Our programs run in RAM - Primary memory. To store data permanently we need to store it in secondary memory in form of files. In fact, this was the primitive way of storing data. In older applications before modern database management system came into picture.

Read speed : 270 mb/s
write speed : 70 mb/s

Two basic operations by which we can modify our files are:

1) Read SD card —copy→ computer 2) Write laptop —copy→ SD card.

but we also have to learn how to open and close files.

What is a file? A named collection of data, typically stored in secondary storage. Ex: non volatile data storage.

How is a file stored? stored as a sequence of bytes, logically contiguous. Every file is characterised with starting of a file, sequence of bytes and end of stream (or end of file).

It allows only sequential access of data by pointer performing.

Meta data (info abt the file) before the stream of actual data can be maintained to have a knowledge about data stored in it.

How is a file stored?

The last byte of a file contains the end-of-file character (EOF, with ascii code 1A(Hex)).

* All file handling utilities are present in <stdio.h> header file.

Opening a file

used to open a file in a specific opening mode to perform operations such as reading, writing etc. It returns a file pointer that can be used to interact with the file.

```
int main() {
    FILE *ptr = fopen("test.txt", "r");
    return 0;
}
```

syntax: fopen(<filename or path>, <opening mode>);

FILE *ptr = fopen(.....)
↳ data type

* fopen function returns a NULL pointer if file cannot be opened.
* opening an existing file causes its old contents to be discarded, while appending preserves previous data.

Reading from a file

To read line by line we can use fgets(<character array>, <maximum-byte-size>, <file-pointer>)
destination source

```
#include<stdio.h>
int main(){
    FILE* ptr = fopen("raghav.txt","r");
    char str[100];
    while(fgets(str,100,ptr)!=NULL)
        printf("%s",str);
}
```

- prints the whole file.

File Mode	Description
"r"	Opens a file for reading
"w"	Creates a file for writing (overwrite, if it contains data)
"w+"	Creates a file for reading and writing (overwrite, if it contains data)
"a"	Opens a file for appending - writing on the end of the file
"rb"	Read a binary file (read as bytes)
"wb"	Write into a binary file (overwrite, if it contains data)

Creating a file

```
#include <stdio.h>
int main() {
FILE* ptr= fopen("Pw.txt", "w");
}
```

creates a file

creates (if not present) and write data in it.

Writing to a file

To write line by line we can use `fputs(<character-array>,<file-pointer>);` or `fprintf(ptr, character array);`

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
FILE *ptr=fopen("newFile.txt","w");
char str[100]="HELLO MY NAME IS SARTHAK SETHI\n I AM 18 YEARS OLD \n I LIVE IN AGRA";
fputs(str,ptr);
return 0;
}
```

→ `fclose(ptr);`

Some more examples

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
FILE *ptr=fopen("file1.txt","w");
if(ptr==NULL)
{
    printf("EROR IN FILE CREATION!\n");
    exit(0);// quit the function
}
else{
    fputs("THIS IS A TEST FILE!\n",ptr); // fprintf(ptr,"THIS IS A TEST FILE!\n");
    printf("FILE CREATED SUCESSFULLY!\n");
    fclose(ptr);
}
return 0;
}
```

Reading from a file

Use this

`fgetc(FILE* ptr);` → pointer to a file.

It returns the next character in the stream ptr as an unsigned char (converted to int). and it returns EOF if end of file or error occurs.

`fscanf(FILE *ptr, char *format, var to store data);`

The `fscanf()` function in C is used for formatted input from a file. It reads data from the specified file stream and interprets it according to the format specifiers provided, similar to how `scanf()` works for standard input.

```c
#include <stdio.h>

int main() {
    FILE *file;
    char name[50];
    int age;
    float salary;

    // Open the file in read mode
    file = fopen("newFile.txt", "r");

    if (file == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Read data from the file
    while (fscanf(file, "%s %d %f", name, &age, &salary) != EOF) {
        printf("Name: %s, Age: %d, Salary: %.2f\n", name, age, salary);
    }

    // Close the file
    fclose(file);
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
FILE *ptr;
ptr=fopen("newFile.txt","r");
char ch;
while((ch=fgetc(ptr))!=EOF)
{
    printf("%c",ch);
}
return 0;
}
```

`fgets(char *str, int n, FILE *stream);`

destination ↑ number of characters → pointer to the file.

The `fgets()` function in C is used to read a string from a file or standard input, including whitespace characters, up to a specified limit or until a newline (`\n`) is encountered. It's a safer alternative to `gets()` as it prevents buffer overflows.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
FILE *ptr=fopen("newFile.txt","r");
char str[100];
while(fgets(str,100,ptr)!=NULL)
{
    printf("%s",str);
}
return 0;
}
```

int getc(FILE *ptr) equivalent to fgetc()

```c
#include <stdio.h>

int main() {
    FILE *file;
    int ch;

    // Open the file in read mode
    file = fopen("example.txt", "r");

    if (file == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Read and print each character until EOF
    while ((ch = getc(file)) != EOF) {
        putchar(ch);  // Print the character to the console
    }

    // Close the file
    fclose(file);
    return 0;
}
```

write: the file is used as an output

read: file is used as an input

Program to print the content of a file

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
char filename[100];
printf("ENTER THE NAME OF THE FILE\n");
scanf("%s",filename);
FILE *ptr=fopen(filename,"r");
printf("THE CONTENTS OF THE FILE ARE:\n");
char ch;
while((ch=fgetc(ptr))!=EOF)
{
    printf("%c",ch);
}
return 0;
}
```

Program to write some text reading from the keyboard and writing them into a file and then print the content from the file on the screen.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main()
{
    char filename[50];
    char content[500];
    printf("ENTER THE NAME OF THE FILE \n");
    scanf("%s", filename);
    FILE *ptr = fopen(filename, "w");
    printf("ENTER THE CONTENTS THAT YOU WANT TO PUT IN THE FILE\n");
    getchar();
    scanf("%[^\n]s", content);
    fputs(content, ptr);
    fclose(ptr);        - closing the file
    printf("THE CONTENTS OF THE FILE ARE:\n");
    char ch;
    FILE *nptr=fopen(filename,"r");   - reopening the file
    while ((ch = fgetc(nptr)) != EOF)
    {
        printf("%c", ch);
    }
    fclose(nptr);     - again closing the file
    return 0;
}
```