File Handling in C

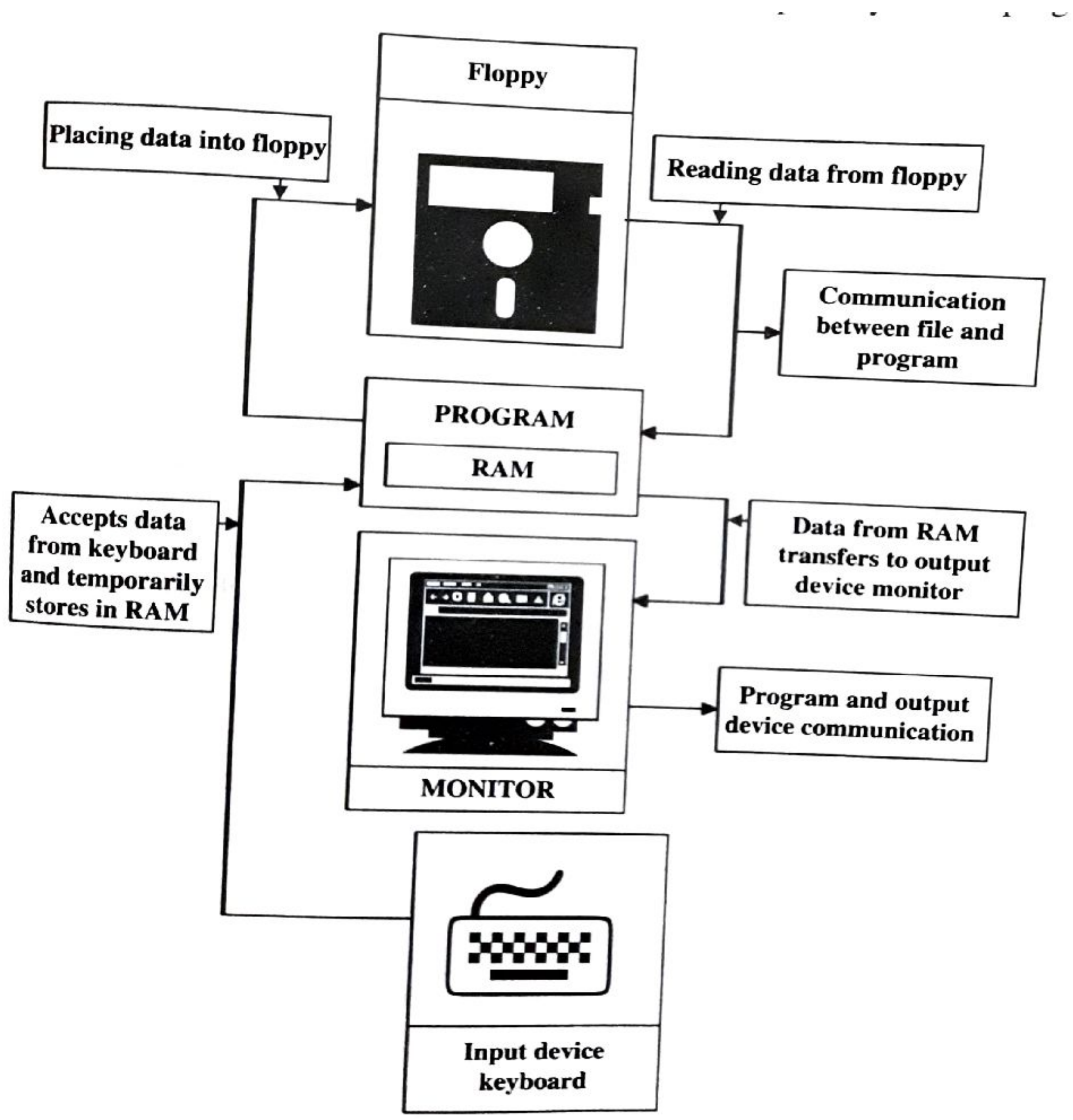
Introduction of a file

- A file is nothing but collection of records.
- Record is group of related data items.
- All data items related to students, employees, customers etc. is nothing but record.
- Files are stored permanently on to the disk and one can access them for further processing if needed.

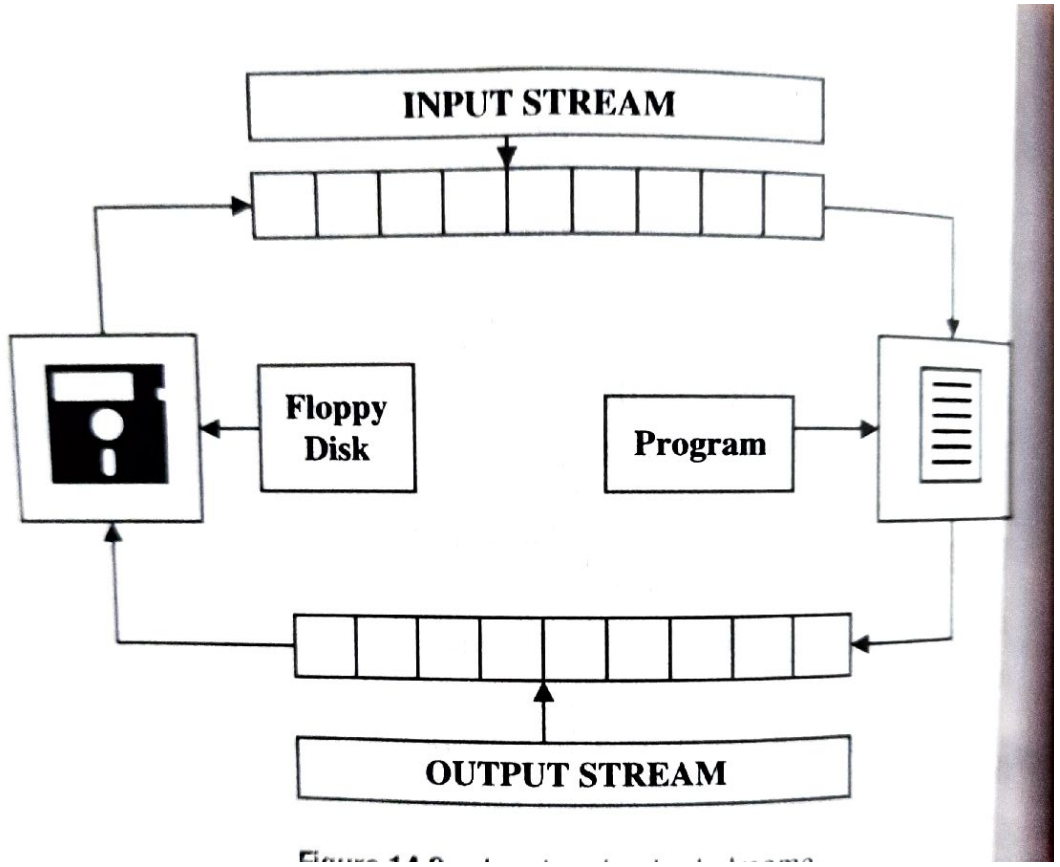
Definition of file

- A file can be considered as a stream of characters.
- A file can be set of records that can be he accessed through the set of library functions.
- The data can be stored in secondary storage devices hey searches floppy or hard disk
- File is stored as sequence of bytes logically contiguous(may not be physically contiguous on disk)

Communication between program file and output device


Floppy
Placing data into floppy
Reading data from floppy
Communication between file and program
PROGRAM
RAM
Accepts data from keyboard and temporarily stores in RAM
Data from RAM transfers to output device monitor
Program and output device communication
MONITOR
Input device keyboard

Input and output streams

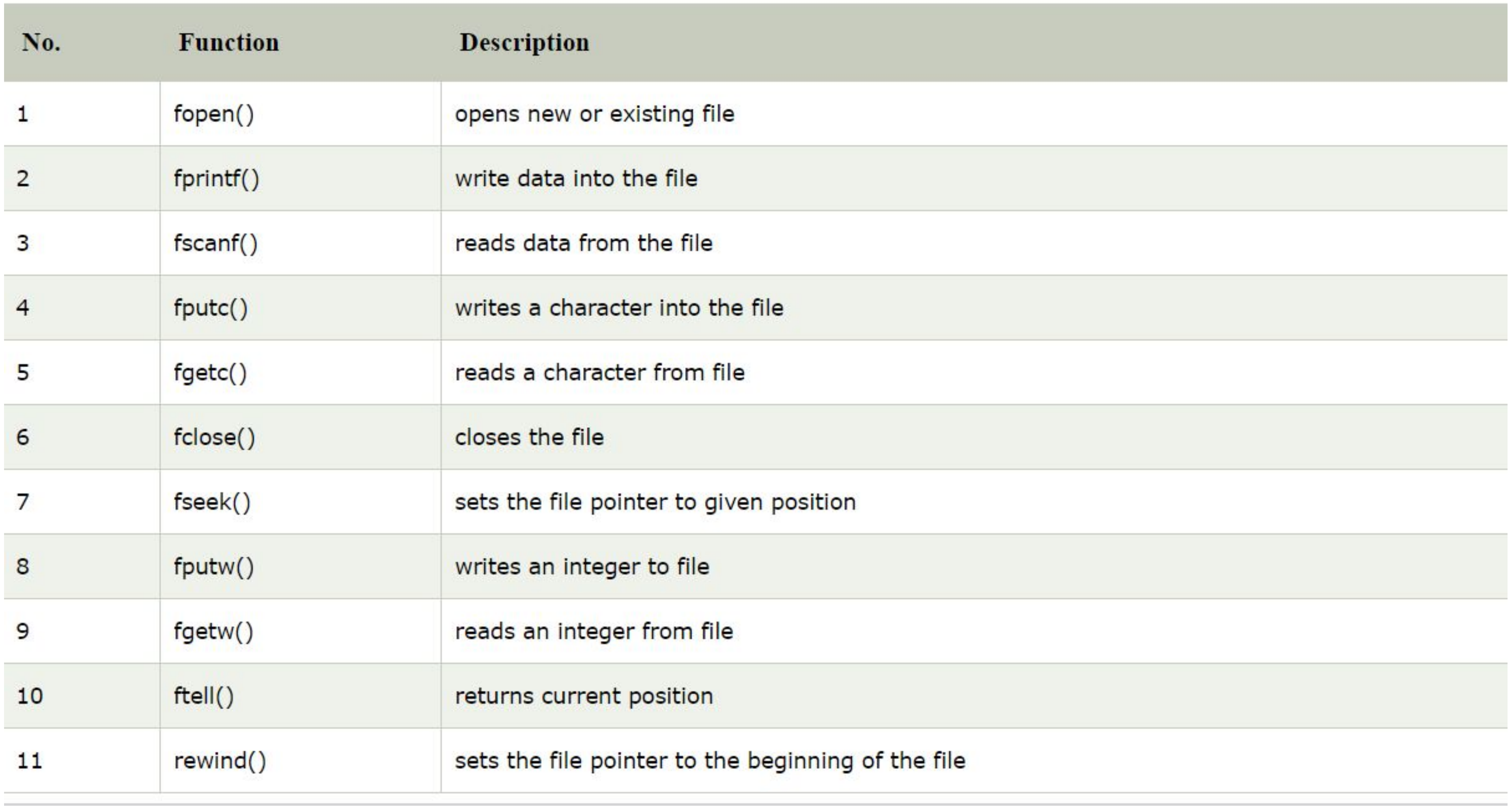


File types

- There are 2 types of files

1. sequential file
2. random access file

Functions for file handling



No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

File opening modes in C

- **“r” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
- **“w” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- **“a” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- **“r+” –** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- **“w+” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
- **“a+” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

Example

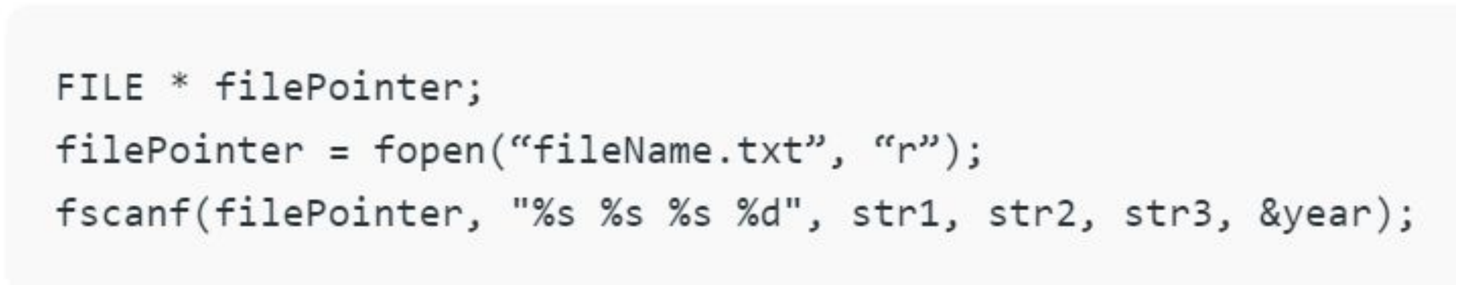
```
FILE *filePointer;
```

So, the file can be opened as

```
filePointer = fopen("fileName.txt", "w")
```

Reading from file

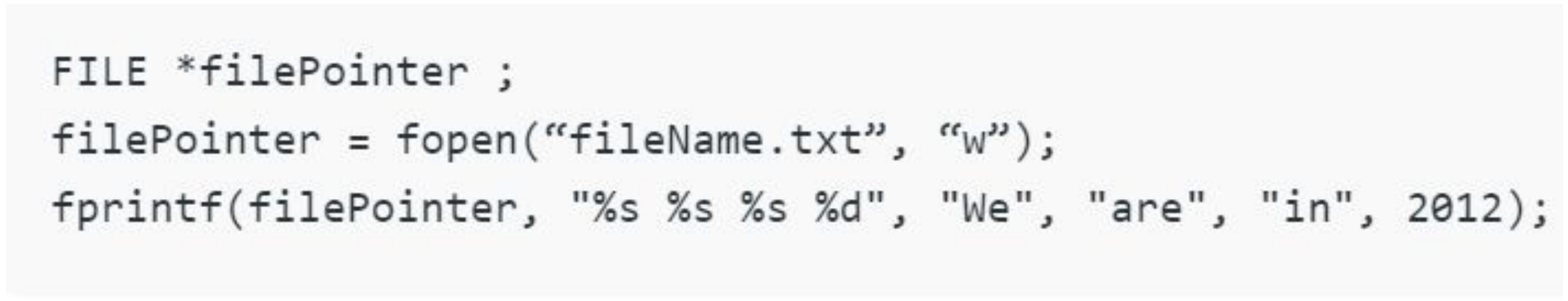
- The file read operations can be performed using functions fscanf or fgets.
- Both the functions performed the same operations as that of scanf and gets but with an additional parameter, the file pointer. So, it depends on you if you want to read the file line by line or character by character.
- And the code snippet for reading a file is as:



```
FILE * filePointer;
filePointer = fopen(“fileName.txt”, “r”);
fscanf(filePointer, "%s %s %s %d", str1, str2, str3, &year);
```

Writing a file

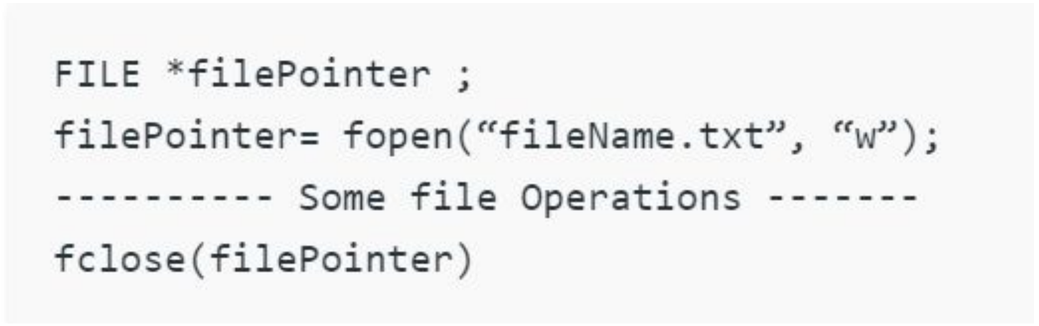
- The file write operations can be perfomed by the functions fprintf and fputs with similarities to read operations.
- The snippet for writing to a file is as :



```
FILE *filePointer ;
filePointer = fopen(“fileName.txt”, “w”);
fprintf(filePointer, "%s %s %s %d", "We", "are", "in", 2012);
```

Closing a file

- After every successful fie operations, you must always close a file. For closing a file, you have to use fclose function.
- The snippet for closing a file is given as :



```
FILE *filePointer ;
filePointer= fopen(“fileName.txt”, “w”);
---------- Some file Operations -------
fclose(filePointer)
```

Example of Writing on File

```
//Program for  writing on file
# include <stdio.h>
#include<string.h>
int main( )
{
    FILE *filePointer ; // Declare the file pointer
    char data[50];
printf("enter data to be written on file\n");
gets(data);
   filePointer = fopen("file1.txt", "w") ;
 if ( filePointer == NULL )
   {
       printf( "file1.txt file failed to open." ) ;
   }
   else
   {
       if ( strlen (  data ) > 0 )
       {
           //fputs(data, filePointer) ;// writing in the file using fputs()
           fprintf(filePointer,"%s",data) ;
       }
       fclose(filePointer) ; // Closing the file using fclose()
       printf("Data successfully written in file \n");
       printf("The file is now closed.") ;
   }
   return 0;
}
```

//Program for Reading from file

```
# include <stdio.h>
#include<string.h>
int main( )
{
   FILE *filePointer ; // Declare the file pointer
   char data[50];
   filePointer = fopen("file1.txt", "r") ;
     if ( filePointer == NULL )
   {
      printf( "file1.txt file failed to open." ) ;
   }
   else
   {    fscanf(filePointer,"%s",data);
     //while( fgets ( data, 50, filePointer ) != NULL )
      //{
         printf( "Data present on file is %s\n" , data ) ; // Print the
dataToBeRead
      //}
      fclose(filePointer) ; // Closing the file using fclose()
      printf("Data successfully read from file \n");
      printf("The file is now closed.") ;
   }
   return 0;
}
```

fgetc() and fputc()

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   FILE *fp;   /* file pointer*/
   char fName[20];
   printf("\nEnter file name to create :");
   scanf("%s",fName);
   fp=fopen(fName,"w");/*creating (open) a file*/
   if(fp==NULL)
   {
      printf("File  not created!!!");
      exit(0); /*exit from program*/
   }

   printf("%s created successfully.",fName);
   fputc('A',fp);
   fputs("India and England are playing cricket",fp);

   printf("\nData written successfully.");
   fclose(fp);

   /*again open file to read data*/
   fp=fopen(fName,"r");
   if(fp==NULL)
   {
      printf("\nCan't open file!!!");
      exit(0);
   }

   printf("\n Contents of file is :\n");
   char ch;
   do
   {
       ch=fgetc(fp);
       printf("%c",ch);
       }while(ch!=EOF);
   fclose(fp);
   return 0;
}
```

Copy content of a file into another file

```
#include <stdio.h>
#include <stdlib.h>
int main()
{       FILE *fptr1, *fptr2;
        char  c;
fptr1 = fopen("file1.txt", "r"); // Open one file for reading
fptr2 = fopen("file_copy.txt", "w"); // Open another file for writing
        c = fgetc(fptr1); // Read contents from file
while (c != EOF)
        {               fputc(c, fptr2);
                c = fgetc(fptr1);
        }
        printf("\nContents copied ");
        fclose(fptr1);   fclose(fptr2);
        return 0;
}
```

fseek()

- fseek() is used to move file pointer associated with a given file to a specific position.
- Syntax:

*int fseek(FILE *pointer, long int offset, int position)*

pointer: pointer to a FILE object that identifies the stream.

offset: number of bytes to offset from position

position: position from where offset is added.

- Returns zero if successful, or else it returns a non-zero value

position defines the point with respect to which the file pointer needs to be moved. It has three values:
SEEK_END : It denotes end of the file.
SEEK_SET : It denotes starting of the file.
SEEK_CUR : It denotes file pointer’s current position.

Example of fssek() and ftell()

```
int main()
{
    FILE *fp;
    fp = fopen("test.txt", "r");
fseek(fp, 0, SEEK_END); // Moving pointer to end
printf("%ld", ftell(fp));   // Printing position of pointer
   return 0;
}
```

Deleting a file

- The remove function in C/C++ can be used to delete a file. The function returns 0 if files is deleted successfully, other returns a non-zero value.
- Example:

```
int main()
{
if (remove(“file1.txt") == 0)
   printf("Deleted successfully");
else
   printf("Unable to delete the file");
return 0;
 }
```