# UNIT-8

## File Structures

**File :—**

A collection of data under permanent storage is called file. (or)

File is a data structure on secondary storage which act as a non-volatile container for data.

**Properties for file : —**

* File is a name given to any kind of document stored in any type of storage device which can be read by the computer.
* A file is identify by a name followed by a filename extension.

**File structure :—**

- A pattern for arranging data in a file.
- It is a combination of representation of data in a file and operations for accessing the data.

**Primary goals for design of file structure:-**

- Minimize the no. of disk access.
- Maximize the space utilization.

## Fundamental file processing operations:—

### Physical file:—

A file as seen by the operating system and which actually exists on secondary storage.

### Logical file:—

A file as seen by the program.

### Relation b/w the physical & logical files:-

- The use of logical files allows a prog. to describe operations to be perform on a file without knowing what actual physical file will be used.
- The program may then be used to process any one of no. of different files that share the same structure.

NOTE:—

→ Program read & write data from file which is logical file only

→ Before a logical file can be used, it must be associated with a physical file. This act of connection is called "Opening" the file.

→ Data in the physical file is persistent.
→ Data in the logical file is temporary.
→ Logical file is identified within the program by a program variable (or) constant.

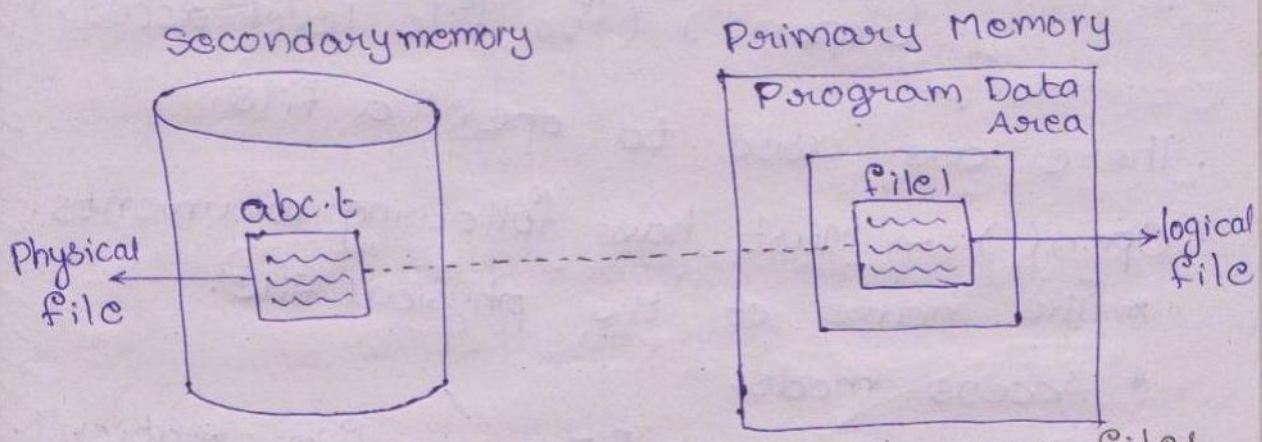


Fig:- Relationship b/w logical & Physical files

→ The name & form of the physical file are dependent on the O.S not on the programming language.

File Structure operations :—

1. Opening file :—

Opening a file makes it ready for use by the program. Means, to associative a logical prog., file with a physical system file is opening file is compulsory.

There are two options for opening file.

1. Opening a existing file

(ii) Creating a new file.

→ Opening a file makes it ready for use by the program to do further operations.

The C++, { open() (class level access)
fopen() (for file level access).

There are used to open a file.

Open() → Must have following arguments.

* The name of the physical file.
* Access mode.
* Protection mode [For new file creation].

Syntax / Prototype :—

opening existing file → int open(const char *Filename, int Access);

creating new file → int open(const char *Filename, int Access, int Protection);

Example :—

```
int Input;
Input = open("Sravani.txt", O_RDONLY);
```

```
File *Input;
Input = fopen("Sravani.txt", "r");
```

Access mode :-

- O_RDONLY → Read
- O_WRONLY → Write
- O_RDWR → Read & Write.
- O_CREATE

Protection mode :-

- DOS :- DOS support's protection modes.
  - ↳ Read only
  - ↳ Hidden
  - ↳ System (For allows all the system users to do read & write operations)
- UNIX :-
  - ↳ Readable
  - ↳ Writable
  - ↳ Executable.
- Window :-
  - ↳ Readable
  - ↳ Writable
  - ↳ Executable
  - ↳ Modifiable

2. Closing file :-

close : It is a file operation to this associate a logical program file from a physical system file.

* closing a file frees system resource for reuse.
* Data may not be actually written to return

the physical file until logical file is closed

* A program should close a file when it is no longer needed.

* close( )

* fclose( )

Example :-

```
int Input
Input = open("Sravani.txt", O_RDONLY)
int close(Input);
```

* The value return by the close is 0 if the close is success.

* The value return by the close is -1 if the close is unsucess.

3, Reading & Writing :-

Read :- To transfer data from a file to program variable(s).

Write :- To transfer data to a file from program variable(s) and constants.

* These two operations are performed on the logical file by the program.

* For read one (or) more variables must be supplied to the read() function. To receive

the data from the file.

* For write one/more variables (or) constants mut be supplied to the write() function. To provide data for the file.

Read():

In C++, read() → used to read a data from the file to handle access. read() has following arguments.,

* Source name of file to read from.
* The address of the memory block in which the data will be stored.
* The no. of bytes to be read [byte count]

* The value return by the read() is the no. of bytes read.

Prototype:

```
int read(int Input, void *Buffer,
         unsigned length);
```

Example:

```
int Input
Input = open("Sravani.txt", O_RDONLY);
int read(Input, &c, 1);
```

Write():

In C++., write() → is used to write data to a file.

The write() has following arguments

* Logical file name used for sending data.
* The address of memory block from which the data will be written.
* The no. of bytes to be write.

→ The value returned by the write() is the no. of bytes written.

Prototype :-

```
int write(int Handle, void *Buffer,
          Unsigned length);
```

Example :-

```
int Input
Input = open("Jaravani.txt", O_RDONLY);
int write(Input, &c, 1);
```

Program by using above Syntax's :-

```
#include<stdio.h>
main()
{
char ch;
FILE *file;
char filename[20];
printf("Enter the name of file");
gets(filename);
file=fopen(filename, "r");
```

```
while(fread(&ch, 1, 1, file) != 0)
    fwrite(&ch, 1, 1, stdout);
    fclose(file);
}
```

## Detecting End of file :—

End of file:- A physical location jeast beyound the last data in a file.

→ The acronym for End of file is EOF.

→ Then form a file reaches EOF, no more data can be read.

→ In C++., eof() is used to detect end-of-file with ⊕ handle level access.

→ In C++., feof() is used to detect end-of-file with FILE level access.

Prototype :—

```
int feof(FILE *stream);
```

Eg:-

```
if(feof(Input))
    Cout << " End of file \n";
```

## Seeking (attempt) :—

The action of moving directly to a certain position in a file is called seeking.

Seek:- To move to a specified location in a file.

byte offset:—

The distance, measured in bytes, from the beginning.

Seeking can be specified from one of the three reference points.

1) The beginning of the file.
2) The end of the file.
3) The current file in pointer position.

The C++ lseek() → used to seek with handle level access.

fseek() → used to seek with FILE level access.

seekg() → seek with class level access for read (get)

seekp() → seek with class level access for write (put).