MODULE 5: STORAGE MANAGEMENT-FILE SYSTEM:

Contents:

10.1 FILE SYSTEM: FILE CONCEPT

- A file is a collection of similar records. x The data can't be written on to the secondary storage unless They are within a file.
- Files represent both the program and the data. Data can be numeric, alphanumeric, alphabetic or binary.
- Many different types of information can be stored on a file ---Source program, object programs, executable programs, numeric data, payroll recorder, graphic images, sound recordings and so on.
- A file has a certain defined structures according to its type:

  Text file:-Text file is a sequence of characters organized in to lines.

  Object file:-Object file is a sequence of bytes organized in to blocks understandable by the systems linker.

  Executable file:-Executable file is a series of code section that the loader can bring in to memory and execute.

  Source File:-Source file is a sequence of subroutine and function, each of which are f further organized as declaration followed by executable statements.

10.1.1 FILE ATTRIBUTES

File attributes varies from one OS to other. The common file attributes are:

1 **Name:-**The symbolic file name is the only information kept in human readable form.

2 **Identifier:-**The unique tag, usually a number, identifies the file within the file system. It is the nonreadable name for a file.

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

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

5 **Size:-**The current size of the file and possibly the maximum allowed size are included in this attribute.

6 **Protection:-**Access control information determines who can do reading, writing, execute and so on.
7 **Time, data and User Identification:-**This information must be kept for creation, last modification and last use. These data are useful for protection, security and usage monitoring.

10.1.2 FILE OPERATION:-

File is an abstract data type. To define a file we need to consider the operation that can be performed on the file. Basic operations of files are:

1. **Creating a file:-**Two steps are necessary to create a file. First space in the file system for file is found. Second an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.
2. **Writing a file**:-System call is mainly used for writing in to the file. System call specify the name of the file and the information i.e., to be written on to the file. Given the name the system search the entire directory for the file. The system must keep a write pointer to the location in the file where the next write to be taken place.
3. **Reading a file:-**To read a file system call is used. It requires the name of the file and the memory address. Again the directory is searched for the associated directory and system must maintain a read pointer to the location in the file where next read is to take place.
4. **Delete a file:-**System will search for the directory for which file to be deleted. If entry is found it releases all free space. That free space can be reused by another file.
5. **Truncating the file:-**User may want to erase the contents of the file but keep its attributes. Rather than forcing the user to delete a file and then recreate it, truncation allows all attributes to remain unchanged except for file length.
6. **Repositioning within a file:-**The directory is searched for appropriate entry and the current file position is set to a given value. Repositioning within a file does not need to involve actual i/o. The file operation is also known as file seeks.

In addition to this basis 6 operations the other two operations include appending new information to the end of the file and renaming the existing file. These primitives can be combined to perform other two operations. Most of the file operation involves searching the entire directory for the entry associated with the file. To avoid this OS keeps a small table containing information about an open file (the open table). When a file operation is requested, the file is specified via index in to this table. So searching is not required. Several piece of information are associated with an open file:

File pointer:-on systems that does not include offset an a part of the read and write system calls, the system must track the last read-write location as current file position pointer. This pointer is unique to each process operating on a file.

File open count:-As the files are closed, the OS must reuse its open file table entries, or it could run out of space in the table. Because multiple processes may open a file, the system must wait for the last file to close before removing the open file table entry.

The counter tracks the number of copies of open and closes and reaches zero to last close.

Disk location of the file:-The information needed to locate the file on the disk is kept in memory to avoid having to read it from the disk for each operation. x

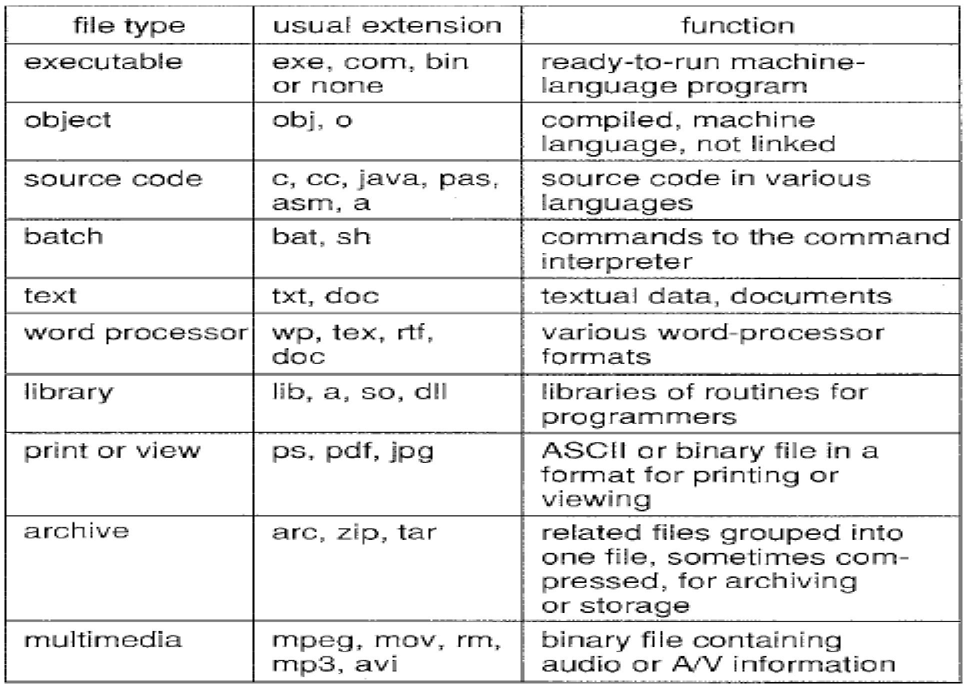
Access rights:-Each process opens a file in an access mode. This information is stored on per-process table the OS can allow OS deny subsequent i/o request.

10.1.3: FILE TYPES

A common technique for implementing file types is to include the type as part of the file name. The name is split into two parts-a name and an extension, usually separated by a period character (Figure 10.2). In this way, the user and the operating system can tell from the name alone what the type of a file is.

For example, most operating systems allow users to specify a file name as a sequence of characters followed by a period and terminated by an extension of additional characters. File name examples include resume.doc, Server.java, andReaderThread.c**.**

The system uses the extension to indicate the type of the file and the type of operations that can be done on that file. Only a file with a .com, .exe, or .bat extension can be executed, for instance. The .com and .exe files are two forms of binary executable files, whereas a .bat file is a batch file containing, in ASCII format, commands to the operating system. MS-DOS recognizes only a few extensions, but application programs also use extensions to indicate file types in which they are interested. For example, assemblers expect source files to have an .asm extension, and the Microsoft Word word processor expects its files to end with a .doc extension.



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Fig 1: Common File types

10.2 ACCESS METHODS The information in the file can be accessed in several ways. Different file access methods are:

1. **Sequential Access:**

Sequential access is the simplest access method. Information in the file is processed
in order, one record after another. Editors and compilers access the files in this fashion. Normally read and write operations are done on the files. A read operation reads the next portion of the file and automatically advances a file pointer, which track next i/I track. Write operation appends to the end of the file and such a file can be next to the beginning.

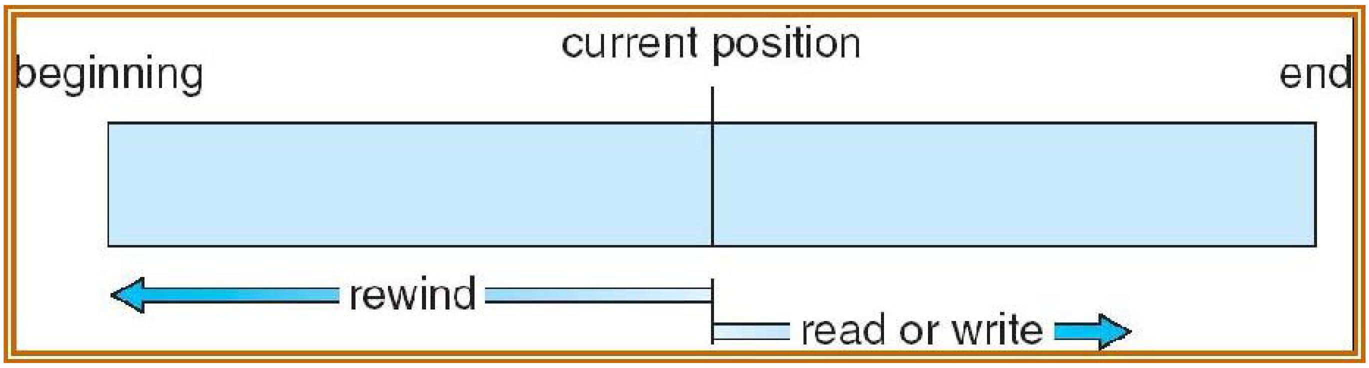


Fig 2 :Sequential access depends on a tape model of a file.

2. **Direct Access:**

Direct access allows random access to any file block. This method is based on disk model of a file.A file is made up of fixed length logical records. It allows the program to read and write records rapidly in any order.A direct access file allows arbitrary blocks to be read or written. *Eg*:-User may need block 13, then read block 99 then write block 12. For searching the records in large amount of information with immediate result, the direct access method is suitable. Not all OS support sequential and direct access. Few OS use sequential access and some OS uses direct access. It is easy to simulate sequential access on a direct access but the reverse is extremely inefficient.

10.3 PROTECTION:

When information is stored in a computer system, we want to keep it safe from physical damage (the issue of reliability) and improper access (the issue of protection).
Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically (or through computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed.
File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism. Files may be deleted accidentally.

10.3.1 Types of Access:

The need to protect files is a direct result of the ability to access files. Systems that do not permit access to the files of other users do not need protection. Thus, we could provide complete protection by prohibiting access. Alternatively, we could provide free access with no protection. Both approaches are too extreme for general use. What is needed is controlled access.

Access is permitted or denied depending on several factors, one of which is the type of access requested. Several different types of operations may be controlled:

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

10.3.2 ACCESS CONTROL:

The most common approach to the protection problem is to make access dependent on the identity of the user. Different users may need different types of access to a file or directory. The most general scheme to implement Identity-dependent access is to associate with each file and directory an **Access Control List (ACL)** specifying user names and the types of access allowed for each user.

When a user requests access to a particular file, the operating system checks the access list associated with that file. If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs, and the user job is denied access to the file. This technique has two undesirable consequences:

- Constructing such a list may be a tedious and unrewarding task, especially if we do not know in advance the list of users in the system.
- The directory entry, previously of fixed size, now must be of variable size, resulting in more complicated space management.

To condense the length of the access-control list, many systems recognize three classifications of users in connection with each file:

Owner: The user who created the file is the owner.

Group: A set of users who are sharing the file and need similar access is a group, or work group.

Universe: All other users in the system constitute the universe.

To illustrate, consider a person, Sara, who is writing a new book. She has hired three graduate students (Jim, Dawn, and Jill) to help with the project. The text of the book is kept in a file named *book*. The protection associated with this file is as follows:

- Sara should be able to invoke all operations on the file.
- Jim, Dawn, and Jill should be able only to read and write the file; they should not be allowed to delete the file.
- All other users should be able to read, but not write, the file. (Sara is interested in letting as many people as possible read the text so that she can obtain feedback.)

With the more limited protection classification, only three fields are needed to define protection. Often, each field is a collection of bits, and each bit either allows or prevents the access associated with it. For example, the UNIX system defines three fields of 3 bits each -rwx, where r controls read access, w controls write access, and x controls execution.
A separate field is kept for the file owner, for the file's group, and for all other users. In this scheme, 9 bits per file are needed to record protection information. Thus, for our example, the protection fields for the file *book* are as follows: for the owner Sara, all bits are set; for the group *text,* the rand w bits are set; and for the universe, only the r bit is set. One difficulty in combining approaches comes in the user interface. Users must be able to tell when the optional ACL permissions are set on a file. In the Solaris example, a"+" appends the regular permissions, as in:

```
19 -rw-r--r--+ 1 jim staff 130 May 25 22:13 file1
```

CHAPTER 12: SECONDARY STORAGE STRUCTURE

12.1: Overview of MASS STORAGE STRUCTURES

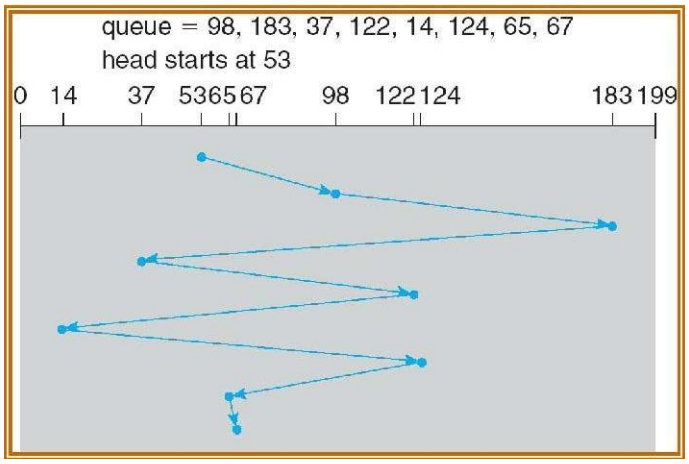
Disks provide a bilk of secondary storage. Disks come in various sizes, speed and information can be stored optically or magnetically.

- Magnetic tapes were used early as secondary storage but the access time is less than disk.
- Modern disks are organized as single one-dimensional array of logical blocks.
- The actual details of disk i/o open depends on the computer system, OS, nature of i/o channels and disk controller hardware.
- The basic unit of information storage is a sector. The sectors are stored on flat, circular, media disk. This disk media spins against one or more read-write heads. The head can move from the inner portion of the disk to the outer portion.
- When the disk drive is operating the disks is rotating at a constant speed.
- To read or write the head must be positioned at the desired track and at the beginning if the desired sector on that track.
- Track selection involves moving the head in a movable head system or electronically selecting one head on a fixed head system.
- These characteristics are common to floppy disks, hard disks, CD-ROM and DVD.

12.2 DISK SCHEDULING

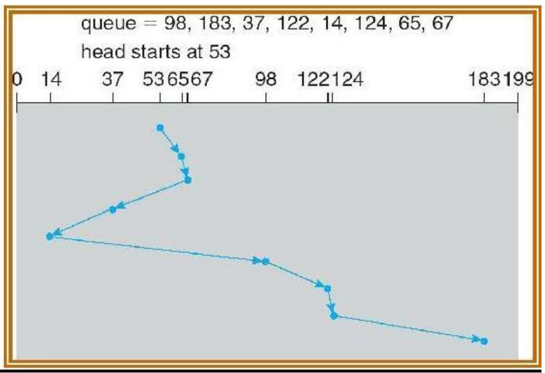
Different types of scheduling algorithms are as follows:

1. **FCFS scheduling algorithm**:This is the simplest form of disk scheduling algorithm. This services the request in the order they are received. This algorithm is fair but do not provide fastest service. It takes no special time to minimize the overall seek time. *Eg:-* consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65, 67



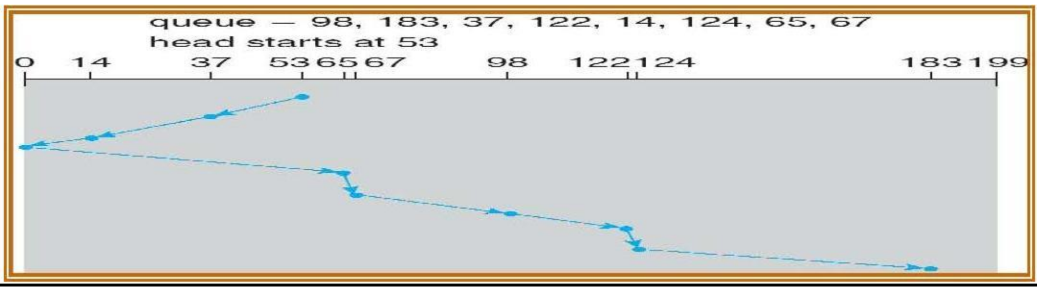
If the disk head is initially at 53, it will first move from 53 to 98 then to 183 and then to 37, 122, 14, 124, 65, 67 for a total head movement of 640 cylinders. The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule. If the requests for cylinders 37 and 14 could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

2. **SSTF** (Shortest seek time first) **algorithm**:This selects the request with minimum seek time from the current head position. Since seek time increases with the number of cylinders traversed by head, SSTF chooses the pending request closest to the current head position. *Eg*:-:-consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65, 67



If the disk head is initially at 53, the closest is at cylinder 65, then 67, then 37 is closer then 98 to 67. So it services 37, continuing we service 14, 98, 122, 124 and finally 183. The total head movement is only 236 cylinders. SSTF is essentially a form of SJF and it may cause starvation of some requests. SSTF is a substantial improvement over FCFS, it is not optimal.

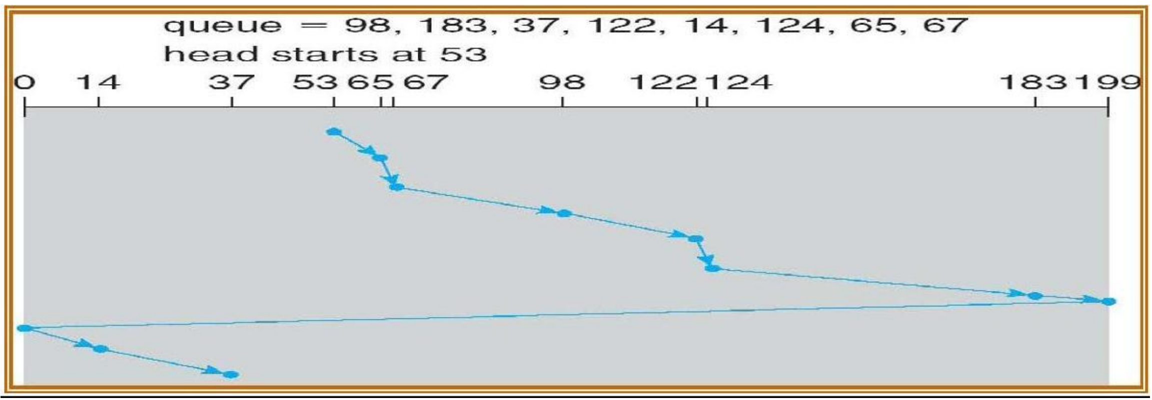
3. **SCAN algorithm**:In this the disk arm starts at one end of the disk and moves towards the other end, servicing the request as it reaches each cylinder until it gets to the other end of the disk. At the other end, the direction of the head movement is reversed and servicing continues. *Eg*:-:-consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65, 67



If the disk head is initially at 53 and if the head is moving towards 0, it services 37 and then 14. At cylinder 0 the arm will reverse and will move towards the other end of the disk servicing 65, 67, 98, 122, 124 and 183. If a request arrives just in from of head, it will be serviced immediately and the request just behind the head will have to wait until the arms reach other end and reverses direction. The SCAN is also called as elevator algorithm.

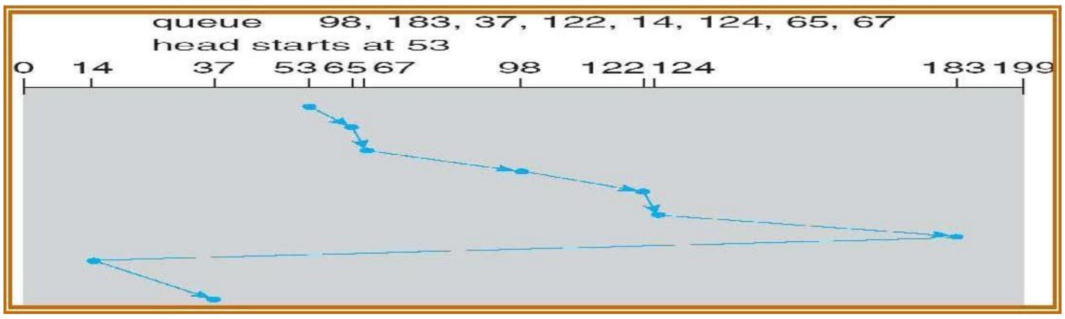
4. **C-SCAN** (Circular scan) **algorithm**:

C-SCAN is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C- SCAN moves the head from end of the disk to the other servicing the request along the way. When the head reaches the other end, it immediately returns to the beginning of the disk, without servicing any request on the return. The C-SCAN treats the cylinders as circular list that wraps around from the final cylinder to the first one. *Eg*:-



5.Look Scheduling algorithm:

Both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice neither of the algorithms is implemented in this way. The arm goes only as far as the final request in each direction. Then it reverses, without going all the way to the end of the disk. These versions of SCAN and CSCAN are called Look and C-Look scheduling because they look for a request before continuing to move in a given direction. *Eg*:



Selection of Disk Scheduling Algorithm:

1. SSTF is common and it increases performance over FCFS.
2. SCAN and C-SCAN algorithm is better for a heavy load on disk.
3. SCAN and C-SCAN have less starvation problem.
4. SSTF or Look is a reasonable choice for a default algorithm.

CHAPTER 17: SYSTEM PROTECTION

17.1 GOALS OF PROTECTION

- Discuss the goals and principles of protection in a modern computer system
- Explain how protection domains combined with an access matrix are used to specify the resources a process may access. Examine capability and language-based protection systems
- 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

17.2 PRINCIPLES OF PROTECTION

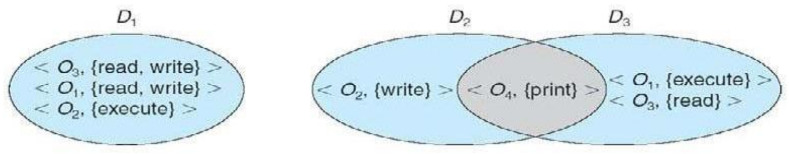
• Guiding principle principle of least privilege
– Programs, users and systems should be given just enough privileges to perform their tasks **Disk Attachment :** Stable-Storage Implementation

- Write-ahead log scheme requires stable storage
- To implement stable storage:
- Replicate information on more than one nonvolatile storage media with independent failure modes
- Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery

17.3 DOMAIN OF PROTECTION

Domain Structure

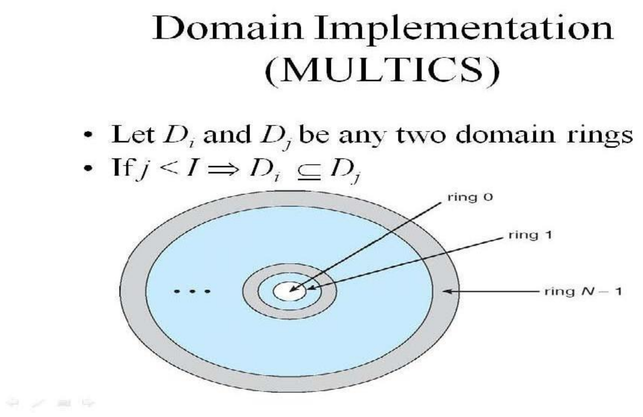
Access-right=<object-name,rights-set> where *rights-set* is a subset of all valid operations that can be performed on the object.



Domain 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 •

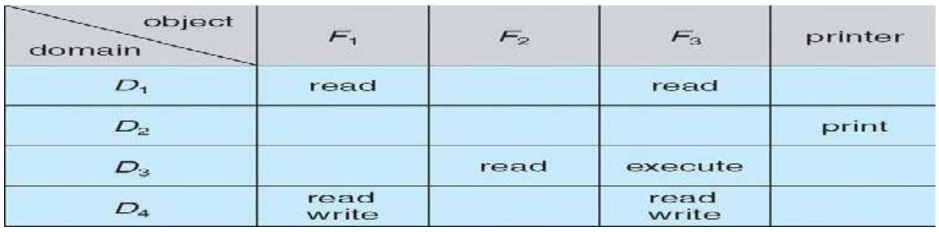


Domain Implementation
(MULTICS)

- Let D_i and D_j be any two domain rings
- If $j < I \Rightarrow D_i \subseteq D_j$

17.4 ACCESS MATRIX

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 Domaini can invoke on Objectj



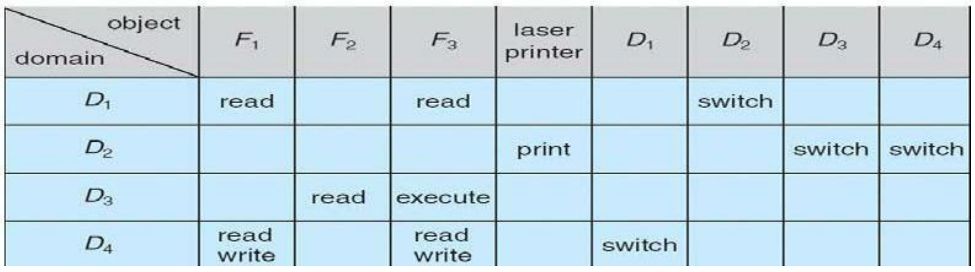
object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Use of Access Matrix

If a process in Domain Di tries to do "op" on object Oj, then "op" must be in the access matrix
Can be expanded to dynamic protection
– Operations to add, delete access rights
– Special access rights:

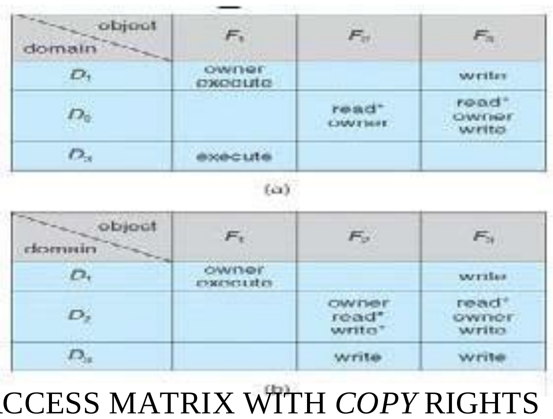
- **owner** of Oi
- **copy** op from Oi to Oj
- **control** – Di can modify Dj access rights
- **transfe**r – switch from domain Di to Dj

- 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



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			

ACCESS MATRIX OF FIGURE A WITH DOMAINS AS OBJECTS



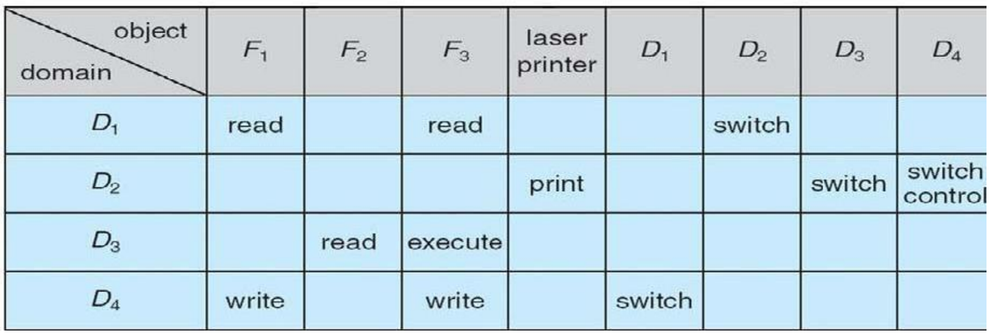
object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

ACCESS MATRIX WITH *COPY* RIGHTS



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 control
D_3		read	execute					
D_4	write		write		switch			

MODIFIED ACCESS MATRIX OF FIGURE B