File Management in C

File: A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a ready made structure.

C provides a number of functions that helps to perform basic file operations. Following are the functions,

Function	Description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

C File Operation

1)Creating a new file
2)Opening an existing file
3)Reading data from file
4)Writing data in file
5)Closing a file

Opening and Creating a File

The fopen() function is used to create a new file or to open an existing file.

General Syntax :

*fp = FILE *fopen(const char **filename*, const char **mode*);

Here filename is the name of the file to be opened and mode specifies the purpose of opening the file. Mode can be of following types,

*fp is the FILE pointer (FILE *fp), which will hold the reference to the opened(or created) file.

Mode	Description
r	Opens an existing text file for reading purpose.
w	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
a	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.

r+	Opens a text file for both reading and writing.
w+	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
a+	Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

Closing a File

To close a file, use the fclose( ) function. The prototype of this function is −

int fclose(FILE *fp);

The fclose(-) function returns zero on success, or EOF if there is an error in closing the file. This function actually flushes any data still pending in the buffer to the file, closes the file, and releases any memory used for the file. The EOF is a constant defined in the header file stdio.h.

There are various functions provided by C standard library to read and write a file, character by character, or in the form of a fixed length string.

Reading and Writing File

```
#include<stdio.h>
#include<conio.h>
struct emp
{
  char name[10];
  int age;
};

void main()
{
  struct emp e;
  FILE *p,*q;
  p = fopen("one.txt", "a");
  q = fopen("one.txt", "r");
  printf("Enter Name and Age");
  scanf("%s %d", e.name, &e.age);
  fprintf(p,"%s %d", e.name, e.age);
  fclose(p);
  do
  {
    fscanf(q,"%s %d", e.name, e.age);
    printf("%s %d", e.name, e.age);
  }
  while( !feof(q) );
  getch();
}
```

In this program, there are two FILE pointers and both are refering to the same file but in different modes. fprintf() function directly writes into the file, while fscanf() reads from the file, which can then be printed on console usinf standard printf() function.

Difference between Append and Write Mode

Write (w) mode and Append (a) mode, while opening a file are almost the same. Both are used to write in a file. In both the modes, new file is created if it doesn't exists already.

The only difference they have is, when you open a file in the write mode, the file is reset, resulting in deletion of any data already present in the file. While in append mode this will not happen. Append mode is used to append or add data to the existing data of file(if any). Hence, when you open a file in Append(a) mode, the cursor is positioned at the end of the present data in the file.

Write a Program to Create a File & Write Data in it

```
#include<stdio.h>
void main()
{
   FILE *fptr;
   char name[20];
   int age;
   float salary;

   /*  open for writing */
   fptr = fopen("emp.txt", "w");

   if (fptr == NULL)
   {
      printf("File does not exists \n");
      return;
   }
   printf("Enter the name \n");
   scanf("%s", name);
   fprintf(fptr, "Name  = %s\n", name);

   printf("Enter the age\n");
   scanf("%d", &age);
   fprintf(fptr, "Age  = %d\n", age);

   printf("Enter the salary\n");
   scanf("%f", &salary);
   fprintf(fptr, "Salary  = %.2f\n", salary);

   fclose(fptr);
}
```

fseek()

int fseek(FILE *stream, long offset, int whence);

The fseek() function is used to set the file position indicator for the stream to a new position. This function accepts three arguments. The first argument is the FILE stream pointer returned by the fopen() function. The second argument ‘offset’ tells the amount of bytes to seek. The third argument ‘whence’ tells from where the seek of ‘offset’ number of bytes is to be done. The available values for whence are SEEK_SET, SEEK_CUR, or SEEK_END. These three values (in order) depict the start of the file, the current position and the end of the file.

Upon success, this function returns 0, otherwise it returns -1.

Example:

```
#include <stdio.h>

struct threeNum
{
  int n1, n2, n3;
};

int main()
{
  int n;
  struct threeNum num;
  FILE *fptr;

  if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
      printf("Error! opening file");

      // Program exits if the file pointer returns NULL.
      exit(1);
  }

  // Moves the cursor to the end of the file
  fseek(fptr, sizeof(struct threeNum), SEEK_END);

  for(n = 1; n < 5; ++n)
  {
    fread(&num, sizeof(struct threeNum), 1, *fptr);
    printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2, num.n3);
  }
  fclose(fptr);

  return 0;
}
```