File Handling in C with Examples (fopen, fread, fwrite, fseek)

As with any OS, file handling is a core concept in Linux. Any system programmer would learn it as one of his/her initial programming assignments. This aspect of programming involves system files.

Through file handling, one can perform operations like create, modify, delete etc on system files. Here in this article I try to bring in the very basic of file handling. Hope this article will clear the top layer of this multilayer aspect.

File handling functions

In this article, we will cover the following functions that are popularly used in file handling :

fopen()

FILE \*fopen(const char \*path, const char \*mode);

The fopen() function is used to open a file and associates an I/O stream with it. This function takes two arguments. The first argument is a pointer to a string containing name of the file to be opened while the second argument is the mode in which the file is to be opened. The mode can be :

* ‘r’    :  Open text file for reading. The stream is positioned at the beginning of the file.
* ‘r+’ :  Open for reading and writing. The stream is positioned at the beginning of the file.
* ‘w’   :  Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
* ‘w+’ : Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
* ‘a’    : Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
* ‘a+’ : Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

The fopen() function returns a FILE stream pointer on success while it returns NULL in case of a failure.

fread() and fwrite()

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

The functions fread/fwrite are used for reading/writing data from/to the file opened by fopen function. These functions accept three arguments. The first argument is a pointer to buffer used for reading/writing the data. The data read/written is in the form of ‘nmemb’ elements each ‘size’ bytes long.

In case of success, fread/fwrite return the number of bytes actually read/written from/to the stream opened by fopen function. In case of failure, a lesser number of byes (then requested to read/write) is returned.

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.

fclose()

int fclose(FILE \*fp);

The fclose() function first flushes the stream opened by fopen() and then closes the underlying descriptor. Upon successful completion this function returns 0 else end of file (eof) is returned. In case of failure, if the stream is accessed further then the behavior remains undefined.

The code

#include<stdio.h>

#include<string.h>

#define SIZE 1

#define NUMELEM 5

int main(void)

{

FILE\* fd = NULL;

char buff[100];

memset(buff,0,sizeof(buff));

fd = fopen("test.txt","rw+");

if(NULL == fd)

{

printf("\n fopen() Error!!!\n");

return 1;

}

printf("\n File opened successfully through fopen()\n");

if(SIZE\*NUMELEM != fread(buff,SIZE,NUMELEM,fd))

{

printf("\n fread() failed\n");

return 1;

}

printf("\n Some bytes successfully read through fread()\n");

printf("\n The bytes read are [%s]\n",buff);

if(0 != fseek(fd,11,SEEK\_CUR))

{

printf("\n fseek() failed\n");

return 1;

}

printf("\n fseek() successful\n");

if(SIZE\*NUMELEM != fwrite(buff,SIZE,strlen(buff),fd))

{

printf("\n fwrite() failed\n");

return 1;

}

printf("\n fwrite() successful, data written to text file\n");

fclose(fd);

printf("\n File stream closed through fclose()\n");

return 0;

}

The code above assumes that you have a test file “test.txt” placed in the same location from where this executable will be run.

Initially the content in file is :

$ cat test.txt

hello everybody

Now, run the code :

$ ./fileHandling

File opened successfully through fopen()

Some bytes successfully read through fread()

The bytes read are [hello]

fseek() successful

fwrite() successful, data written to text file

File stream closed through fclose()

Again check the contents of the file test.txt. As you see below, the content of the file was modified.

$ cat test.txt

hello everybody

hello

Example of file functions like fread(),

fwrite(), fopen(), fclose() and fseek()

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#define MAX 50**

**typedef struct**

**{**

**char name[10];**

**int key;**

**} file\_record;**

**/\* this function adds the relatiuve addres to the index for a key \*/**

**void create\_index(long index[], int key, long rel\_add )**

**{**

**index[key] = rel\_add;**

**}**

**/\* this function writes a record to the file \*/**

**void write\_rec(FILE \*fp, file\_record rec)**

**{**

**fwrite(&rec,sizeof(rec),1,fp);**

**}**

**void main()**

**{**

**long rel\_add;**

**int key;**

**file\_record frec;**

**long index[MAX];/\* an index list\*/**

**int n,i;**

**FILE \*recfile=NULL,\*ifile=NULL;**

**/\* this initializes the index list to all –1 \*/**

**for(i=0; i< MAX; i++)**

**index[i]= (-1);**

**recfile=fopen(\"mfile\",\"w\");**

**if(recfile == NULL)**

**{**

**printf(\"Error in openeing file mfile\\n\");**

**exit(0);**

**}**

**rel\_add = 0 ;**

**do**

**{**

**printf(\" Enter the data vlue and the key of the record to be added to file mfile\\n\");**

**scanf(\"%s %d\",frec.name,&frec.key);**

**while(index[frec.key] != (-1))**

**{**

**printf(\" A record with this key value already exist in a file enter record key**

**value\\n\");**

**scanf(\"%s %d\",frec.name,&frec.key);**

**}**

**create\_index(index,frec.key,rel\_add);**

**write\_rec(recfile,frec);**

**rel\_add = ftell(recfile); /\* this sets the relative address for the next record to be**

**the value of current file position pointer in bytes from**

**the beginning of the file \*/**

**printf(\"Enter 1 to continue adding records to the file\\n\");**

**scanf(\"%d\",&n);**

**}while(n == 1);**

**ifile=fopen(\"index\_file\",\"w\");**

**if(ifile == NULL)**

**{**

**printf(\"Error in openeing file index\_file\\n\");**

**exit(0);**

**}**

**fwrite(index,sizeof(index),1,ifile);/\*writes the complete index into the index\_file \*/**

**fclose(recfile);**

**fclose(ifile);**

**printf(\"Enter 1 if you want to retrieve a record\\n\");**

**scanf(\"%d\",&n);**

**if( n == 1)**

**{**

**ifile=fopen(\"index\_file\",\"r\");**

**if(ifile == NULL)**

**{**

**printf(\"Error in openeing file index\_file\\n\");**

**exit(0);**

**}**

**fread(index,sizeof(index),1,ifile);/\* reads the complete index into the index list**

**from the index\_file\*/**

**fclose(ifile);**

**recfile=fopen(\"mfile\",\"r\");**

**if(recfile == NULL)**

**{printf(\"Error in openeing file mfile\\n\");**

**exit(0);**

**}**

**}**

**printf(\"THE CONTENTS OF FILE IS \\n\");**

**while( (fread(&frec,sizeof(frec),1,recfile)) != 0)**

**printf(\"%s %d\\n\",frec.name,frec.key);**

**do**

**{**

**printf(\"Enter the key of the record to be retrieved\\n\");**

**scanf(\"%d\",&key);**

**rel\_add = index[key]; /\*gets the relative address of the record from index list \*/**

**if( (fseek(recfile,rel\_add,SEEK\_SET))!= 0)**

**{**

**printf(\"Error\\n\");**

**exit(0);**

**}**

**fread(&frec,sizeof(frec),1,recfile);**

**printf(\"The data value of the retrieved record is %s\\n\",frec.name);**

**printf(\"Enter 1 if you want to retrieve a record\\n\");**

**scanf(\"%d\",&n);**

**} while(n == 1);**

**fclose(recfile);**

**}**