C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

Jagatpur P. G. College, Varanasi

Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi

Email- dharmrajsingh67@yahoo.com

Outline

unit 6 : File handling

- File I/O
- Create a File
- Opening Files
- Closing a File:
- Reading from a File
- Writing a File
- fscanf() and fprintf()
- rewind( )
- fseek():
- ftell()

File I/O

- A file represents a sequence of bytes, does not matter if it is a text file or binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices.

File Operations:

There are different operations that can be carried out on a file.

These are:

1. Creation of a new file
2. Opening an existing file
3. Reading from a file
4. Writing to a file
5. Moving to a specific location in a file (seeking)
6. Closing a file

Create a File

- Whenever you want to work with a file, the first step is to create a file. A file is nothing but space in a memory where data is stored.
- Syntax:

```
FILE *fp;
fp = fopen ("file_name", "mode");
```

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

Note : File is created in the same folder where you have saved your code.

You can specify the path where you want to create your file

```
#include <stdio.h>
int main() {
FILE *fp;
fp  = fopen ("D://data.txt", "w");
}
```

Opening Files: You can use the fopen( ) function to create a new file or to open an existing file, this call will initialize an object of the type FILE, which contains all the information necessary to control the stream.

Syntaxe :

```
FILE *fopen( const char * filename, const char * mode );
```

Here, filename is string literal, which you will use to name your file and access mode can have one of the following values:

Mode	Description
r	opens a text file for reading (must exist)
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
rb	opens a binary file for reading
wb	opens a binary file for writing
ab	appends to a binary file
r+	Opens a text file for reading and writing both.
w+	Opens a text file for reading and writing both. It first truncate the file to zero length if it exists otherwise create the file if it does not exist.
a+	Opens a text file for reading and writing both. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.
rb+	opens a binary file for read/write
wb+	opens a binary file for read/write
ab+	append a binary file for read/write

Closing a File:

If we want to close a file in c we can use the fclose() function. Syntax of fclose() function is as fallows

int fclose(FILE *fp);

```
/* Display contents of a file on screen. */
# include "stdio.h"
main( )
{
FILE *fp ;
char ch ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
printf ( "%c", ch ) ;
}
fclose ( fp ) ;
}
```

Reading from a File

- To read the file's contents from memory there exists a function called fgetc( ). This has been used in our program as, ch = fgetc (fp) ;
- The fgetc() function reads a character from the input file referenced by fp and returns the character that is read, which we collected in the variable ch.

```
void main()
{
FILE *fp;
int x;
 fp=fopen("first.txt","r");
if(fp==NULL)
{
 printf("File not found.");
 return 1;
}
while(1)
{
 x=fgetc(fp);
 if(feof(fp))
 break;
 printf("%c",x);
}
fclose(fp);
getch() ;
}
```

The following functions allow you to read a string from a stream:

```
char *fgets( char *buf, int n, FILE *fp );
```

The functions fgets() reads up to n - 1 characters from the input stream referenced by fp. It copies the read string into the buffer buf, appending a null character to terminate the string.

```
#include <stdio.h>
main()
{
FILE *fp;
char buff[100];
fp = fopen("test.txt", "r");
fscanf(fp, "%s", buff);
printf("1 : %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("2: %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("3: %s\n", buff );
fclose(fp);
}
```

Writing a File

- Following is the simplest function to write individual characters to a stream:

int fputc(int c, FILE *fp);

- The function fputc() writes the character value of the argument c to the output stream referenced by fp. It returns the written character written on success otherwise EOF if there is an error.
- You can use the following functions to write a null-terminated string to a stream:

int fputs(const char *s, FILE *fp);

- The function fputs() writes the string s to the output stream referenced by fp.

```
#include <stdio.h>
main()
{
FILE *fp;
fp = fopen("test1.txt", "w+");
fprintf(fp, "This is testing for fprintf...\n");
fputs("This is testing for fputs...\n", fp);
fclose(fp);
}
```

When the above code is compiled and executed, it creates a new file test.txt in and writes two lines using two different functions.

getc( ) function and putc( ) function : The getc() and putc() are exactly similar to that of fgetc() and fputc(). The putc function writes the character to the file associated with the file pointer.

The getc() function takes the file pointer as its argument, reads the character from the file and returns that character as a integer or EOF when it reaches the end of the file.

```
int putc (int ch, FILE *file)   // define putc() method
int getc (FILE *file)           // define getc() method
```

```
#include<stdio.h>
int main () {
  FILE *fp;
  int ch;
  fp = fopen("myfile.txt", "w");
  for( ch = 65 ; ch <= 90; ch++ ) {
     putc(ch, fp);
  }  fclose(fp);
fp=fopen("myfile.txt","r");
while((ch=getc(fp))!=EOF)
{
printf("%c",ch);
}
fclose(fp);
}
```

fscanf() and fprintf()

Functions fprintf() and fscanf() are same as printf() and scanf() difference is that, these functions work on files instead of standard input . These function has one more parameter which is a pointer of file. Syntax of these function are as follows-

```
fscanf(filepointer,"format specifier",&variable1,&variable2,….);
Ex: fscanf(fp,"%d%d%d",&a,&b,&c);
fprintf(filepointer,"format specifier",&variable1,&variable2……);
Ex: fprintf(fp,"%d\t%d\t%d\t",a,b,c);
```

```
#include <stdio.h>
int main(){
  FILE *fp;
  char buff[255];        //creating char array to store data of file
  fp = fopen("file.txt", "r");
  while(fscanf(fp, "%s", buff)!=EOF){
  printf("%s ", buff );
  }
  fclose(fp);
  return 0;
}
```

rewind( ) : The rewind( ) function places the pointer to the beginning of the file, irrespective of where it is present right now.

```
#include<stdio.h>
#include<conio.h>
int main(){
FILE *fp;
char c;
fp=fopen("file.txt","r");
 while((c=fgetc(fp))!=EOF){
printf("%c",c);
}
printf("\n");
rewind(fp);       //moves the file pointer at beginning of the file
while((c=fgetc(fp))!=EOF){
printf("%c",c);
}
  fclose(fp);
return 0;
}
```

fseek():

➢ The **fseek( )** The fseek() function takes three arguments, first is the file pointer, second is the offset that specifies the number of bytes to moved and third is the position from where the offset will move. It will return zero if successfully move to the specified position otherwise return nonzero value.

int fseek(FILE *fp, long offset, int pos);

There are three positions from where offset can move.

Constant Name	Constant Value	Description
SEEK_SET	0	The begining of file
SEEK_CUR	1	The current position in file
SEEK_END	2	The end of file

ftell()

ftell() in C is used to find out the current position of file pointer in the file with respect to starting of the file. Syntax of ftell() is:

```
long ftell(FILE *pointer)
```

```
#include<stdio.h>
     int main()
{
   /* Opening file in read mode */
   FILE *fp = fopen("test.txt","r");
    /* Reading first string */
   char string[20];
   fscanf(fp,"%s",string);

   /* Printing position of file pointer */
   printf("%ld", ftell(fp));
   return 0;
}
```

Exercise

1. Write a program to read a file and display contents with its line numbers.
2. Write a program to find the size of a text file without traversing it character by character.
3. Write a program to add the contents of one file at the end of another.
4. Suppose a file contains student's records with each record containing name and age of a student. Write a program to read these records and display them in sorted order by name.
5. Write a program to copy one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.
6. Write a program in C to remove a file from the disk.
7. Write a program in C to Find the Number of Lines in a Text File.
8. Write a program in C to append multiple lines at the end of a text file.

References

- Kanetkar, Yashavant P. "Let UsC Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks