CL118 Programming Fundamentals

LAB 12

FILE PROCESSING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

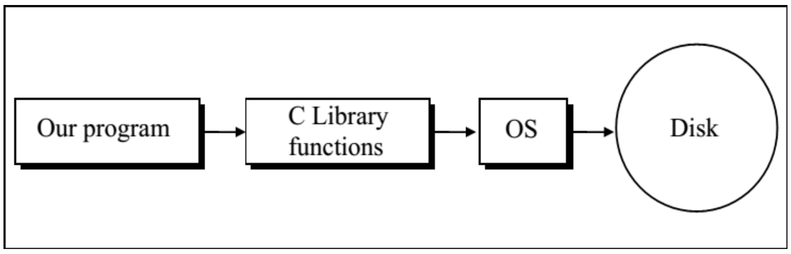
File Handling in C

Often it is not enough to just display the data on the screen. This is because if the data is large, only a limited amount of it can be stored in memory and only a limited amount of it can be displayed on the screen. It would be inappropriate to store this data in memory for one more reason. Memory is volatile and its contents would be lost once the program is terminated. So if we need the same data again it would have to be either entered through the keyboard again or would have to be regenerated programmatically.

Obviously both these operations would be tedious. At such times it becomes necessary to store the data in a manner that can be later retrieved and displayed either in part or in whole. **This medium is usually a 'file' on the disk**

How data is organized???

1. All data stored on the disk is in binary form.
2. How this binary data is stored on the disk varies from one OS to another.
3. However, this does not affect the C programmer since he has to use only the library functions written for the particular OS to be able to perform input/output.
4. It is the compiler vendor's responsibility to correctly implement these library functions by taking the help of OS.
5. Following figure illustrates the concept,



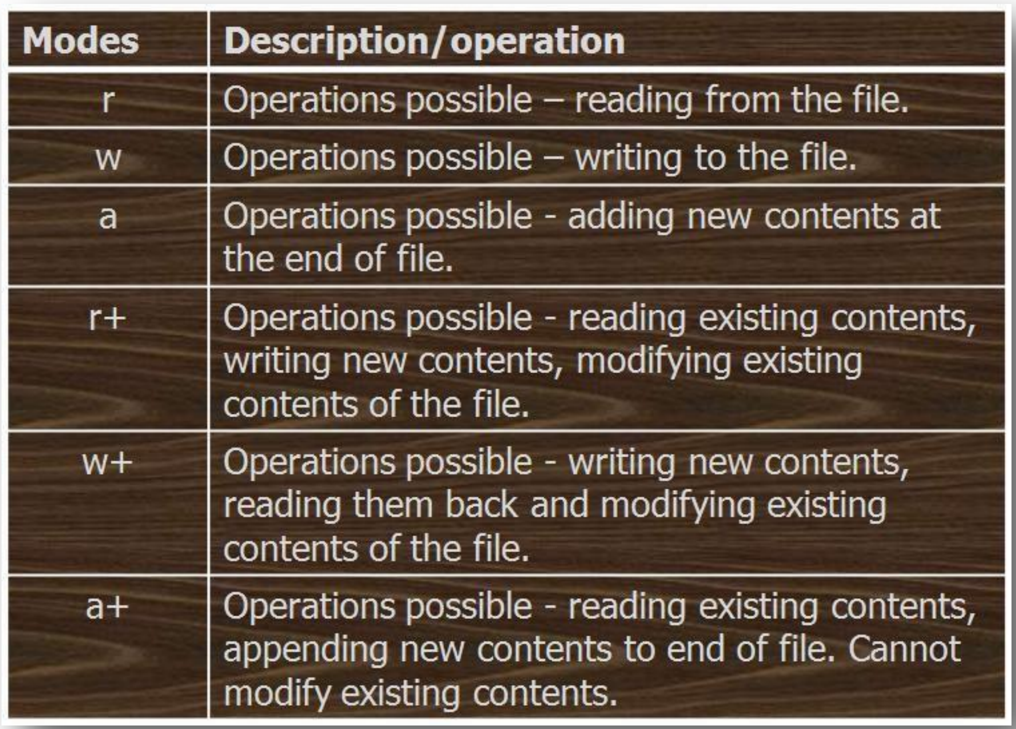
File Operations

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. Closing a file

Let us now write a program to read a file and display its contents on the screen. We will first list the program and show what it does, and then dissect it line by line.

File Modes



Modes	Description/operation
r	Operations possible – reading from the file.
w	Operations possible – writing to the file.
a	Operations possible - adding new contents at the end of file.
r+	Operations possible - reading existing contents, writing new contents, modifying existing contents of the file.
w+	Operations possible - writing new contents, reading them back and modifying existing contents of the file.
a+	Operations possible - reading existing contents, appending new contents to end of file. Cannot modify existing contents.

1. Simple program

```
//reading the content of a file.
#include<stdio.h>
int main( )
{
FILE *fp;
char ch;
fp = fopen ( "abc.txt", "r" );
while( 1 )
{
ch = fgetc ( fp );
if ( ch == EOF )
break;
printf ( "%c", ch );
}
fclose ( fp ); return 0;
}
```

Line by Line Program Illustration

1. To open the file we have called the function fopen ( ).
2. open( )performs three important tasks when you open the file in “r” mode:
    a. Firstly it searches on the disk the file to be opened.
    b. Then it loads the file from the disk into a place in memory called buffer.
    c. It sets up a character pointer that point to the first character of the buffer.

3. To be able to successfully read from file information like mode of opening, size of file, place in the file from where the next read operation would be performed, etc. has to be maintained.

Reading from the File

1. Since all this information is inter-related, all of it is gathered together by fopen() in a structure called FILE.
2. fopen() returns the address of this structure, which we have collected in the structure pointer called fp. We have declared fp as FILE *fp ;
3. The FILE structure has been defined in the header file "stdio.h"
4. Once the file has been opened for reading using fopen( ), file's contents are brought into buffer (partly or wholly) and a pointer is set up that points to the first character in the buffer. This pointer is one of the elements of the structure to which fp is pointing.

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) ;

fgetc() function

fgetc( )reads the character from the current pointer position, advances the pointer position so that it now points to the next character, and returns the character that is read, which we collected in the variable ch. Note that once the file has been opened, we no longer refer to the file by its name, but through the file pointer fp.

End of File

1. We have used the function fgetc( )within an indefinite while loop.
2. There has to be a way to break out of this while. When shall we break out... the moment we reach the end of file.
3. But what is end of file? A special character, whose ASCII value is 26, signifies end offile.
4. This character is inserted beyond the last character in the file, when it is created.
5. In our program we go on reading each character from the file till end of file is not met. As each character is read we display it on the screen. Once out of the loop, we close the file.

Trouble in opening a file

1. Crux of the matter is that it is important for any program that accesses disk files to check whether a file has been opened successfully before trying to read or write to the file.
2. If the file opening fails due to any of the several reasons mentioned above, the fopen( )function returns a value NULL.

2. Program

```
#include <stdio.h>
int main( )
{
FILE *fp;
fp = fopen ( "abc.txt", "r" );
if ( fp == NULL )
{
puts ( "cannot open file" );
exit( );
}
}
```

Closing the File

1. When we have finished reading from the file, we need to close it.
2. This is done using the function fclose( )through the statement,

   fclose (fp) ;

3. Once we close the file we can no longer read from it using getc( ) unless we reopen the file.
4. Note that to close the file we don’t use the filename but the file pointer fp.
5. On closing the file the buffer associated with the file is removed from memory.

6. When we attempt to write characters into this file using fputc( ) the characters would get written to the buffer.
7. When we close this file using fclose( ) three operations would be performed:
   1. The characters in the buffer would be written to the file on the disk.
   2. At the end of file a character with ASCII value 26 would get written.
   3. The buffer would be eliminated from memory.

3. Program

```
//Count characters,tabs,spaces and newline from a file
#include<stdio.h>
int main( )
{
FILE *fp;
char ch;
int no_char=0,no_lines=0,no_spaces=0,no_tabs=0;
fp = fopen ( "abc.txt", "r" );
while( 1 )
{
ch = fgetc ( fp );
if ( ch == EOF )
break;
   if(ch==' ')
   no_spaces++;
   if(ch=='\t')
   no_tabs++;
   if(ch=='\n')
   no_lines++;
no_char++;
printf("%c", ch);
}
printf("\nTotal Characters are:%d \n",no_char);
printf("Total Lines are:%d \n",no_lines);
printf("Total Spaces are:%d \n",no_spaces);
printf("Total Tabs are:%d \n",no_tabs);
fclose(fp);
return 0;
}
```

fgetc() and fputc() Functions

1. The function fgetc( ) which reads characters from a file.
2. Its counterpart is a function called fputc( )which writes characters to a file.
3. As a practical use of these character I/O functions we can copy the contents of one file into another, as demonstrated in the program on the next slide

4. Program: Copying contents of a File to another file

```
#include<stdio.h>
int main( )
{
FILE *fs,*ft;// fs=> source file and ft => target file
char ch;

fs=fopen("abc.txt", "r");
   if ( fs == NULL )
   {
   puts ("cannot open file");
   }
ft = fopen ("xyz.txt", "w");
   if (ft == NULL)
   {
   puts("cannot open file");
   }
while( 1 )
{
ch=fgetc(fs);
if(ch==EOF)
break;
else
putc(ch,ft);
}
fclose(fs);
fclose(ft);
return 0;
}
```

Strings in Files

Reading or writing strings of characters from and to files is as easy as reading and writing individual characters.

5. Program

```
//Reading strings from the file and display on screen
#include<stdio.h>
#include<string.h>
int main( )
{
FILE *fp;
char ch[100];
fp = fopen("ijk.txt", "r");
        if ( fp == NULL )
        {
        puts ("cannot open file");
        }
printf("The lines in files are:\n");
while(fgets(ch,100,fp))
{
printf("%s",ch);
}
fclose(fp);
}
```

A little bit explanation of the program.

- The function fgets( )takes three arguments.
- The first is the address where the string is stored.
- The second is the maximum length of the string.
- The third argument, as usual, is the pointer to the structure FILE.

RANDOM NUMBER GENERATOR

rand() % 7 + 1

EXPLANATION:

- rand() returns a random number between 0 and a large number.
- % 7 gets the remainder after dividing by 7, which will be an integer from 0 to 6 inclusive.
- + 1 changes the range to 1 to 7 inclusive.

This number is generated by an algorithm that returns a sequence of apparently non-related numbers each time it is called. This algorithm uses a seed to generate the series, which should be initialized to some distinctive value using function srand.

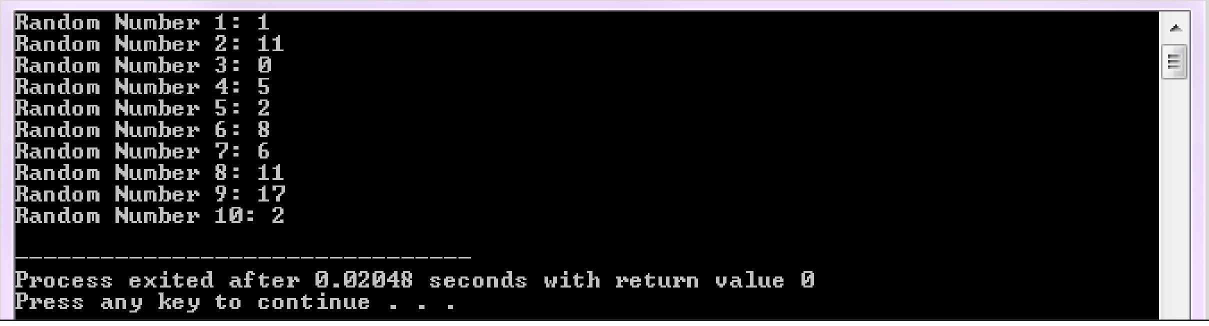
INITIALIZE RANDOM NUMBER GENERATOR

void srand (unsigned int seed);

- The random number generator is initialized using the argument passed as seed.
- For every different seed value used in a call to srand, the pseudo-random number generator can be expected to generate a different succession of results in the subsequent calls to rand.
- In order to generate random-like numbers, srand is usually initialized to some distinctive runtime value, like the value returned by function time (declared in header <time.h>). This is distinctive enough for most trivial randomization needs.

```
#include
<stdio.h>
#include <time.h>
#include<stdlib.h>

int main()
{
        int i=0;
        srand(time(NULL)); //seeds the random number generator used by the function rand.
        for(i=0;i<10;i++)
        {
                printf("Random Number %d: %d\n",i+1,rand()%20); /*generates random numbers from
                                                                  0 to 19 */
        }
        return 0;
}
```



```
Random Number 1: 1
Random Number 2: 11
Random Number 3: 0
Random Number 4: 5
Random Number 5: 2
Random Number 6: 8
Random Number 7: 6
Random Number 8: 11
Random Number 9: 17
Random Number 10: 2

--------------------------------
Process exited after 0.02048 seconds with return value 0
Press any key to continue . . .
```

LAB 12 EXERCISES

QUESTION#1

On the basis of the given scenario create a file Expenses.txt, Open the file and copy into another Expenses2.txt, Also show output on screen

A college has announced the total budget of 50,000Rs.for each game. Games are done four times in a year. Take expenses as input from user (as your own).

Calculate the average expenses for a game:

1. If the expenses greater than 80% show as" Very Expensive".
2. If the expenses are greater than 60% and less than 80%than show "Expensive "
3. If the expenses are greater than 50% and less than 60%than show "Less Expensive "
4. If the expenses are greater than 40% and less than 50% than show "Not Costly".
5. If the expenses are less than 40% than show "Best".

QUESTION#2

Write a C Program to Convert the Content of File to LowerCase.

QUESTION#3

Write a program to read a file and display contents with its line numbers.

QUESTION#4

Write a program that can be used as a math tutor for a young student. The program should display two random numbers between 10 and 50 that are to be added, such as:

```
  24
+12
____
```

The program should then wait for the student to enter the answer. If the answer is correct, a message of congratulations should be printed. If the answer is incorrect, a message should be printed showing the correct answer.

QUESTION#5

Generate a random number (RN) (within a particular range) and ask the user to guess the number (say GN). Keep guessing the number till the correct answer. The program should give direction on each guess.

Possible output:

```
Guess the number (1 to 10): 5 The secret number is higher
Guess the number (1 to 10): 8 The secret number is lower
Guess the number (1 to 10): 7 Congratulations!
```