

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

C File Handling

In this tutorial, you will learn about file handling in C. You will learn to handle standard I/O in C using `fprintf()`, `fscanf()`, `fread()`, `fwrite()`, `fseek()` etc. with the help of examples.

A file is a container in computer storage devices used for storing data.

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 a few commands in C.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

Types of Files

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

1. Text files
2. Binary files

1. Text files

Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.

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 the least security and takes bigger storage space.

2. Binary files

Binary files are mostly the **.bin** files in your computer.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

asily, and provides better security than text files.

File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

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

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

Opening a file - for creation and edit

Opening a file is performed using the `fopen()` function defined in the `stdio.h` header file.

The syntax for opening a file in standard I/O is:

```
ptr = fopen("filename", "mode");
```

For example,

```
fopen("E:\\cprogram\\newprogram.txt", "w");  
  
fopen("E:\\cprogram\\oldprogram.bin", "rb");
```

- Let's suppose the file `newprogram.txt` doesn't exist in the location `E:\\cprogram`. The first function 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.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

The reading mode only allows you to read the file, you cannot write into the file.

Opening Modes in Standard I/O		
Mode	Meaning of Mode	During Inexistence of file
<code>r</code>	Open for reading.	If the file does not exist, <code>fopen()</code> returns NULL.
<code>rb</code>	Open for reading in binary mode.	If the file does not exist, <code>fopen()</code> returns NULL.
<code>w</code>	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
<code>wb</code>	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup)).

	is added to the end of the file.	
<code>ab</code>	Open for append in binary mode. Data is added to the end of the file.	If the file does not exist, it will be created.
<code>r+</code>	Open for both reading and writing.	If the file does not exist, <code>fopen()</code> returns NULL.
<code>rb+</code>	Open for both reading and writing in binary mode.	If the file does not exist, <code>fopen()</code> returns NULL.
<code>w+</code>	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup)).

<code>wb+</code>	and writing in binary mode.	If the file does not exist, it will be created.
<code>a+</code>	Open for both reading and appending.	If the file does not exist, it will be created.
<code>ab+</code>	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

Closing a File

The file (both text and binary) should be closed after reading/writing.

Closing a file is performed using the `fclose()` function.

```
fclose(fptr);
```

Here, `fptr` is a file pointer associated with the file to be closed.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup)).

`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.

Example 1: Write to a text file

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

```
int num;
FILE *fptr;

// use appropriate location if you are using MacOS or
fptr = fopen("C:\\program.txt", "w");

if(fptr == NULL)
{
    printf("Error!");
    exit(1);
}

printf("Enter num: ");
scanf("%d", &num);

fprintf(fptr, "%d", num);
fclose(fptr);

return 0;
}
```

This program takes a number from the user and stores in the file `program.txt`.

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.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

```
#include <stdlib.h>

int main()
{
    int num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.txt", "r")) == NULL){
        printf("Error! opening file");

        // Program exits if the file pointer returns NULL.
        exit(1);
    }

    fscanf(fptr, "%d", &num);

    printf("Value of n=%d", num);
    fclose(fptr);

    return 0;
}
```

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

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

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

Reading and writing to a binary file

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

Writing to a binary file

To write into a binary file, you need to use the `fwrite()` function. The functions take four arguments:

1. address of data to be written in the disk
2. size of data to be written in the disk
3. number of such type of data
4. pointer to the file where you want to write.

```
fwrite(addressData, sizeData, numbersData, pointerToFile)
```

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

```
struct threeNum
{
    int n1, n2, n3;
};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.bin", "wb")) == NULL)
        printf("Error! opening file");

    // Program exits if the file pointer returns NU
    exit(1);
}

for(n = 1; n < 5; ++n)
{
    num.n1 = n;
    num.n2 = 5*n;
    num.n3 = 5*n + 1;
    fwrite(&num, sizeof(struct threeNum), 1, fptr);
}
```

In this program, we create a new file `program.bin` in the C drive.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup)).

The first parameter takes the address of `num` and the second parameter takes the size of the structure `threeNum`.

Since we're only inserting one instance of `num`, the third parameter is `1`. And, the last parameter `*fptr` points to the file we're storing the data.

Finally, we close the file.

Reading from a binary file

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

```
fread(addressData, sizeData, numbersData, pointerToFile);
```

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

```
struct threeNum
{
    int n1, n2, n3;
};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.bin", "rb")) == NULL)
        printf("Error! opening file");

    // Program exits if the file pointer returns NU
    exit(1);
}

for(n = 1; n < 5; ++n)
{
    fread(&num, sizeof(struct threeNum), 1, fptr);
    printf("n1: %d\\tn2: %d\\tn3: %d", num.n1, num.n2,
}
fclose(fptr);
```

In this program, you read the same file `program.bin` and loop through the records one by one.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup)).

You'll get the same records you inserted in **Example 6**.

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);
```

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

Different whence in fseek()

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.

Example 5: fseek()

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

```
int n1, n2, n3;
};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.bin", "rb")) == NULL)
        printf("Error! opening file");

    // Program exits if the file pointer returns NU
    exit(1);
}

// Moves the cursor to the end of the file
fseek(fptr, -sizeof(struct threeNum), SEEK_END);

for(n = 1; n < 5; ++n)
{
    fread(&num, sizeof(struct threeNum), 1, fptr);
    printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n
```

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

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

Examples

examples)

([https://twitter.com/intent/tweet?text=Check this amazing article: C File Handling&via=programiz&url=https://www.programiz.com/c-programming/c-file-input-output](https://twitter.com/intent/tweet?text=Check%20this%20amazing%20article%3A%20C%20File%20Handling&via=programiz&url=https://www.programiz.com/c-programming/c-file-input-output))

Was this article helpful?



Related Tutorials

[C Tutorial](#)

[C Files Examples](#)

(</c-programming/c-file-examples>)

[C Tutorial](#)

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

([https://www.programiz.com/learn-c?utm_campaign=programiz-App homepage&utm_source=programiz-website-python-app-popup](https://www.programiz.com/learn-c?utm_campaign=programiz-App%20homepage&utm_source=programiz-website-python-app-popup))

[C Tutorial](#)

[C Dynamic Memory Allocation](#)

([/c-programming/c-dynamic-memory-allocation](#))

[C Tutorial](#)

[C Functions](#)

([/c-programming/c-functions](#))