Shastri 6th Semester

Computer Science

Unit: 4th

Concept of Files in C language

In C, files are treated as streams of bytes, and can be read from or written to using a variety of functions provided by the C standard library. These functions include fopen(), fclose(), fread(), fwrite(), fprintf(), fscanf(), and many others.

The fopen() function is used to open a file for reading or writing. It takes a filename and a mode as arguments and returns a pointer to a FILE structure, which is used to represent the file in subsequent operations. The mode can be "r" for reading, "w" for writing, or "a" for appending.

The fclose() function is used to close a file that has been opened with fopen(). It takes a pointer to a FILE structure as an argument and returns zero if the operation was successful.

The fread() and fwrite() functions are used to read and write binary data from and to a file, respectively. They take a pointer to a buffer, the number of bytes to read or write, and a pointer to a FILE structure as arguments.

The fprintf() and fscanf() functions are used to read and write formatted text data from and to a file, respectively. They work similarly to their non-file equivalents printf() and scanf() but take a pointer to a FILE structure as the first argument.

It is important to note that when working with files in C, it is the programmer's responsibility to handle errors that may occur during file operations, such as

attempting to read from a file that doesn't exist or trying to write to a file that is read-only.

File opening in various modes and closing of a file

Mode	Description	Opening	Closing	Reading	Writing
"r"	Open for reading.	FILE *fp = fopen("example.txt", "r");	fclose(fp);	fread(buffer, 1, size, fp); or fscanf(fp, "%s", buffer);	Not applicable, as the file is opened for reading
"w"	Open for writing. If the file already exists, its contents will be truncated.	FILE *fp = fopen("example.txt", "w");	fclose(fp);	Not applicable, as the file is opened for writing	fwrite(buffer, 1, size, fp); or fprintf(fp, "%s", buffer);
"a"	Open for writing. If the file already exists, new data will be appended to the end of the file.	FILE *fp = fopen("example.txt", "a");	fclose(fp);	Not applicable, as the file is opened for writing	fwrite(buffer, 1, size, fp); or fprintf(fp, "%s", buffer);
"r+"	Open for reading and writing. The file pointer is positioned at the beginning of the file.	FILE *fp = fopen("example.txt", "r+");	fclose(fp);	fread(buffer, 1, size, fp); or fscanf(fp, "%s", buffer);	fwrite(buffer, 1, size, fp); or fprintf(fp, "%s", buffer);

"w+"	Open for reading and writing. If the file already exists, its contents will be truncated. The file pointer is positioned at the beginning of the file.	FILE *fp = fopen("example.txt", "w+");	fclose(fp);	fread(buffer, 1, size, fp); or fscanf(fp, "%s", buffer);	fwrite(buffer, 1, size, fp); or fprintf(fp, "%s", buffer);
"a+"	Open for reading and writing. If the file already exists, new data will be appended to the end of the file. The file pointer is positioned at the end of the file.	FILE *fp = fopen("example.txt", "a+");	fclose(fp);	fread(buffer, 1, size, fp); or fscanf(fp, "%s", buffer);	fwrite(buffer, 1, size, fp); or fprintf(fp, "%s", buffer);

Advantages of File Handling

File handling has several advantages in C programming. Some of these include:

Persistence: Data stored in files can be saved and retrieved even after the program that created the data has been terminated.

Separation of concerns: File handling allows you to separate the concerns of data storage and data manipulation by allowing you to write code that specifically handles data storage in files, while other code handles data manipulation.

Portability: Programs that use file handling can be easily ported to different platforms and operating systems, as the C standard library provides a consistent set of functions for working with files across different environments.

Large data handling: File handling allows you to work with large amounts of data that cannot be stored in memory, by reading and writing data in small chunks.

Organization: File handling allows you to organize data in a way that makes it easy to find and retrieve specific pieces of information, for example by storing data in separate files based on specific criteria.

Backup: File handling allows you to create backup copies of data, which can be useful in case of data loss or corruption.

Multi-user: File handling allows multiple users to access the same file simultaneously, and can be useful in multi-user environments where data needs to be shared between multiple users or programs.

In C, reading and writing to a file is done using file streams, which are implemented using the FILE type and the fopen(), fread(), fwrite(), fclose() and other related functions.

To read from a file, you first need to open the file using the fopen() function and assign the returned file pointer to a FILE type variable. Then, you can use the fread() or fscanf() function to read the data from the file and store it in a variable. Once you are done reading, you should close the file using the fclose() function.

For example, the following code reads the contents of a text file called "example.txt" and stores it in a character array called "buffer":

Example

```
FILE *fp;

fp = fopen("example.txt", "r");

char buffer[100];

fread(buffer, sizeof(char), 100, fp);

printf("File contents: %s\n", buffer);

fclose(fp);
```

To write to a file, you first need to open the file using the fopen() function and assign the returned file pointer to a FILE type variable, this time with the mode "w" or "a" (write or append). Then, you can use the fwrite() or fprintf() function to write the data to the file. Once you are done writing, you should close the file using the fclose() function.

For example, the following code writes the contents of a string variable called "message" to a text file called "example.txt":

Example

```
FILE *fp;
fp = fopen("example.txt", "w");
char *message = "Hello World!";
fwrite(message, sizeof(char), strlen(message), fp
```

Some Question and Answer

1. What is file handling in C?
2. How do you open a file in C?
3. What are the different modes for opening a file in C?
4. How do you read from a file in C?
5. How do you write to a file in C?
6. How do you close a file in C?
7. How do you move the file pointer in C?
8. How do you check the end of a file in C?
9. How do you append data to a file in C?
10. How do you use command line arguments to open a file in C?
11. How do you use error handling techniques in file handling in C?
12. How do you use structures and binary files in C?
13. How do you use file locking mechanisms in C?
14. How do you use file I/O redirection in C?
15. How do you use fprintf and fscanf for formatted file I/O in C?