UNIT VII

FILE SYSTEM INTERFACE

7.1 The Concept Of a File

A file is a named collection of related information that is recorded on secondary storage. The information in a file is defined its creator. Many different types of information may be stored in a file.

File attributes:-

A file is named and for the user's convince is referred to by its name. A name is usually a string of characters. One user might create file, where as another user might edit that file by specifying its name. There are different types of attributes.

1)**name:-** the name can be in the human readable form.

2)**type:-** this information is needed for those systems that support different types.

3)**location:-** this information is used to a device and to the location of the file on that device.

4)**size:-** this indicates the size of the file in bytes or words.

5)**protection:-**

6)time,date, and user identifications:-

the information about all files is kept in the directory structure, that also resides on secondary storage.

File operations:-

Creating a file:-

Two steps are necessary to create a file first, space in the file system must be found for the file. 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 system.

Writing a file:-

To write a file give the name of the file, the system search the directory to find the location of the file. The system must keep the *writer* pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

Reading a file:- to read from a file, specifies the name of the file and directory is search for the associated directory entry, and the system needs to keep *read* pointer to the location in the file where the next read is to take place. Once the read has taken place, read pointer is updated.

Repositioning with in a file:-

The directory is searched for the appropriate entry and the current file position is set to given value. this is also known as a file seek.

Deleting a file:- to delete a file , we search the directory for the name file. Found that file in the directory entry, we release all file space and erase the directory entry.

Truncate a file:- this function allows all attributes to remain unchanged(except for file length) but for the file to be reset to length zero.

Appending:- add new information to the end of an existing file .

Renaming:- give new name to an existing file.

Open a file:-if file need to be used, the first step is to open the file, using the *open* system call.

Close:- close is a system call used to terminate the use of an already used file.

File Types:-
A common technique for implementing file type is to include the type as part of the file name. The name is split in to two parts
1) the name 2) and an extension .
the system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

7.2 : ACCESSMETHODS:-
There are several ways that the information in the file can be accessed.
1)sequential method 2) direct access method 3) other access methods.
1)sequential access method:-
the simplest access method is S.A. information in the file is processed in order, one after the other. the bulk of the operations on a file are reads & writes. It is based on a tape model of a file. Fig 10.3
2)Direct access:- or relative access:-
a file is made up of fixed length records, that allow programs to read and write record rapidly in no particular order. For direct access, file is viewed as a numbered sequence of blocks or records. A direct access file allows, blocks to be read & write. So we may read block15, block 54 or write block10. there is no restrictions on the order of reading or writing for a direct access file. It is great useful for immediate access to large amount of information.
The file operations must be modified to include the block number as a parameter. We have read n, where n is the block number.
3)other access methods:-
the other access methods are based on the index for the file. The indexed contain pointers to the various blocks. To find an entry in the file , we first search the index and then use the pointer to access the file directly and to find the desired entry. With large files. The index file itself, may become too large to be kept in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index files which would point to the actual data iteams

7.3 Directory Structures:-
operations that are be on a directory (read in text book)
single level directory:-
the simple directory structure is the single level directory. All files are contained in the same directory. Which is easy to understand. Since all files are in same directory, they must have unique names.
In a single level directory there is some limitations. When the no.of files increases or when there is more than one user some problems can occurs. If the no.of files increases, it becomes difficult to remember the names of all the files. FIG 10.7
Two-level directory:-
The major disadvantages to a single level directory is the confusion of file names between different users. The standard solution is to create separate directory for each user.
In 2-level directory structure, each user has her own user file directory(ufd). Each ufd has a similar structure, the user first search the master file directory . the mfd is indexed by user name and each entry point to the ufd for that user.fig 10.8

To create a file for a user, the O.S search only that user's ufd to find whether another file of that name exists. To delete a file the O.S only search to the local ufd and it can not accidentally delete another user's file that has the same name.
This solves the name collision problem, but it still have another. This is disadvantages when the user wants to cooperate on some task and to access one another's file . some systems simply do not allow local user files to be accessed by other user.
Any file is accessed by using path name. Here the user name and a file name defines a path name.
Ex:- user1/ob
In MS-DOS a file specification is
C:/directory name/file name

Tree structured directory:-

This allows users to create their own subdirectories and to organize their files accordingly. here the tree hasa root directory. And every file in the system has a unique path name. A path name is the path from the root, through all the subdirectories to a specified file.FIG 10.9.
A directory contains a set of subdirectories or files. A directory is simply another file, but it is treated in a special way. Here the path names can be of two types. 1)absolute path and 2) relative path.
An absolute path name begins at the root and follows a path down to the specified file, giving the directory name on the path.
Ex:- root/spell/mail/prt/first.
A relative pathname defines a path from the current directory ex:- prt/first is relative path name.

A cyclic- graph directory:-

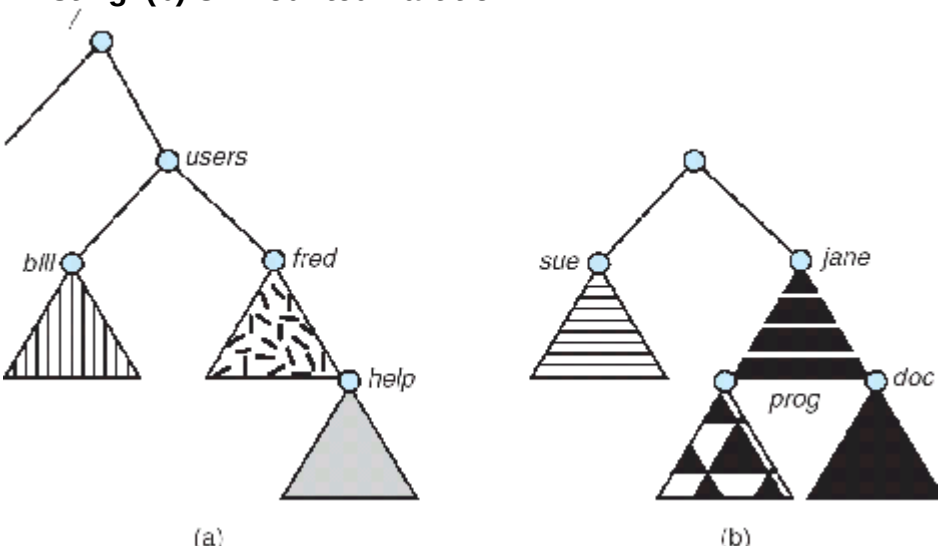
Consider two programmers who are working on a joint project. The files associated with that project can be stored in a sub directory , separating them from other projects and files of the two programmers. The common subdirectory is shared by both programmers. A shared directory or file will exist in the file system in two places at once. Notice that a shared file is not the same as two copies of the file with two copies, each programmer can view the copy rather than the original but if one programmer changes the file the changes will not appear in the others copy with a shared file there is only one actual file, so any changes made by one person would be immediately visible to the other.
A tree structure prohibits the sharing of files or directories. An acyclic graph allows directories to have shared subdirectories and files
FIG 10.10 . it is more complex and more flexiable. Also several problems may occurs at the traverse and deleting the file contents.
.

7.4:File System Mounting

A file system must be **mounted** before it can be accessed

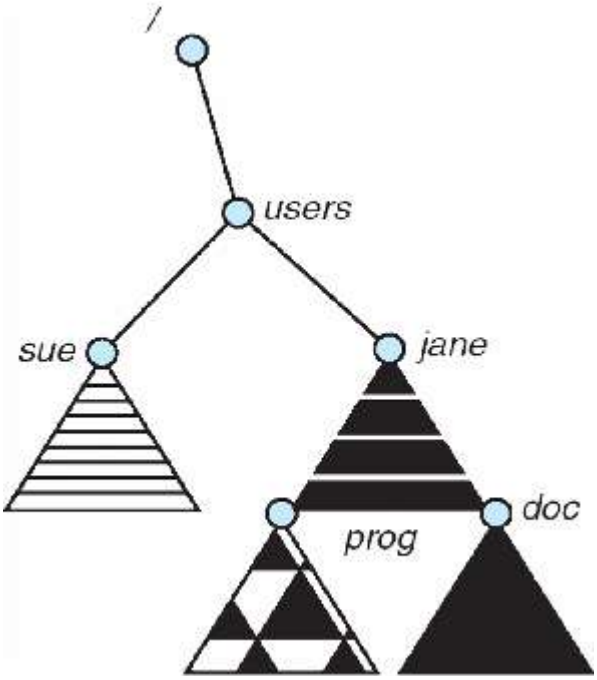
A unmounted file system (i.e. Fig. 11-11(b)) is mounted at a **mount point**

(a) Existing. (b) Unmounted Partition



(b) (a) (b)

Mount Point



7.5:File Sharing

Sharing of files on multi-user systems is desirablenSharing may be done through a **protection** schemenOn distributed systems, files may be shared across a networknNetwork File System (NFS) is a common distributed filesharing method

File Sharing – Multiple Users

User IDs identify users, allowing permissions and protections to be perusernGroup **IDs** allow users to be in groups, permitting group access rights

File Sharing – Remote File Systems

nUses networking to allow file system access between systems

lManually via programs like FTP

lAutomatically, seamlessly using **distributed file systems**

lSemi automatically via the **world wide web**

n**Client-server** model allows clients to mount remote file systems from servers

lServer can serve multiple clients

lClient and user-on-client identification is insecure or complicated

l**NFS** is standard UNIX client-server file sharing protocol

l**CIFS** is standard Windows protocol

lStandard operating system file calls are translated into remote calls

nDistributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

File Sharing – Failure Modes

Remote file systems add new failure modes, due to network failure, server failure

Recovery from failure can involve state information about status of each remote request

Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- **Consistency semantics** specify how multiple users are to access a shared file simultaneously
  - Similar to Ch 7 process synchronization algorithms
    - Tend to be less complex due to disk I/O and network latency (for remote file systems
  - Andrew File System (AFS) implemented complex remote file sharing semantics
  - Unix file system (UFS) implements:
    - Writes to an open file visible immediately to other users of the same open file
    - Sharing file pointer to allow multiple users to read and write concurrently
  - AFS has session semantics
    - Writes only visible to sessions starting after the file is closed

7.6:Protection

File owner/creator should be able to control:
- what can be done
- by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Protection:-]
When the information is kept in the system the major worry is its protection from the both physical damage (Reliability) and improper access(Protection).
The reliability is generally provided by duplicate copies of files.
The protection can be provided in many ways . for some single system user, we might provide protection by physically removing the floppy disks . in a multi-user systems, other mechanism are needed.
1) **types of access:-**
if the system do not permit access to the files of other users, protection is not needed. Protection mechanism provided by controlling accessing. This can be provided by types of file access. Access is permitted or denied depending on several factors. Suppose we mentioned read that file allows only for read .
Read:- read from the file.
Write:- write or rewrite the file.
Execute:- load the file in to memory and execute it.
Append:- write new information at the end of the file.
Delete:- delete the file and free its space for possible reuse.

FILE SYSTEM IMPLEMENTATION

7.7:File allocation methods:-

There are 3 major methods of allocating disk space.

1) **Contiguous allocation:-**

1) The contiguous allocation method requires each file to occupy a set of contiguous block on the disk.

2) Contiguous allocation of a file is defined by the disk address and length of the first block. If the file is 'n' block long and starts at location 'b' , then it occupies blocks b,b+1,b+2,…..,b+n-1;

3) The directory entry for each file indicates the address of the starting block and length of the area allocated for this file. Fig **11.3**

4) Contiguous allocation of file is very easy to access. For the *sequential access* , the file system remembers the disk address of the last block referenced and, when necessary read next block. For *direct access* to block 'i' of a file that starts at block 'b' , we can immediately access block b+i. Thus both sequential and direct access can be supported by contagious allocation.

5) One difficulty with this method is finding space for a new file.

6) Also there are many problems with this method

a) **external fragmentation:-** files are allocated and deleted , the free disk space is broken in to little pieces. The E.F exists when free space is broken in to chunks(large piece) and these chunks are not sufficient for a request of new file.

There is a solution for E.F i.e compaction. All free space compact in to one contiguous space. But the cost of compaction is time.

b) Another problem is determining how much space is needed for a file. When file is created the creator must specifies the size of that file. This becomes to big problem. Suppose if we allocate too little space to a file , some times it may not sufficient.

Suppose if we allocate large space some times space is wasted.

c) Another problem is if one large file is deleted, that large space is becomes to empty. Another file is loaded in to that space whose size is very small then some space is wasted . that wastage of space is called internal fragmentation.

2) **Linked allocation:-**

1) Linked allocation solves all the problems of contagious allocation. With linked allocation , each file is a linked list of disk blocks, the disk block may be scattered any where on the disk.

2) The directory contains a pointer to the first and last blocks of the file. **Fig11.4**

Ex:- a file have five blocks start at block 9, continue at block 16,then block 1, block 10 and finally block 25. each block contains a ponter to the next block. These pointers are not available to the user.

3) To create a new file we simply creates a new entry in directory. With linked allocation, each directory entry has a pointer to the first disk block of the file.

3) There is no external fragmentation with linked allocation. Also there is no need to declare the size of a file when that file is created. A file can continue to grows as long as there are free blocks.

4) But it have disadvantage. The major problem is that it can be used only for sequential access-files.
5) To find the I th block of a file , we must start at the beginning of that file, and follow the pointers until we get to the I th block. It can not support the direct access.
6) Another disadvantage is it requires space for the pointers. If a pointer requires 4 bytes out of 512 byte block, then 0.78% of disk is being used for pointers, rather than for information.
7) The solution to this problem is to allocate blocks in to multiples, called clusters and to allocate the clusters rather than blocks.
8) Another problem is reliability. The files are linked together by pointers scattered all over the disk what happen if a pointer were lost or damaged.

FAT(file allocation table):-

An important variation on the linked allocation method is the use of a file allocation table.
The table has one entry for each disk block, and is indexed by block number. The FAT is used much as is a linked list.
The directory entry contains the block number of the first block of the file. The table entry contains the block number then contains the block number of the next block in the file. This chain continuous until the last block, which has a special end of file values as the table entry. Unused blocks are indicated by a '0' table value. Allocation a new block to a file is a simple. First finding the first 0-value table entry, and replacing the previously end of file value with the address of the new block. The 0 is then replaced with end of file value.

Fig 11.5

3)Indexed allocation:-

1) linked allocation solves the external fragmentation and size declaration problems of contagious allocation. How ever in the absence of a FAT , linked allocation can not support efficient direct access.
2) The pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order.
3) Indexed allocation solves this problem by bringing all the pointers together in to one location i.e *the index block.*
4) Each file has its own index block ,which is an array of disk block addresses. The I th entry in the index block points to the ith block of the file.
5) The directory contains the address of the index block. **Fig 11.6**
To read the ith block we use the pointer in the ith index block entry to find and read the desired block.
6) When the file is created, all pointers in the index block are set to nil. When the ith block is first written, a block is obtained from the free space manager, and its address is put in the ith index block entry.
7) It supports the direct access with out suffering from external fragmentation, but it suffer from the wasted space. The pointer overhead of the index block is generally greater than the pointer over head of linked allocation.

7.8:Free space management:-

1) to keep track of free disk space, the system maintains a free space list. The free space list records all disk blocks that are free.
2) To create a file we search the free space list for the required amount of space, and allocate that space to the new file. This space is then removed from the free space list.
3) When the file is deleted , its disk space is added to the free space list.
There are many methods to find the free space.

1) **bit vector:-**

The free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free the bit is 1 if the block is allocated the bit is 0.

Ex:- consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,25, are free and rest of blocks are allocated the free space bit map would be
001111001111110001100000010000……..
the main advantage of this approach is that it is relatively simple and efficient to find the first free block or 'n' consecutive free blocks on the disk

2) **Linked list:-**

Another approach is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. This first block contain a pointer to the next free disk block, and so on.
How ever this scheme is not efficient to traverse the list, we must read each block, which requires I/O time.
Disk space is also wasted to maintain the pointer to next free space.

3) **Grouping:-**

Another method is store the addresses of 'n' free blocks in the first free block. The first (n-1) of these blocks are actually free. The last block contains the addresses of another 'n' free blocks and so on. **Fig 11.8**
Advantages:- the main advantage of this approach is that the addresses of a large no.of blocks can be found quickly.

4) **Counting:-**

Another approach is counting. Generally several contiguous blocks may be allocated or freed simultaneously. Particularly when space is allocated with the contiguous allocation algorithm rather than keeping a list of 'n' free disk address. We can keep the address of first free block and the number 'n' of free contiguous blocks that follow the first block. Each entry in the free space list then consists of a disk address and a count.

7.9:Directory Implementation:-

1) Linear list:-

1) the simple method of implement ting a directory is to use a linear list of file names with pointers to the data blocks.
2) A linear list of directory entries requires a linear search to find a particular entry.
3) This method is simple to program but is time consuming to execute.
4) To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.

5) To delete a file we search the directory for the named file, then release the space allocated to it.
6) To reuse directory entry, we can do one of several things.
7) We can mark the entry as unused or we can attach it to a list of free directory entries.
Disadvantage:- the disadvantage of a linear list of directory entries is the linear search to find a file.