, the same process cannot perform any other operation on file F beyond the specified read and write actions.

Domains can have shared access rights, meaning multiple domains can have access to the same resource with the same set of rights. For instance, in the given figure 2.6, we have three domains: D1, D2, and D3. The access right <O4, {print}> is shared by domains D2 and D3. This implies that any process executing in either of these two domains can print the object O4, and they are not allowed to perform any other operation on this object.

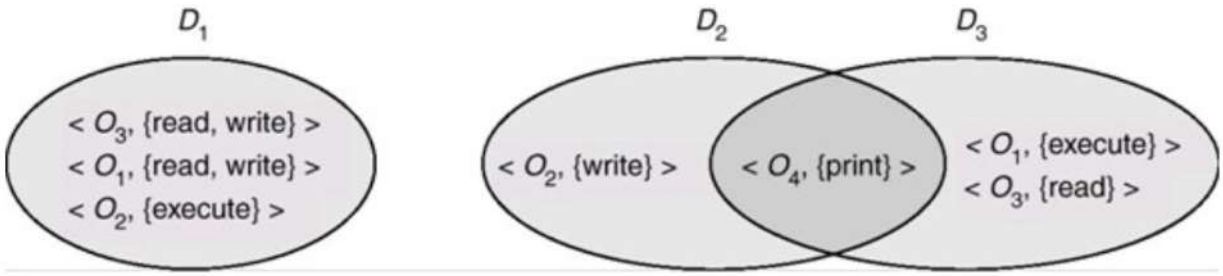

Figure 2.6: System having 3 protection domains

Domains need not be disjoint, meaning they can have overlapping access rights. This allows for a more flexible and efficient protection mechanism. The same access rights can be shared across multiple domains, promoting resource sharing and controlled access.

Realizing a domain can be achieved in various ways. It can be associated with a user, process, or procedure. Each user can be considered a domain, where their access rights are defined based on their role or permissions. Similarly, each process or procedure can be treated as a domain, with specific access rights granted to them according to their requirements.

In summary, a domain in protection mechanisms comprises objects and their associated access rights. Processes executing within a domain can access the objects based on the specified rights, and domains need not be disjoint, allowing for shared access rights. Domains can be realized as users, processes, or procedures, providing flexibility and control in managing resource access and ensuring system security.

2.4 Access Matrix

The protection model is represented by an access matrix, a general framework that facilitates protection without enforcing any specific policy. The access matrix is a matrix consisting of rows representing domains and columns representing objects. Each entry in the matrix contains a set of access rights, defining the operations a process in a particular domain can perform on an object.

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	

Figure 2.7: Access Matrix

In the given diagram, there are four domains (D1, D2, D3, and D4) and four objects (F1, F2, F3, and the printer). For example, a process executing in domain D1 can read files F1 and F3, while a process in domain D4 has the same privileges as D1 but can also write to files F1 and F3.

When a user creates a new object, the corresponding column for that object is added to the access matrix, with initialization entries based on the creator's specifications. Processes can switch from one domain to another by executing the "switch" operation on the domain object. Such domain switching is allowed only if the access right for "switch" is present in the corresponding access matrix entry.

To allow controlled changes in the access-matrix entries, three additional operations are needed: "copy," "owner," and "control." The "copy" operation permits copying an access right from one domain (or row) to another, but only within the same column where the right is defined. For instance, a process in domain D2 can copy the "read" operation into any entry associated with file F2.

The access matrix is a flexible model for protection, providing a comprehensive view of domain-object access rights. It allows processes to execute operations on objects based on their domain's permissions. Adding new objects involves initializing the corresponding matrix column. Domain switching is restricted by access rights. To enable controlled changes, the "copy" operation permits copying rights

within the same column. The access matrix is a powerful tool for enforcing protection in various scenarios, adapting to different protection policies, and ensuring secure resource management.

Implementation of the Access Matrix can be achieved using different methods, each with its advantages and drawbacks:

1. Global Table:
   The simplest implementation involves maintaining a global table containing ordered triples <domain, object, rights-set>. When an operation is executed on an object within a domain, the table is searched for a matching triple. If found, the operation is allowed; otherwise, an exception is raised. However, this method has drawbacks like its large size, making it difficult to keep in the main memory, and additional I/O requirements.

2. Access Lists for Objects:
   In this method, each column in the access matrix is represented as an access list for one object. The list consists of ordered pairs <domain, rights-set>, defining the access rights for that object. When an operation is executed on an object in a domain, the access list for that object is checked. If an entry with the appropriate domain and rights is found, the operation is allowed; otherwise, a default set is checked. If the operation is present in the default set, access is granted; otherwise, access is denied.

3. Capability Lists for Domains:
   A capability list for a domain is a list of objects with the operations allowed on those objects. Each object is represented by its name or address, known as a capability. To execute an operation on an object, the process specifies the corresponding capability as a parameter. Simple possession of the capability allows access. Capabilities can be distinguished from other data by using tags denoting their type or by having a separate address space accessible only by the operating system.

4. Lock-Key Mechanism:
   The lock-key scheme is a compromise between access lists and capability lists. Each object has a list of unique bit patterns called locks, and each domain has a list of unique bit patterns called keys. Access to an object is allowed only if the domain possesses a key matching one of the locks of the object.

The access matrix can be implemented using different approaches, each with its benefits and limitations. The choice of implementation depends on the specific requirements and characteristics of the operating system.

2.5 Access Control

Access control in an operating system involves assigning access-control information to files and directories based on their owners, groups, or a list of users. Solaris 10, as part of Sun Microsystems

operating system, implements role-based access control (RBAC), which incorporates the principle of least privilege. This approach uses privileges to grant specific rights to processes or roles, limiting their access to only what is necessary for their tasks. Privileges and programs can be assigned to roles, allowing users to assume roles based on passwords to gain access to certain privileges. This mitigates security risks associated with superusers and setuid programs, enhancing system security.

Revocation of access rights in a system with distributed capabilities can be achieved through various schemes:

1. Reacquisition: Capabilities are regularly removed from every domain. If a process tries to utilize a capability that has been revoked, it will be unable to regain access to it.
2. Back-pointers: Every object keeps a record of pointers to all linked capabilities. When revocation occurs, these pointers are traced to modify or remove the necessary capabilities.
3. Indirection: Capabilities indirectly reference objects through a global table entry. When revocation is carried out, the global table is searched, and the relevant entry is removed, causing the capability to become invalid.
4. Keys: A key is an exclusive bit pattern linked to each capability, established during the creation of the capability. Objects possess a primary key that can be substituted. When a capability is utilized, its key is compared to the master key. If they correspond, the operation is permitted to proceed; otherwise, an exception is raised.

In key-based schemes, operations to define, insert, or delete keys from lists should not be accessible to all users, as it could lead to unauthorized access.
These mechanisms for access control and revocation ensure that users, processes, and roles have appropriate permissions and privileges, minimizing the risk of unauthorized access and maintaining a secure computing environment.

2.6 Security

2.6.1 Introduction

Up to this point, our focus has been on protection, addressing the challenge of controlling access to programs and data within a computer system—an internal issue. However, the realm of security involves safeguarding these resources from external threats. When programs and data in a system are exposed to the external environment, they become susceptible to misuse, potentially leading to system inconsistencies. In this section, we will delve into the various external threats that can jeopardize a system's security and explore methods for providing that security.

2.6.2 The Security Problem

Ensuring the security of a computer system is of paramount importance, especially when dealing with confidential and financial data that may attract malicious intent. True security exists when resources are consistently used according to their permissions. Unfortunately, achieving absolute security is challenging, so we must implement mechanisms to prevent security breaches.

Security violations or misuse of the system can be intentional or accidental. While accidental misuse can be relatively easy to protect against, intentional breaches are more complex. In our discussion, we will use the following terms:

- Intruder or cracker: The individual attempting to breach security.
- Threat: The potential for a security violation.
- Attack: When security is compromised.

Several forms of security breaches include:

- Breach of confidentiality: Unauthorized reading or theft of information, leading to the exposure of secret data.
- Breach of integrity: Unauthorized modification of data.
- Breach of availability: Unauthorized destruction of data.
- Theft of service: Unauthorized use of resources.
- Denial of service: Preventing legitimate system use.

To secure a system comprehensively, security mechanisms must be applied at different levels, including physical security, human practices, the operating system, and network-level security.

2.6.3 Program Threats

Attackers often write malicious programs to compromise a system's security, causing normally executing processes to behave abnormally. Let's explore some common methods by which programs can cause security breaches:

- Trojan Horse: Mechanisms allow programs written by one user to be executed by another. If the writer's access rights are available to another user, misuse may occur. Trojan horses typically enter systems through emails and can execute attacks as designed by the attacker.

- Trap Door: Programmers may intentionally create hidden entry points, known as trap doors, to enable unauthorized access without proper credentials.

- Logic Bomb: This is a code that lies dormant until specific conditions are met, triggering malicious actions. Detecting logic bombs can be challenging, as they remain inactive under normal circumstances.

- Viruses: Viruses are pieces of code embedded in other programs. They self-replicate and infect systems, potentially modifying or destroying files and causing system crashes. Viruses spread through methods such as email attachments, downloaded files, or infected disks.

2.6.4 System and Network Threats

Threats can also emerge from system and network connections, targeting both the operating system and

user programs and data. Open-source operating systems are especially vulnerable due to numerous potential entry points. Let's examine common examples of threats:

- Worms: Worms are self-replicating programs that consume system resources, often leading to the shutdown of entire networks when they spread from one computer to another.

- Port Scanning: Attackers attempt to identify open network ports, which serve as entry points for information exchange. When an open port is found, attackers may attempt to insert trojan horses, viruses, worms, and more.

- Denial of Service (DoS): In this type of attack, attackers disrupt the availability of system resources and services without stealing information. These attacks are purely network-based and preventing them is considered extremely challenging.

Now that we understand various types of attacks, it's important to know that several well-established mechanisms and algorithms are available to defend against these threats. A widely used method among system developers is cryptography, which ensures the security of computer systems and data.
This section, provides a structured overview of computer system security, covering the importance of security, common threats, and the mechanisms used to defend against them. Use this as a foundation for deeper exploration and study in the field of computer security.

2.7 Self-Assessment Questions

Q1. What is a cylinder in a disk drive and how does it relate to the number of tracks on each platter? [2 marks, L2]
Q2. What is a magnetic disk and how does it store data? [3 marks, L2]
Q3. What is seek time and how does it relate to accessing data on a disk drive? [2 marks, L2]
Q4. What is the difference between Constant Linear Velocity (CLV) and Constant Angular Velocity (CAV) disk structure? [3 marks, L2]
Q5. What is the difference between protection and security in an operating system? [3 marks, L2]
Q6. Describe how disk management is done. [8 marks, L2]
Q7. List the goals of protection. [4 marks, L1]
Q8. What is the primary purpose of the access matrix in a protection model? [2 marks, L1]
Q9. Explain the concept of the "owner right" in the context of the access matrix. How does it enable controlled changes in access rights? [4 marks, L2]
Q10. Provide an example of how the "switch" operation restricts domain switching based on access rights. Use a scenario to illustrate your answer.[6 marks, L3]
Q11. Evaluate the importance of having distinct mechanisms and policies in protection systems. How does this separation contribute to system adaptability and stability? .[8 marks, L4]
Q12. Compare and contrast the advantages and drawbacks of implementing the access matrix using the "Global Table" method and the "Access Lists for Objects" method. [8 marks, L4]

2.8 Self-Assessment Activities

A1. Suppose you have a system that stores sensitive information. How can the operating system ensure that this information is protected from unauthorized access, modification, or deletion?

A2. Disk scheduling is performed by the operating system to schedule I/O requests for the disk. The purpose is to improve the performance of the service. For the below-given reference string of cylinders find out the number of the head moves with different algorithms and find out which algorithm performs better than the others. Assume that there are a total of 200 cylinders and the head is current at cylinder 50.

80,170,45,140,24,16,190

A3. Imagine you want to copy a file from one location to another on the hard disk. How does the operating system locate the file's physical location on the disk?

A4. Suppose you have a system that uses the First-Come, First-Serve (FCFS) disk scheduling algorithm. How can you optimize the disk access time and avoid the disk from being idle for long periods?

A5. Analyze the benefits and drawbacks of implementing the access matrix using the "Global Table" method and the "Capability Lists for Domains" method for the data storage system. Consider the scalability, management, and resource utilization aspects.

2.9 Multiple-Choice Questions

Q1. What are magnetic disks? [1 mark, L1]

A. A type of primary storage
B. A type of secondary storage
C. A type of RAM
D. A type of CPU

Q2. What is a cylinder in a magnetic disk? [1 mark, L1]

A. A collection of tracks at one arm location
B. A circular track on the disk platter
C. The basic unit of storage on the disk
D. The number of platters in a disk drive

Q3. What is seek time? [1 mark, L1]

A. The time taken for the disk to rotate until the required sector is beneath the read/write head
B. The duration required for moving the disk arm to the desired track
C. The time that elapses between the initial service request and the completion of the last transfer
D. The speed at which data flows between the drive and the computer

Q4. What is disk bandwidth? [1 mark, L1]

A. The sum of seek time and rotational latency
B. The time taken for the disk to rotate until the required sector is beneath the read/write head
C. The duration required for moving the disk arm to the desired track
D. The total bytes transferred divided by the time that elapses between the initial service request and the completion of the last transfer

Q5. What is a head crash? [1 mark, L1]

A. When the disk head is sustained by a paper-thin air cushion and comes into contact with the disk surface, causing damage to the magnetic surface
B. When the disk platter is scratched
C. When the disk drive is unable to read or write data
D. When the disk bandwidth is too low

Q6. What is the main goal of protection in a multiprogramming system? [1 mark, L1]

A. To ensure that only authorized processes can access resources
B. To improve the reliability of the system by detecting errors
C. To limit the amount of damage that can be caused by a faulty process
D. All of the above

Q7. What is the difference between protection and security in an operating system? [1 mark, L1]

A. Protection deals with controlling access to programs and data within the system, while security focuses on protecting the system from external threats that may damage it.
B. Protection focuses on protecting the system from external threats that may damage it, while security deals with controlling access to programs and data within the system.
C. Protection and security are interchangeable terms that refer to the same concept.
D. None of the above

Q8. In the access matrix, what do the rows represent? [1 mark, L1]

A. Access rights for processes.
B. Objects within the system.
C. Domains or processes.
D. Access control mechanisms

Q9. What is the role of the "copy" operation in the access matrix? [1 mark, L1]

A. To allow copying rights between domains within the same column.
B. To enable changing entries within a row.
C. To grant the control right to a domain.
D. To facilitate changing domain ownership

Q10. Which implementation of the access matrix involves maintaining a global table containing ordered triples? [1 mark, L1]

A. Capability Lists for Domains
B. Global Table
C. Lock-Key Mechanism
D. Access Lists for Objects

Q11. In key-based revocation schemes, what is the purpose of the master key associated with objects? [1 mark, L1]

A. To grant access to all users.
B. To create new capabilities.
C. To facilitate role-based access control.
D. To control access to objects.

Q12. What is the term used to define a set of access permissions associated with an object in a domain? [1 mark, L1]

A. Object name
B. Access rights

C. Policies
D. Domain structure

2.10 Keys to Multiple-Choice Questions

Q1. A type of secondary storage (B)
Q2. A collection of tracks at one arm location (A)
Q3. The duration required for moving the disk arm to the desired track (B)
Q4. The total bytes transferred divided by the time that elapses between the initial service request and the completion of the last transfer (D)
Q5. When the disk head is sustained by a paper-thin air cushion and comes into contact with the disk surface, causing damage to the magnetic surface. (A)
Q6. All of the above (D)
Q7. Protection deals with controlling access to programs and data within the system, while security focuses on protecting the system from external threats that may damage it. (A)
Q8. Domains or processes. (C)
Q9. To allow copying rights between domains within the same column. (A)
Q10. Global Table (B)
Q11. To control access to objects. (D)
Q12. Access rights. (B)

2.11 Summary of the Unit

In computer systems, protection methods are really important to keep things safe from being misused. These things can be stuff like parts of the computer (memory, CPU time, and things like printers) or software stuff like files you save, programs you run, and special tools called semaphores. Access rights mean permissions that let you do certain actions with these things.

A domain is like a group of these access rights. When a computer task (process) runs, it uses the rights from its domain to work with these things. This task can stay in one group or sometimes switch to another. There's something called an access matrix which helps with protection. It's like a big chart that shows who can do what, without making everyone follow the same rules. It's smart because it can change and fit different situations.This chart can be set up in two main ways: with lists that say what each thing can do, or lists that say what each group can do. If we want to change or take away permissions, it's usually easier using the first way.

So, in simple words, protection is important to keep computer stuff safe from being used in the wrong way. And there are clever tools like the access matrix that help make sure everyone uses things the right way. The access matrix is a powerful tool to manage protection in computer systems, enabling efficient access control to various objects while allowing for flexibility in designing protection policies and mechanisms. The choice between access lists and capability lists depends on the system's specific requirements and the ease of implementing dynamic protection and access rights revocation. This unit introduces the fundamental concepts of computer system security. It distinguishes between protection

and security, highlighting the need to defend against external threats. We explore various forms of security breaches, including program-based and network-related threats, and the essential role of security mechanisms, such as cryptography, in maintaining system integrity and data security.

2.12 Keywords

Protection mechanisms, Access rights, Domains, Dynamic protection, Access matrix, Sparse matrix, Access lists, Capability lists, Revocation of access rights

2.13 Recommended Learning Resources

[1] Silberschatz, A., Galvin, P., & Gagne, G. (2005). Operating System Concepts, 7th ed., Hoboken.

[2] William stalling. Operating Systems: Internal and design principles, 7th edition PHI

[3] D.M. Dhamdhere. Operating Systems: A concept-based Approach, 2nd Edition, TMH