Pemrograman Dasar

Week 15 : *IO Files*

OUTLINES

1. Background
2. Creating File
3. Opening File
4. Closing File
5. Reading & Writing Text File
6. Reading & Writing Binary File

1. Background

A file represents a sequence of bytes, regardless of it being a text file or a 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.

Why files are needed?

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all.
- However, if you have a file containing all the data, you can easily access the contents of the file using few commands in C.
- You can easily move your data from one computer to another without any changes.

- So far the operations using C program are done on a prompt / terminal which are not stored anywhere. But in software industry, most of the programs are written to store the information fetched from the program. One such way is to store the fetched information in a file.

- There are types and functions in the library stdio.h that are used for file I/O. Make sure you always include that header when you use files.

Types of Files

When dealing with files, there are two types of files you should know about:

1. **Text files**

- Text files are the normal .txt files that you can easily create using Notepad or any simple text editors.
- When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.
- They take minimum effort to maintain, are easily readable, and provide least security and takes bigger storage space.

2. **Binary files**

- Binary files are mostly the .bin files in your computer.
- Instead of storing data in plain text, they store it in the binary form (0's and 1's).
- They can hold higher amount of data, are not readable easily and provides a better security than text files.

2. Creating File

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

```
FILE *fptr;
```

2. Opening File

Opening a file is performed using the library function in the "`stdio.h`" header. 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. The prototype of this function call is as follows –

```
FILE *fopen(file, mode);
```

or

```
FILE *ptr;
ptr = fopen("E:\\cprogram\\newprogram.txt","w");
```

or

```
ptr = fopen("E:\\cprogram\\oldprogram.bin","rb");
```

Let's suppose the file “`newprogram.txt`” doesn't exist in the location “`E:\cprogram`”. The first function (1) creates a new file named `newprogram.txt` and opens it for writing as per the mode '`w`'. **The writing mode allows you to create and edit (overwrite) the contents of the file**.

Now let's suppose the second (2) binary file “`oldprogram.bin`” exists in the location “`E:\cprogram`”. The second function opens the existing file for reading in binary mode 'rb'. **The reading mode only allows you to read the file, you cannot write into the file**.

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exists, it will be created.

File Mode	Meaning of Mode	During Inexistence of file
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exists, it will be created.

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 (`w`) mode, the file is reset, resulting in deletion of any data already present in the file.

While in append (`a`) mode this will not happen. Append (`a`) 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.

3. Closing File

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

```
int fclose( FILE *ptr );
```

The `fclose()` function returns zero on success, or End Of File (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`.

What is `fflush()` in C ?

`fflush()` is typically used for output stream only. Its purpose is to clear (or flush) the output buffer and move the buffered data to console.

A buffer is an area of memory used for temporary storage when data is moved from one place to another.

ProgDas15-0.c

ProgDas15-0-1.c

ProgDas15-0-2.c

The use of `fflush()` with `stdin` (standard input streams) is an inappropriate method because it can produce something unknown. `fflush(stdin)` is an undefined behavior and will always be problematic in ANSI C (because it does not comply with standard C).

The second reason is because `fflush(stdin)` is not portable because it only runs on Windows. Your code will be different in its behavior on each platform and compiler

4. Reading & Writing Text File

Function	Description
fopen()	create a new file or open a existing file
fclose()	closes a file
READ	
fscanf()	reads a set of data from a file
fgets()	reads a line. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.
getc()	reads a character from a file
getw()	reads a integer from a file
WRITE	
fprintf()	writes a set of data to a file
fputs()	writes a string to the specified stream up to but not including the null character.
putc()	writes a character to a file
putw()	writes a integer to a file

4. Reading & Writing Text Files

For reading and writing to a text file, we use the functions `fprintf()` and `fscanf()`.

They are just the file versions of `printf()` and `scanf()`. The only difference is that, `fprint()` and `fscanf()` expects a pointer to the structure FILE.

Writing Text File: ProgDas15-1-1.c | ProgDas15-1-2.c

After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

Reading Text File: ProgDas15-2.c

This program reads the integer present in the program.txt file and prints it onto the screen.

If you succesfully created the file from Example 1, running this program will get you the integer you entered.

Other functions like `fgets(), fgetc(), fputs(), fputc(),` etc. can be used in similar way.

There are 3 special `FILE`s that are always defined for a program. They are `stdin` (standard input), `stdout` (standard output) and `stderr` (standard error).

Standard Input

Standard input is where things come from when you use `scanf()`.

```
scanf("%d", &val);
```

is equivalent to the following:

```
fscanf(stdin, "%d", &val);
```

Standard Output

Similarly, standard output is exactly where things go when you use printf().

```
printf("Value = %d\n", val);
```

is equivalent to the following:

```
fprintf(stdout, "Value = %d\n", val);
```

Remember that standard input is normally associated with the keyboard and standard output with the screen, unless redirection is used.

Standard Error

Standard error is where you should display error messages. We've already done that above:

```
fprintf(stderr, "Can't open input file in.list!\
n");
```

Standard error is normally associated with the same place as standard output; however, redirecting standard output does not redirect standard error.

5. Reading & Writing Binary File

5. Reading & Writing Binary Files

Functions `fread()` and `fwrite()` are used for reading from and writing to a file on the disk respectively in case of binary files.

Writing Binary File: ProgDas15-3.c

To write into a binary file, you need to use the function `fwrite()`. The functions takes four arguments: (1) Address of data to be written in disk, (2) Size of data to be written in disk, (3) number of such type of data and (4) pointer to the file where you want to write.

```
fwrite(address_data,size_data,numbers_data,pointer);
```

Writing Binary File: ProgDas15-4.c

Function `fread()` also take 4 arguments similar to `fwrite()` function as above.

```
fread(address_data,size_data,numbers_data,pointer);
```

6. File Control Operation

Getting data using `fseek()`

If you have many records inside a file and need to access a record at a specific position, you need to loop through all the records before it to get the record.

This will waste a lot of memory and operation time. An easier way to get to the required data can be achieved using `fseek()`.

As the name suggests, `fseek()` seeks the cursor to the given record in the file.

Syntax of `fseek()`

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

The first parameter stream is the pointer to the file. The second parameter is the position of the record to be found, and the third parameter specifies the location where the offset starts.

Whence	Meaning
SEEK_SET	Starts the offset from the beginning of the file.
SEEK_END	Starts the offset from the end of the file.
SEEK_CUR	Starts the offset from the current location of the cursor in the file.

ProgDas15-5.c | ProgDas15-5-1.c

The program will start reading the records from the file program.bin in the reverse order (last to first) and prints it.

fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

`rewind()`

A file called myfile.txt is created for reading and writing and filled with the alphabet. The file is then rewinded, read and its content is stored in a buffer.

Exercice

1. C Program to Write a Sentence to a File

The program stores a sentence entered by user in a file.

After termination of this program, you can see a text file program.txt created in the same location where this program is located.

ProgDas15-8.c

2. C Program to Read a Line From a File and Display it

The program reads text from a file and stores it in a string until enter 'newline' character is encountered.

ProgDas15-9.c

3. C Program to Display its Own Source Code as Output

ProgDas15-10.c

4. C Program to Merge Contents of Two Files Into a Third File

Let the given two files be file1.txt and file2.txt. The following are steps to merge.

1) Open file1.txt and file2.txt in read mode.

2) Open file3.txt in write mode.

3) Run a loop to one by one copy characters of file1.txt to file3.txt.

4) Run a loop to one by one copy characters of file2.txt to file3.txt.

5) Close all files.

ProgDas15-11.c

5. C Program to Copy Contents of One File to Another File

Output:

```
Enter the filename to open for reading
a.txt
Enter the filename to open for writing
b.txt
Contents copied to b.txt
```

ProgDas15-12.c

6. C program to Compare Two Files and Report Mismatches

We are given almost two identical files. But some characters are corrupted. We need to find the line number and their position where those corrupted letter exists as well as Total number of corrupted (error) letters.

Input :

```
file1.txt contains
it is
fun
file2.txt contains
it is
run
```

Output :

```
Line Number : 2          Error Position : 1
Total Errors : 1
```

Steps :

1. Open two file using File pointer in read only mode.

2. Fetch data of file in two char variable one by one until end of file.

3. If variable encounter new line then increment line number and reset position to zero.

4. If variables are not equal then increment number of error and print error line as well as error index.

ProgDas15-13.c

7. C program to delete 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.

ProgDas15-14.c

8. C Program to Count Number of Lines in A File

Output:

```
Enter file name: countLines.c
The file countLines.c has 41 lines
```

ProgDas15-19.c

9. C Program to Read Name & Marks of N Number of Students from User & Store Them in a File

ProgDas15-20.c

THANKYOU