File Management

Lecture 16

What is a File?

- A file is a collection of related data that a computers treats as a single unit.
- Computers store files to secondary storage so that the contents of files remain intact when a computer shuts down.
- When a computer reads a file, it copies the file from the storage device to memory; when it writes to a file, it transfers data from memory to the storage device.
- C uses a structure called FILE (defined in stdio.h) to store the attributes of a file.

Why do we need Files ?

In the software industry, most programs are written to store the information fetched from the program. The use of file handling is exactly what the situation calls for.

In order to understand why file handling is important, let us look at a few features of using files:

Reusability: The data stored in the file can be accessed, updated, and deleted anywhere and anytime providing high reusability.

Portability: Without losing any data, files can be transferred to another in the computer system. The risk of flawed coding is minimized with this feature.

Efficient: A large amount of input may be required for some programs. File handling allows you to easily access a part of a file using few instructions which saves a lot of time and reduces the chance of errors.

Storage Capacity: Files allow you to store a large amount of data without having to worry about storing everything simultaneously in a program.

Types of File - Text Files

A text file contains data in the form of ASCII characters and is generally used to store a stream of characters.

- Each line in a text file ends with a new line character (‘\n’).
- It can be read or written by any text editor.
- They are generally stored with .txt file extension.
- Text files can also be used to store the source code.

Types of File - Binary Files

A binary file contains data in binary form (i.e. 0’s and 1’s) instead of ASCII characters. They contain data that is stored in a similar manner to how it is stored in the main memory.

- The binary files can be created only from within a program and their contents can only be read by a program.
- More secure as they are not easily readable.
- They are generally stored with .bin file extension.

Steps in Processing a File

1. Create the stream via a pointer variable using the FILE structure:

FILE *p;

2. Open the file, associating the stream name with the file name.

3. Read or write the data.

4. Close the file.

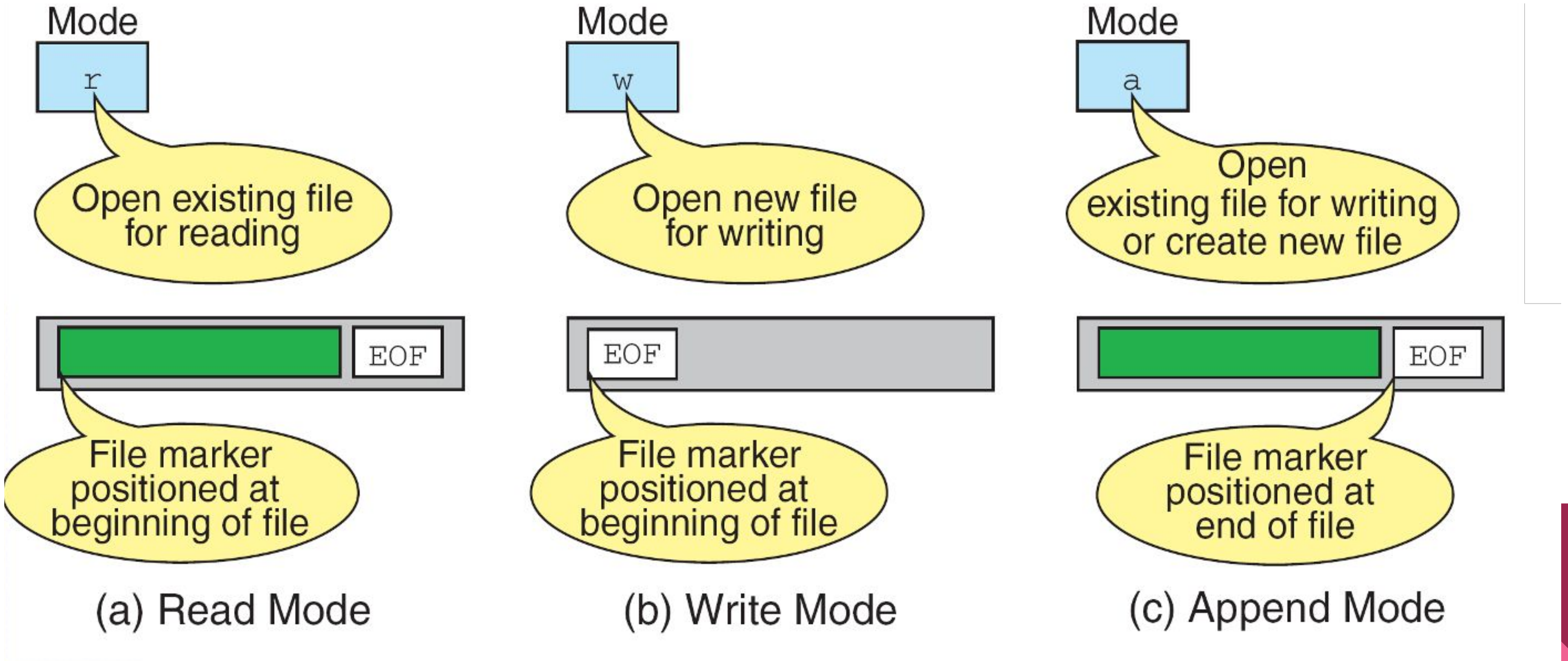
The basic file operations are

- fopen - open a file- specify how its opened (read/write) and type (binary/text)
- fclose - close an opened file
- fread - read from a file
- fwrite - write to a file
- fgetc - read from a file by character
- fputc - write to a file by character
- fseek/fsetpos - move a file pointer to somewhere in a file.
- ftell/fgetpos - tell you where the file pointer is located.

File Open Modes

Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	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.
a	Open for append.	Data is added to the end of the file. If the file does not exist, it will be created.

More on File Open Modes


Mode
r
Open existing file for reading
EOF
File marker positioned at beginning of file
(a) Read Mode
Mode
w
Open new file for writing
EOF
File marker positioned at beginning of file
(b) Write Mode
Mode
a
Open existing file for writing or create new file
EOF
File marker positioned at end of file
(c) Append Mode

Additionally

- r+ - open for reading and writing, start at beginning
- w+ - open for reading and writing (overwrite file)
- a+ - open for reading and writing (append if file exists)

Homework: Check the difference between these 3.

File Open

The file open function (fopen) serves two purposes:

- It makes the connection between the physical file and the stream.
- It creates "a program file structure to store the information" C needs to process the file.

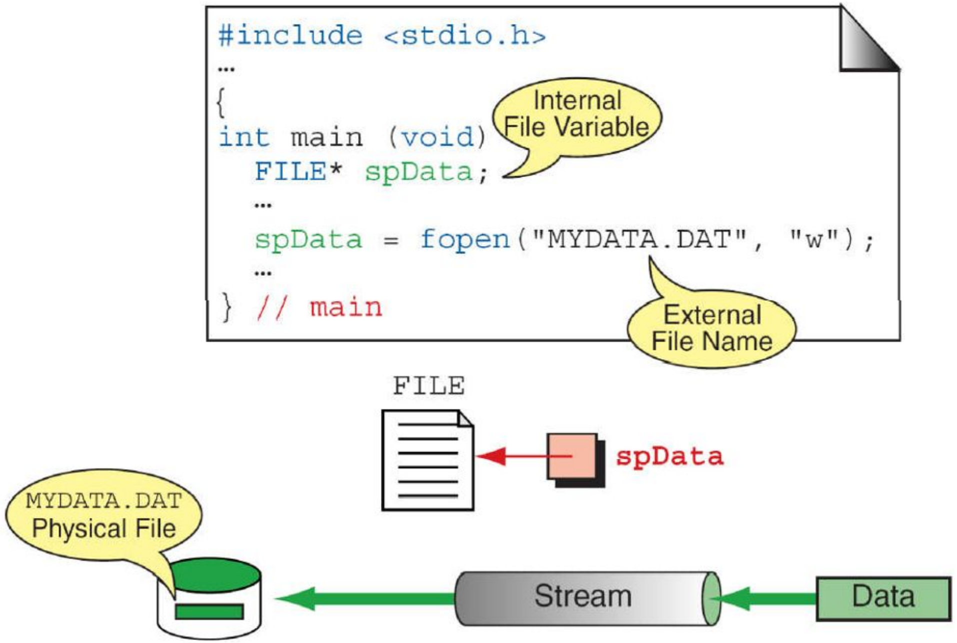
Syntax: `filepointer=fopen("filename", "mode");`

More on fopen

- The file mode tells C how the program will use the file.
- The filename indicates the system name and location for the file.
- We assign the return value of fopen to our pointer variable:

```
spData = fopen("MYFILE.TXT", "w");
```

More on fopen



```
#include <stdio.h>
...
{
int main (void)
   FILE* spData;
   ...
   spData = fopen("MYDATA.DAT", "w");
   ...
} // main
```

Closing a File

- When we finish with a mode, we need to close the file before ending the program or beginning another mode with that same file.
- To close a file, we use fclose and the pointer variable:
- fclose(spData);

fprintf()

Syntax: fprintf (fp,"string",variables);

Example:

```
int i = 12;

float x = 2.356; char ch = 's'; FILE *fp;

fp=fopen(“out.txt”,”w”);

fprintf (fp, "%d %f %c", i, x, ch);

fclose(fp);
```

fscanf()

Syntax:

fscanf (fp,"string",identifiers);

Example:

```
FILE *fp; fp=fopen(“input.txt”,”r”); int i;
fscanf (fp,“%d", &i);
printf(“%d”, i);
fclose(fp);
```

fgetc()

Syntax:

```
identifier = fgetc (file pointer);
```

Example:

```
FILE *fp;

char ch;

fp=fopen(“input.txt”,”r”);

ch = fgetc (fp);

printf(“%c”,ch);

fclose(fp);
```

fputc()

write a single character to the output file, pointed to by fp.

Example:

FILE *fp;

char ch=’a’;

fp=fopen(“input.txt”,”w+”);

fputc (ch, fp); -> Notice the order of ch and fp here.

fclose(fp);

End of File

There are a number of ways to test for the end-of-file condition. One way is to use the value returned by the fscanf function:

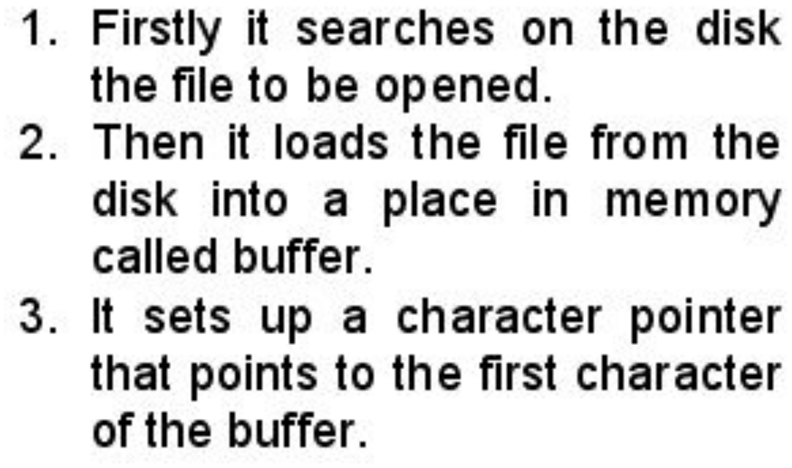
```
FILE *fptr; int istatus ;

fptr=fopen(“input.txt”,”r”);

istatus = fscanf (fptr, "%d", &var) ;

if ( istatus == feof(fptr) )

{

        printf ("End-of-file encountered.\n”) ;

}
```

Reading and Writing Files

```
#include <stdio.h> int main ( )

{

        FILE *outfile, *infile ; int b = 5, f ;

        float a = 13.72, c = 6.68, e, g ;

        outfile = fopen ("testdata", "w") ;

        fprintf (outfile, “ %f %d %f ", a, b, c) ;

        fclose (outfile) ;

        infile = fopen ("testdata", "r") ;

        fscanf (infile,"%f %d %f", &e, &f, &g) ;

         printf (“ %f %d %f \n ", a, b, c) ;

        printf (“ %f %d %f \n ", e, f, g) ; fclose (infile) ;

}
```

Example

```
#include <stdio.h>
#include<conio.h>
void main()
{
    char ch; FILE *fp;
    fp=fopen("out.txt","r");
    while(1)
    {
        ch=getc(fp);
        if(ch==EOF)
            break;
        printf("\n%c",ch);
    }
    fclose(fp);
}
```



1. Firstly it searches on the disk the file to be opened.
2. Then it loads the file from the disk into a place in memory called buffer.
3. It sets up a character pointer that points to the first character of the buffer.

References

https://www.geeksforgeeks.org/basics-file-handling-c/

https://www.programiz.com/c-programming/c-file-examples