Unit-IV

Information Management : Files (Low level files, structured files, database management systems, multimedia storage), Low level file implementation. Storage Abstraction (Structure sequential files, indexed sequential files, database Management Systems, Multimedia documents). Memory mapped files, Directories, diectory implementation, file sharing information across network remote Viruses and Worms, Security Design principles, Authentications, Protection mechanisms, encryption, protection of User FIles.

Chapter

8 Information Management

Introduction

To access information at any time, it is necessary to store information for long-term, it is also essential to make data sharable for various processes. And large data required storage devices. For all these essential requirements, it is necessary to store data on disks and other external media in units called **files**.

Files are managed by the O/S. Processes running in the system can read or write these storage media via files. How to structure, named, accessed protected and implemented are few considerations in operating system design.

8.1 File concept

A file is the smallest allotment of secondary storage device (disks) or a file is sequence of logical records (sequence of bits & bytes). A fill system normally organized into directories to ease their use. These directories may contain files and other directions. The five major activities of an Operating system in regard to file management are :

1. The creation and deletion of files.
2. The creation and deletion of directions
3. The support of primitives for manipulating files & directions.
4. The mapping of files onto secondary storage.
5. The back up of files on stable storage media.

Files are fundamental abstraction of secondary storage devices (such as magnetic tapes or disk drives), although the system software way also provide other higher level abstractions such as virtual memory.

A file is a container for a collection of information. The file manager provider a protection mechanism to allow machine users to administer how processes executing.

Storage devices are capable of storing only linearly addressed blocks of bytes. The file system provides abstraction from storage blocks to data structures suitable for use by application programs. The file system provides an abstraction that links blocks of the storage system.

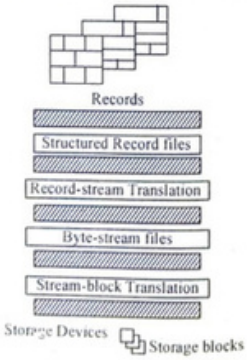


Figure 8.1 : Information structure

together to form a logical collection of information, it is called stream block translation in figure. Conceptually, this translation allows one to store and retrieve an arbitrary stream of linearly addressed bytes on the block oriented storage system.

When an application's data structure is written to a storage device, it will have to be converted into a byte stream by a record stream translation procedure. Then stream can be stored as a set of blocks. Later, when we retrieved data, it will be read block by block, converted into a stream of bytes, and then converted back into the application level data structure.

8.1.1 Low level files

If an O/S provides only the stream-block translation facilities, it is said to provide a low-level file system. Window NT & UNIX provide low level file.

If the file system provides the record stream translation, it is a structured (or high level) file system. IBM MVS provides high level file system.

A byte stream file system implements the block stream translation. A byte stream file is a named sequence of bytes indexed by the non negative integers. Each byte in the stream of bytes has an index frame O onwards. The file position is used to reference bytes in the file. At any given moment after the file has been opened, the file pointer shift the index of the next byte in the file to be read from or written to.

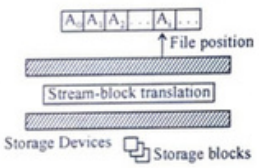


Figure 8.1.1 : Block stream translation

The file system keeps a data structure called a file descriptor in which it stores detailed information about each file. Different file managers store different information. The most are maintain the following :

Current state : This can be archived, for reading writing, executing, appending and soon.

Locks : A process that allow a file system to lock file for reading only and for writing only.

Storage device details : That contains the details of how the blocks in the file can be accessed.

Length : The number of bytes contained in the file.

Sharable : A field in which identify the file as read sharable, write sharable, execution sharable etc.

Users : The list of processes currently having the file open.

Count : The number of directories in which the file appears, if the directory system allows a file to appear in more than one directory.

External name : A character string name for the file, used at the command line or from an application program. This is symbolic name.

Structured files : A generic structured file can be used to hold any kind of information, including absolute program images, relocatable object programs, libraries numeric data organized

according to the needs of some set of applications textual data such as source programs, word processing documents and information ready to print.

Structured files are represented on the storage medium as a collection of blocks. Structure file descriptors must contain the same kind of information that are kept for a low-level file, as well as additional fields to support data structures as follows :

- **Type :** A tag that is describing the type of the file.
- **Access method :** A composite of functions to be used to read from, write to append to, update, or otherwise access the file.
- **Other :** Specific structured file types may have other fields to represent the relationship of this file to related files, the minimum version number of the file manager handling this file type and so on.

8.1.2 Structured sequential file

It is a sequence of logical records indexed by the non negative integers. As with byte stream files, access to the file is defined by a file position, but in this case the position indexes as records in the file instead of bytes like open (file name), close (file ID), get Record (file ID, record), seek (file ID, position). These operations are equivalent to the operations for the byte stream file, excepts the data are stored in records instead of bytes.

8.1.2.1 Indexed sequential files : Sequential access information is not useful in some applications when we need interactive query system, such as ATM, any particular work done by the program will be concerned only with a specific records rather than with every record in the file. The file system must be able specific record independent of the record's location in the file.

Indexed sequential files provide the capability, while retaining the ability to access the records sequentially. Each record header includes an integer index field. The indexed sequential file system uses a more general read/write interface than do pure sequential files. Like-get Record (file ID, index), put record (file ID, record), delete Record (file ID, index).

Index sequential files force the program to manage the indices so that the can access desired records. It is widely used in business computing for files that have very large number. of records, particularly if the records are often referenced in a non-sequential manner. It is also be accessed sequentially if the application intends to systematically process every record in the file.

8.1.2.2 Inverted files : There are many applications in which the index field must be searched by the application before a record can be retrieved. In general, each record access would cause a storage I/O operation, since the records would not usually be stored in the same physical block. The number of operations could be substantially reduced by extracting the index field from each record and placing it in an index table. This table has a conceptual form but it is maintained by the file system, not the application.

The external index table can be generalized so that an entry points to various records or field in the file. When a record is placed in the file, keyword in the record are extracted and placed in the index table with a pointer back into the record to where the keyword appears. This is called an

Inverted files, since records are accessed based on their appearance in the table rather than their logical location or address.

8.1.3 Data base management systems

A database is a very highly structured set of information, typically stored access several files and with the organization optimized to minimize access time. The DBMS enables the program user to define complex data types in terms of **data schema.** These schema specifications are then used by the database administrator to organize the way information is to be stored in files so the data can be accessed rapidly. some DBMS use the normal files provided by the OS for generic use. obviously, a low-level file system will be better suited to DBMS support than will a structure file system. This is because the low-level system does not presume any particular structure on files; it expects the application the DBMS to do this.

While conceptually every DBMS uses the file system to implement its functions, in many cases it has its own storage device block organization and access routines; hence, it completely by passes the file system in order to work directly with devices. This enables the DBMS to be more efficient in the way it accesses the storage devices. However, it precludes information stored in the database from being accessibly using the file manager's interface to the storage device.

8.1.4 Multimedia storage

Multimedia documents are highly structured files (or set of files) designed to contain information represented as numbers, characters, formatted text, executable programs, graphics, images, audio representations and so on. Storage requirements (containing images) are five or more orders of magnitude higher than traditional alphanumeric information For example formatted text page of characters might require 0.5 to 1 KB to same the information, but a similar sized color image might require 10 KB of storage. These very large information contains cause operating system designers to rethink the mechanism for storing files and for copied the information from the storage device to memory.

8.2 Low level file Implementations

The byte stream file system provides a minimal mechanism to enables a process to read from and write to information on storage devices. The storage device may be a sequentially accessed device, such as a magnetic tape, or a randomly accessed device, such as a magnetic disk.

A tape realization of the low-level file abstraction requires the logical byte stream to be mapped to logical blocks, which are mapped onto the physical records on a tape. The contiguous bytes are grouped into logical records and then are stored in physical records. Bytes O through K-1 are stored in logical record O, which is stored in physical record O, where K is the number of bytes in a tape physical record.

The order of logical blocks on the byte stream maps to contiguous physical blocks in a magnetic tape realization of the low-level file. So for K-byte physical blocks on the tape, bytes b_0 to b_{K-1} are

stored in physical block O of the tape, bytes b_k to b_{2K-1} are stored in physical block 1 on the tape, and so on.

A disk realization also maps logical blocks of contiguous bytes from the byte stream, but the mapping of logical to physical blocks will not normally be contiguous on the disk. The disk realization of the low-level file system must provide a mechanism for managing a collection of blocks to store the bytes of a particular file so that they can be accessed as if they were stored as a contiguous byte stream.

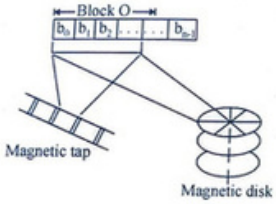


Figure 8.2 : Low-level file system architecture

8.3 Storage Abstraction

The low-level file systems attempt to avoid encoding record-level functionality into the file manager. If the applications will typically use very large or very small records, a generic file manager cannot be written to take advantage of that fact; this would not be a general purpose solution. Any strategy that is widely used by applications can usually be implemented to be more efficient if it is implemented in the Operating System. The modern open operating system has been toward low level file systems on the other hand, proprietary Operating System for machine aimed at particular application domains (such as transaction processing) will provides a higher layer file system. In this section we have some of classic approached for supporting storage abstractions.

8.3.1 Structured sequential files

Structured sequential files contain collection of logical records. Records are referenced using an index as in byte stream files. The two types of files are differ in that structured sequential files cause a full record to be read from or written to. The implementation of structure sequential file manager is logically the same as a byte stream file manager. If the structured sequential file provides an option of inserting a record at an arbitrary point in the file, then the manager must be designed with the same issues in mind as for the **insert byte** operation.

8.3.2 Indexed sequential files

In indexed sequential files, each record contains an index field used to select it from the file An application program provides an index value with each **read** or **write** operation. The file manager implements a mechanism to search of storage device for finding physical block which contains record. The implementation may use the same file structure as structured sequential files. The new task for the file manager is to manage the mapping of records to blocks. Insertion and deletion may occur at any logical location in the file. So, internal fragmentation and compaction issues arise in indexed sequential files.

The file manage may be implemented with a mechanism for direct access of records. Rather than the records being kept in a sequentially accessed data structure, they can be placed in different block and referenced from an index for the file. This require that the file manage keep a table for each open file and map the index to the block number that contains the record. This direct access implementation can substantially reduce the access times for records in the file. The cost is primarily the space cost of the index, since complexity of the approach is not substantial compared to the sequential approach. Read and write operations are for more complex for indexed sequential files than for stream-oriented files, since the file manager must access records according to the index value.

8.3.3 Database Management System

A database management system (DBMSs) contacted an entire area of computer science itself. DBMSs are built on a fundamental **storage manager,** which replaces the file manager. The storage manager interface is at a relatively low level so as to enable the DBMS designer to directly manipulate storage devices. This avoids performance penalties associated with generalized operation in favour of specialized strategies for databases. DMBS depends on database administrator to choose the organization of records across and within files. RDBMS requires that there be a means for very efficient search of records, while object oriented databases requires the access methods be defined by application code conventional file system interfaces cannot support either of these functions; they require the storage manager to replace the fill manager in order to accommodate them. By using storage manager rather than file manager a system that supports a DBMS and a file system usually cannot allow data to be stored in a database yet still be accessed via the file interface.

8.3.4 Multimedia Documents

Multimedia files make demands on the storage system like other database (object oriented database are often justify by the need to support multimedia documents). Multimedia documents tend to be implemented using abstract data types, it is compulsory to use application defined access methods to reference different parts of the document. This argues for a specialized storage manager interface to the secondary storage devices, the same that is used in database implementation.

The need for high band width through put is at odds with the basic operation of contemporary file managers. If a multimedia documents contains 10 MB record (with an image) and a disk block is 4 KB, then a read operation will span over 2400 blocks. The block allocation strategy will have

a major impact on the rate at which the image can be transferred. The contiguous allocation strategy is best for transfer time, although the association fragmentation problem is a costly trade-off so the multi media document transfers are not provided with high performance service. Operating System technology is evoluing to meet the performance requirements of multimedia files and documents because there are no widely accepted Operating System that provide high performance support for multimedia files.

8.4 Memory Mapped Files

Memory mapped files are especially useful when a file is shared. In sharing the file contents are bound to virtual addresses in the two or more address spaces of processes. If two or more process have the file opened at the same time, the OS recognizes this and manages both processes's page table pointers so that they reference one copy of the disk block in memory.

For example, suppose P_i and P_j have opened a memory mapped file. As shown in figure below blocks i to i + 3 have been cached into memory. P_i is using blocks i + 2 and i + 3 while P_j is using blocks i, i + 1 and i + 2. When either process changes a file block, the other process immediately sees the change, since the change is captured in the cached in storage blocks. The memory mapped pages could be paged out at any time. However, performance will be best if the pages being used by either process remain in memory.

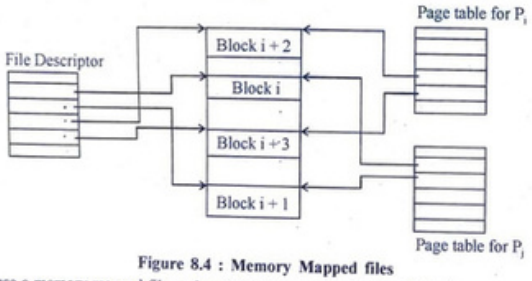


Figure 8.4 : Memory Mapped files

To use a memory mapped file, a thread in a process must do the following-

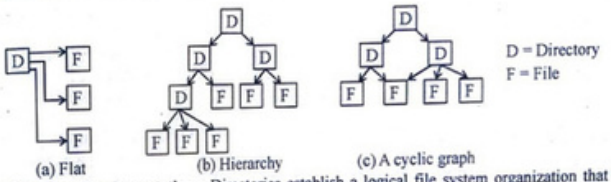
1. Obtain a handle to the file by creating or opening it.
2. Reserve virtual addresses for the file.
3. Establish a mapping between the file and the virtual address space.
4. Release the mapping when it is no longer being used.

Directories : A directory is a set of logically associated files and other directories of files. Directories are the mechanism by which humans organize the set of files in the entire system. A directory is a logical block of a set of files and possibly nested directories. The file manager

provides users a set of commands to administer a directory, including these-

- **Enumerate :** To return a list of files and nested directories,
- **Copy :** Creates a new duplicate of an existing files.
- **Rename :** Changes the symbolic name of an existing file.
- **Delete :** Removes the identified file from directory & destroys it.
- **Traverse :** To navigating from one directory to another directory in hierarchy order traverse operation enables to contain majority of directories and subdirectories.

8.5 Directory Structures :



(a) Flat (b) Hierarchy (c) A cyclic graph

Directory Implementation : Directories establish a logical file system organization that is independent of the device organization. The first task in the directory design is to map the logical file organization to the storage devices. This is done by partitioning the organization and assigning the partitions to devices. Each device keeps a directory of the part of the file system that is implements.

Device Directories : The device directory is an index of the files stored on the device. The simplest structure of the device directory is a sorted linear list with elements of the form-

<file name, file descriptor location>

File name is the relative name of the file, and **file Descriptor location** is the address where the file descriptor is stored on the device. This information is used by the file manager for the **open** command.

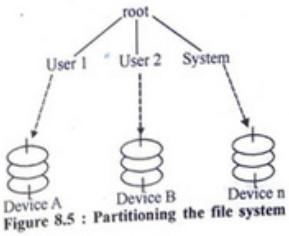


Figure 8.5 : Partitioning the file system

As disk sizes increase, it is useful to divide the physical disk into two or more logical disks. Administrators find this useful because it simplifies the mechanical process of archiving the disk's contents-only a portion of the disk needs to be dumped at a time. On personal computer, disk partitions are used to accommodate different O/S. In unix, they are used to define smaller **file system** that are treated much like a set of files on a particular device.

8.6 File Directory

A general directory is implemented as a structured file in which each record is an abbreviated version of a file descriptor. For example the record might contains the name, owner, protection state, length and a disk address for the file descriptor. Depending on the file implementation, the complete file descriptor may be distributed between a file descriptor kept with the file and the record in the directory.

Opening a file in a hierarchy directory : When a file is opened in a hierarchical directory, a descriptor for the file must be obtained using the directory that contains the file. In the UNIX absolute path name **/user/abc/books/OS/ch13** example, before **ch13** could be opened for reading or writing the system would need to find the file descriptor in **/user/abc/books/OS.** But before the entry can be found in OS, its directory descriptor must be found in **/user/abc/books,** and so on. Hence, when the file is opened using an absolute file name, the **open** routine must first search the root directory to find the entry for the first level directory **user** in the example. So, the first level directory in the path name is opened and searched for the second-level directory and so on.

Mounting Removable file system : The UNIX file manager incorporates a mechanism to allow file system to be combined into one directory hierarchy. This is especially useful for adding a set of files that are on a removable medium on to an existing file system on a fixed disk. The **mount** command appends a file system into an existing directory hierarchy. It does this by replacing a directory in the permanent file system by the root of the mountable file system when the corresponding replaceable medium is placed on the system.

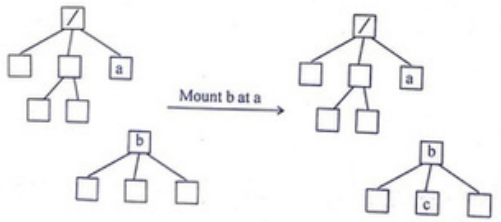


Figure 8.6 : The UNIX mount Command

Exercises

Part I (Very Short Answer)

1. What is file ? **[Raj. Univ. 2004, 2005]**
2. What are the attributes associated with a file. **[Raj. Univ. 2007]**
3. What is file descriptor ?
4. Why DBMS preferred ?
5. Why a file system supports indexed sequential files ?

Part II (Short Answer)

1. Explain file access mechanism.
2. What is a directory ? What are the way to implement directory ?
3. What is structured file ?
4. Explain multimedia storage ?
5. Explain device directories.

Part III (Long Answer)

1. Explain difference between sequential, linked & indexed files.
2. Explain the purpose of create, open and close in file system.
3. What do you understand by file management ? **[Raj. Univ. 2004, 2005]**

Chapter

9 Remote Files

Remote files : Network technology is the principal technology for distributed computation, processes can communicate with one another at band width than with traditional serial communication devices. In this chapter, we use the file abstraction of memory to consider how the OS takes advantage of networks to reference secondary storage devices located on machines across the network. This remote file technology is the foundation support for distributed computing provided by the OS.

9.1 Sharing Information Across the Network

The most important application of network technology is to have the OS enable a process on one machine to use the primary/Secondary memory on another machine. OS abstracts the secondary memory's use so that the memory can be referenced using a virtual memory model, in today's scenario OS should allow processes to read from and write to the memory on remote machines that are accessible through the network.

In basic **VON Neumann** architecture, a process must use separate interfaces to reference primary and secondary memories. Under these circumstances, a process should be able to write a memory address to the primary memory interface thus causing the address to be stored in a related primary memory location or another machine. It should also be able to read a remote file as if the file were stored in the local file system.

After the **ARPA/net** network, software was developed to implement primary knowledge of remote file reference. The earliest system facilities allowed a user to copy files over the network form one host to another by using explicit commands. Remote file technology evaluate today so it is possible to reference remote files using the local file management interface.

9.1.1 Explicit file copying system

Information can be shared across machines by explicit copying files from one machine to another across the communication network. In WAN communications, explicit file copy operation were the most common mechanism available to sharing information. When a process on a local machine needed information from a process on a remote machine the remote process wrote the info. to a file and then the user explicitly copied the file from the remote machine to the local machine.

Explicit operations are used in a conventional file manager to copy file from one place to another in a system directory using a simple shell command . In earliest network system this can be accomplished by using either the operating system point to point communication function or

transport layer software to connect with a remote machine and then having a surrogate process on the remote machine retrieve a copy of the target file and write it to the connection. After the local end of connection for the surrogate to begin sending a copy of the file over the connection. The local and accept the copy and stores it in the local file system for subsequent use. The **ARPnet ftp** and UNIX **uucp** commands represent examples of using the network and communication facilities.

There are other manual file transfer packages, used across the heterogeneous networks (network has different kinds of host machines). The ISO OSI File Transfer, Access and Management (FTAM) protocol is the standard ISO OSI mechanism for manual file transfer; **telnet** and **ftp** are widely used for the same purpose in computer networks.

9.1.1.1 The ARPA met file transfer protocol : The file transfer protocol (FTP) can be invoked from a user interface, the **ftp** program in UNIX systems to explicitly copy files between host machines on the a network by using TCP/IP. Each host that supports FTP starts a server process to accept service requests when an application process intends to use FTP, it executes **ftp** client code on the local machine.

FTP uses a control connection and a data connection for file transfer. The server's initial service request to arrive on well known port 21, so when a client wishes to use FTP, its TCP connection request is directed at

<net, host, 21>

When the connection has been opened the client & server use the control connection to exchange these commands and control responses.

When the data is transferred and client obtains a listing of the files in a directory. Each time the client issues a command causing this kind of data transfer, a data connection is opened to place the transfer. During FTP session, connection may be opened & closed more than once, depending on the nature of command.

9.1.1.2 The UNIX UUCP command : UUCP (Unix to Unix copy) is a unix program that exchange files using serial communication devices & dial up modems. The unix **uucp** command has same like the cp command except that one source or destination of the copy is a path name to and inside of, a remote machine.

System_name_1 ! system_name 2!.....system_name_N! n_pathnames

Each system name is the name of a UNIX machine stored in the UUCP routing table. If there is more than one system_name in remote file name, they define a path like_system_name_1 to system_name_N, where N_pathname defines a location for the file in the file system for machine system_name_N. Authentication applies to UUCP operation at each system UUCP is normally used to read a file from remote host. If the client write a file to a remote host, then the protection settings may allow the user only to log in to the remote machine and to perform a UUCP back to the original local machine.

9.1.2 Implicit file sharing

Implicit file sharing makes better use of existing file and secondary memory interfaces rather

than explicit file transfer. The multics style memory interface enables a programer to reference all primary and secondary memories using an extended primary memory interface. Since a primary memory address employs a segment and offset, the segment names can be mapped to file names. The memory manager in the OS automatically loads segments on reference and then binds physical addresses to the resulting two component address in the primary memory interface. All the information stored on the storage devices can be referenced as segments and offsets.

Ideally, when information stored at a remote site is to be accessed by a process the corresponding program could use one of the two customary memory interfaces. Using existing interfaces is the main feature of remote files, this approach uses the secondary memory interface to reference files stored on remote machines.

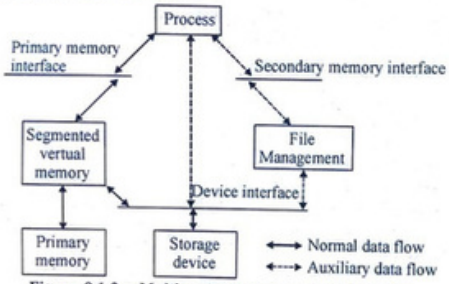


Figure 9.1.2 : Multics segmented memory Interface

9.1.3 The Remote storage Interface

The device driver interface is used primarily by system software rather than application software. The application program access to local storage devices occurs over the local file interface. For consistency, the remote disk and remote file interfaces also use the local file interface with little or no change for remote storage access. This is for by reimplementing the local file manager so that it can either reference local files in the normal way or reference remote storage using the remote service interface. Low-level file system interfaces are easier to implement with remote file services than are file system, access to a remote file is said to be transparent.

Remote disk always provide location transparency, meaning that the access to the remote storage device containing the file is handled entirely by the file system. By the application program's view, references to remote disks are the same as references to a local disk, since both use the local file interface. In the extreme the local machine may not even be configured with a disk, meaning every file (and disk) access is to a remote machine. If this too may be transparent depending on the relative speed of a local disk and the remote disk and network.

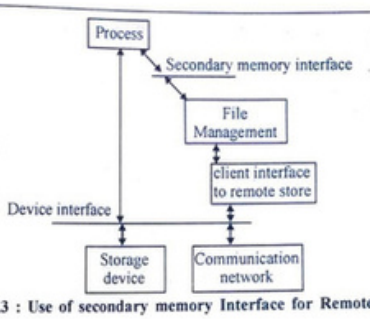


Figure 9.1.3 : Use of secondary memory Interface for Remote Memory

9.1.4 Distributing the work

In implicitly file sharing local file interface assumes that the system implements the logic that is shown in diagram. In figure, a local server interface implementing functionality to the remote machine using the network controller and protocols.

Here is an elaboration of previous remote memory interface diagram, which is a client server architecture with the application program executing on the client machine and the remote file facility acting as a server. The client machine OS implements part of the function for implementing remote files with in the "remote access" module. In fact this local part of the remote server may be a significant part of the remote file service. The approach is to keep the local file manager as similar as possible to one with as similar as possible to one with no remote file capability so, the " remote access" model is the part of the OS that is responsible for implementing the remote file operations network interactions, transference and other features required when using remote files. However, it allows an application process to simply invoks the client part of the file server without knowing the details of the remote access.

The "remote storage" module in the server machine accepts storage commands from th client remote access module performs the operation on its storage devices, and then returns resul to the client remote access module.

9.2 Remote Disk system

The organization of functions between a client workstation and the remote disk server. The client machine incorporates a local virtual disk driver (VDD) to replace the conventional local disk driver for remote accesses. This software works like a local disk driver for file system to reference an abstract storage device. The remote disk application (RDA). The remote disk applications accepts commands from the VDD, applies them to the server's disk drive, and returns the result to the client VDD. It must contain local disk addresses for normal local file accesses and virtual disk addresses for remote file accesses.

Just as a physical disk has a devices address and a set of logical block addresses, the remote disk has a transport layer address, <net, host, port> and a set of logical block addresses. This is accomplished by using the transport layer.

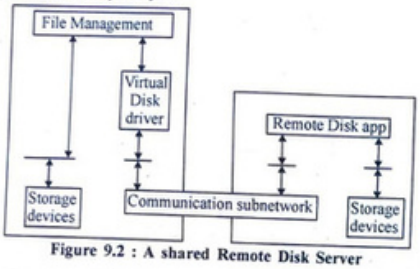


Figure 9.2 : A shared Remote Disk Server

Exercise

Very Short Question (Up to 20 words)

1. What is file sharing ?
2. What is Remote files ?.
3. What is explicit file copy ?
4. Write a command of UUCP ?
5. What is remote storage.

Short Question (Up to 80 words)

1. Explain Network file system.
2. What is distributed information system ?
3. Explain FTP.
4. What is remote disk system ?
5. What is remote storage ?

Long Questions

1. Write notes on following :
   (i) File sharing
   (ii) Remote file system
2. Explain explicit file copying system concept.
3. Explain implicit file sharing concept with diagram.

Chapter

10

Viruses and Worms

10.1 Computer virus

A computer virus is a small program that can copy itself and infect the system without the permission or knowledge of the user. How we recogniz virus ? If it meets two criteria :

1. It must execute itself

2. It must replicate itself, it may replace other executable files with a copy of the virus infected file.

Viruses can increase their chances of spreading to other computers by infecting files on a network file system or a file system that is accessed by another computer, It can carried it on a removable medium such as a floppy disk, CD, DVD or USB drive.

The term "computer virus" is sometimes used as a catch-all phrase to include all types of malware. Malware includes computer viruses, worms, Trojan horses, most root kits, skyware, dishonest adware, crime ware and other malicious and unwanted software, including true viruses. There are few kinds of known viruses. File infector viruses, boot sector viruses, master boot record viruses, multi partite viruses and macro viruses. This classification depends upon their target area a mode of operation.

10.2 Computer Worm

A computer worm is a self replicating computer program from system to system without the use of a host file. It uses a network to send copies of itself to other nodes. They contain malicious code that can cause major damage to the files, software and data on a computer. There are several ways that worms can get into a computer. The most common is through the Internet & E-mail, which contains the names & addresses of other reachable machines. After gaining access to a mailing, the worm sends copies of itself (an executable program) to some or all listed machines, the worm may or may not check whether the new destination is already infected. If there is no infection or if no check is made, the worm will attempt to gain access to the local mailing lists for further spreading. As a result, computers on the network become infected with copies of executing & travelling worms. This infection spreads quickly, so the consuming time & network band width will drown and any other work and communication can proceed slowly or none.

A computer worm differs from a computer virus in that a computer worm can run itself a virus needs a host program to run, and the virus code runs as a part of the host program. A computer worm can spread without a host program, although some modern computer worms also use files to hide inside.

10.2.1 Types of Viruses

Viruses can be divided into two types based on their behavior when they are executed.

1. Non resident viruses : Non resident viruses immediately search for other hosts that can be infected, infect those targets, and transfer control to the application program they infected.

2. Resident viruses : Resident viruses do not search for hosts when they are started. Instead a resident virus loads itself into memory on execution and transfers control to the host program. The virus stays active in the background and infects new hosts when those files are accessed by other programs or the Operating System itself.

10.2.2 Types of worms

Following are the different types of computer worms.

1. E mail worm : Spreading goes through infected e-mail messages. Any form of attachment or link in an e-mail may contain a link to an infected website, firstly activation starts. When the user clicks on the attachment while in the second case the activation starts. When clicking the link in the e-mail

2. Instant Messaging Worms : Spreading through instant messaging applications by sending links to infected websites to everyone on the local contact list.

3. Internet worms : They scan all available network resources using local Operating System services and scan the internet for vulnerable machines, attempt will be made to connect to these machines and gain full access to them.

4. IRC worms : Same spreading method is used to sending infected files or links to websites. Infected files sending is less effective as the recipient needs to confirm receipt, same the file and open it before infection will take place.

5. Sharing network worms : Sharing network worms copies itself into a shared folder, most likely located on the local machines.

10.3 Dealing with viruses and Worms

1. New viruses can spread extremely quickly, so one should install anti virus software on desktops and servers and ensure they are kept up to date frequently at short notice. To protect e-mail, e-mail filtering software at the e-mail gateway can be used to avoid threats of e-mail borne viruses span & spyware.

2. Blocking the files immediately that often carry viruses. These include EXE, COM, SHS, PIF, SCR, VBS and BAT file types. It is unlikely that the organization will ever need to receive files of these types from the outside world.

3. Blocking all the files with more than one file type extension. Some viruses disguise the fact that they are programs by using a double extension such as TXT.VBS after the file name.

4. Check the programs received from the outside world via e-mail, that they are virus free, properly licensed unlikely to conflict with existing software, and suitable.

5. Use a firewall on computer connected to the internet to protect laptops, desktops from viruses.

6. Subscribe to an e-mail alert service against viruses. It offers virus identities that will enable the anti virus software to detect them.

7. Back up the data regularly and check them that the backup were successful. One way also find a safe place to store the backups, perhaps even offset in case of fire.

8. Regularly stay up to data with software patches that can close loopholes. Watch out for security news and download patches that can make the system vulnerable to viruses or worms.

10.4 Security Design principles

To present security violations, a set of protection mechanism must be held on system. Sophisticated users of most systems are aware of at least one way to crash the system, denying other users authorized access to stored information. Some useful principles can guide the design and contribute to an implementation without security flaws. Following are some design principles that may be particularly protect mechanism-

(i) Economic Approach : The design should be as simple & small as possible for protection mechanism. For this reason that design and implementation errors can result in unwanted access path will not be noticed during normal use.

(ii) Fail safe approach : The default situation is lack of access, and the protection scheme identifies conditions under which access is permitted. The alternative in which mechanisms attempt to identify conditions under which access should be refused presents the wrong psychological base for secure system design.

(iii) Complete medium : Every access to every object must be cheeked for authority. This in primary base of protection system. It forces a system wide view of access control, which in addition to normal operation includes initialization, recovery, shutdown, maintenance.

(iv) Open design approach : The design should be open. This mechanisms should not depend on the ignorance of attackers, but rather on the possession of specific, more easily protected keys or passwords. This decoupling of protection mechanisms from protection keys, permits the mechanisms to be examined by many review without concern that the review may itself compromise the safeguards.

(v) Privilege approach : Once the mechanism is locked, the two keys can be physically separated and distinct programs, organizations or individual made responsible for them. From then on no single accident, deception or breach of trust is sufficient to compromise the protected information.

(vi) Least privileges provided : Every system and program should operate using the least set of privilages necessary to complete the job. This principle limits the damage that can result from an accident or error. It also reduces the number of potential interactions among privileged programs to the minimum for correct operation, so that inintentional, unwanted or improper uses of privilege are less likely to occur.

(vii) Least common mechanism : Minimize the amount of mechanism common to more than one user and depended on by all uses. Every shared represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security. If one or a few users are not satisfied with the level of certification of the function, they can provide a substitute or not use it at all.

10.5 Authentication

Authentication is the process of identifying an individual usually based on a usernance and password. In security system authentication is distinct from authorization, which is the process of giving individuals access to system objects based on their identity.

Computer security authentication means. Verifying the identity of a user logging onto a network. Password digital certificates smart card and biometrics can be used to prove the identity of the user to the network. It also includes verifying message integrity, e-mail authentication and MAC (Message Authentication Code), checking the integrite of a transmitted message.

There are human authentication, challenge response authentication, password digital signature, IP spoofing and biometerics, human authentication is verify that a person initiated the transaction, not the computer. Challenge response authentication is an authentication method used to prove the identity of a user logging onto the network. Biometric is a more secure authentication then typing password or smart cards that can be stolen. Password is a secret word or code used to serve as a security measure against unauthorized access to data.

The two major applications of digital signature are for setting up a secure connection to a website and verifying the integrity of files transmitted. IP spoofing refers to inserting the IP address of an authorized user into the transmission of an unauthorized user in order to gain illegal access to a computer system.

10.5 Protection Mechanism

A control mechanism can be implemented for each resource of operation. A process should be allowed to access only those resources it has been authorized to access. A process should follow some principle so that it should be able to access the resources, which it currently needs it will protected and others will be totally safe, which is not required.

10.5.1 Access rights

At any instance, protection should be provided to those resources, which may be currently used by a process. Thus, a domain known as protection domain can be defined for a process. Such a domain will contain the set of objects and the types of operating that can be invoked an each object. This means that domain in a set of <object, rights> pair. 'Right' refers to the operations that can be invoked by a process on the on the corresponding object. These rights are access rights of a particular process on the corresponding object.

10.5.2 Access matrix

The access matrix model is the policy for user authentication and has several implementations like access control lists (ACLS) and capability lists. It is for checking that which user have access to what object or resources. So it is check the access rights of user on system resources. It consists list of objects, list of subject, function returning object type, matrix of resource columns and subject making the rows.

10.6 Protection of files

To protect files from accidental and deliberate damage some modern computer system provide methods for protect files. Computer that allow for multiple users implement file permissing to control who may or may not modify files an folders. A given user may be granted permissions to modify a file or folder, but not to delete it or a user may be given permission to create files or folders, but not to delete them. Permission may also be used to allow only certain users to see the contents of a file, it protects unauthorized competing or destruction of information in files and keep private information confidential. Another protection mechanism is read only flag. It is used to invisible certain files in hidden part of system. When this flag is turned on for a file, the file can be examined put is cannot be modified or erased.

10.7 Encryption

As computer networks gain popularity, more sensitive informatin is being transmitted over channels where eavesdropping and message interception is possible. Thus, the operating system should have some provision to fight such situations i.e. to protect the data that are transferred over the network.

Encryption is one such mechanism, which allows such data to be scrambled so that even if some one intercepts it on the network, it is not readable to him/her. Thus, the basic purpose of Encryption is to make the data transfer secure over the network.

encryption works :

1. It transforms information from ("Clear" or "plain text") to coded information ("cipher text"), which cannot be read by outside parties.
2. This transformation process is controlled by an algorithm and a key.
3. This process should be reversible so that the intendal intended can read the infomation transmitted to him in the form of cipher text. But for this a decryption mechanism which decrypts (decodes) the cipher text to plain text is must. In Figure, Shows a general encryption and decryption mechanism.

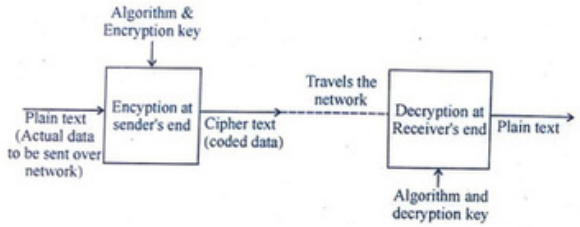


Figure 10.7 : General Encryption and decryption mechanism

The main challenge in using this approach is the development of encryption schemes that are impossible to break.

These are two kinds of encryption.

1. 'Symmetrical Encryption' or secret key which uses a single key to encypt and decrypt the transmitted data.
2. 'Asymmetrical Encryption' which uses 'Private key', in which one key is used to encrypt and another to decrypt the transmitted data.

Encryption Schemes :

- One scheme (that uses single secret key) called data encryption standard (DES), was once adopted as standard for encryption. This scheme, how ever, suffers from key distribution problem.

Before communication can take place, the secret keys must be sent securely to both the senaler and the receiver. This task it self is prone to the threat of eves dropping by the unauthorized users who may be trying hard to know the secret keys.

- A solution to the problem encountered in DES is to use a scheme known as public key encryption scheme. In this scheme, each user has both a public and a private key, and two users can communicate knowing only each other's public key. The algorithms based on this concepts are believed to be almost unbreakable and therefore, this scheme is the most preferred and relaible scheme, for secure network communication.

10.8 Protection of user files

Let's look at some ways to protect your all important user data from loss and or unauthorized access.

10.8.1 Back up early and often

The most important step in protecting your files from loss is to back it up regularly. How often should you back up ?

You can use the back up utility built into windows to perform basic backups. You can use mode to simplify the process of creating and restoring backups or you can configure the backup settings manually and you can schedule backup jobs to be performed auto matically.

10.8.2 Use file level and share level security

To keep others out of your files, the first step is to set permissions on the data files & folders. If you have data in network shares, you can set share permissions to control access the files across the network with windows 2000/ xP. This is done by clicking the per missions button on the sharing tab of the files or folder's sheet.

However, these share-level permissions won't apply to some one who is using the local computer on which the data is stored. If you share the computer with some one else, you will have to use file level permission (also called NTFS permission, because they are available only for files & folders stored on NTFS-formatted partitions.

10.8.3 Use EFS encryption

Windows support the encryption file system (EFS). You can use this built in certificate based encryption method to protect individual files and folders stored on NTFS partitions. EFS uses a combination of asymmetric and symmetric encryption for both security & performance.

10.8.4 Password protect files

Many applications will allow you to set password on individual documents. To open the document you must enter the password. To password protect a document in microsoft. Unfortunately, password protection is easy to crack. There are programs on the market designed to recover office password, such as Advanced office password recovery.

10.8.5 Make use of public key in infrastructure

PKI is a system for managing public/Private key pairs and digital certificates. Because keys and certificates are issued by a trusted thirt party, certificate based security is stronger.

You can protect data you want to share with some one else by encrypting it with the public key of its intendent recipient, which is available to any one. The only person who will be able to decrypt it is the holder of the private key that corresponds to that public key.

Exercise

Part I (Very Short Answer)

1. Which is the most common security method ? [2008]
2. How many types of worms ?
3. Give example of any two type of viruses.
4. What is security ?
5. Explain system threats ?

Part II (Short Answer)

1. Write short notes on authentication.
2. Explain virus and worms and explain the differences of them. [2005]
3. Discuss the major program and system threats [2007]
4. Explain protection mechanism.
5. Explain the types of viruses.

Part III (Long Answer)

1. How you deal with viruses & worms.
2. Explain file protection.
3. What are the basic principles of security design ?
4. Explain the incryption in details ?
5. Describe the protection of the fiels ?