FEBRUARY 1, 2020 / #C

File Handling in C — How to Open, Close, and Write to Files

If you’ve written the C `helloworld` program before, you already know basic file I/O in C:

```c
// Import IO functions.
#include <stdio.h>

int main() {
    // This printf is where all the file IO magic happens!
    // How exciting!
    printf("Hello, world!\n");
    return EXIT_SUCCESS;
}
```

File handling is one of the most important parts of programming. In C, we use a structure pointer of a file type to declare a file:

```c
FILE *fp;
```

C provides a number of build-in function to perform basic file operations:

- `fopen()` - create a new file or open a existing file
- `fclose()` - close a file
- `getc()` - reads a character from a file
- `putc()` - writes a character to a file
- `fscanf()` - reads a set of data from a file
- `fprintf()` - writes a set of data to a file
- `getw()` - reads a integer from a file
- `putw()` - writes a integer to a file

- `rewind()` - set the position to the beginning point

Opening a file

The `fopen()` function is used to create a file or open an existing file:

```
fp = fopen(const char filename,const char mode);
```

There are many modes for opening a file:

- `r` - open a file in read mode
- `w` - opens or create a text file in write mode
- `a` - opens a file in append mode
- `r+` - opens a file in both read and write mode
- `a+` - opens a file in both read and write mode
- `w+` - opens a file in both read and write mode

Here's an example of reading data from a file and writing to it:

```
#include<stdio.h>
#include<conio.h>
main()
{
FILE *fp;
char ch;
}
fclose(fp);
fp = fopen("hello.txt", "r");

while( (ch = getc(fp)! = EOF)
  printf("%c",ch);

fclose(fp);
}
```

Now you might be thinking, "This just prints text to the screen. How is this file IO?”

The answer isn’t obvious at first, and needs some understanding about the UNIX system. In a UNIX system, everything is treated as a file, meaning you can read from and write to it.

This means that your printer can be abstracted as a file since all you do with a printer is write with it. It is also useful to think of these files as streams, since as you’ll see later, you can redirect them with the shell.

So how does this relate to `helloworld` and file IO?

When you call `printf` , you are really just writing to a special file called `stdout` , short for **standard output**. `stdout` represents the standard output as decided by your shell, which is usually the terminal. This explains why it printed to your screen.

There are two other streams (i.e. files) that are available to you with effort, `stdin` and `stderr` . `stdin` represents the **standard input**,

Rudimentary File IO, or How I Learned to Lay Pipes

Enough theory, let’s get down to business by writing some code! The easiest way to write to a file is to redirect the output stream using the output redirect tool, `>`.

If you want to append, you can use `>>`:

```
# This will output to the screen...
./helloworld
# ...but this will write to a file!
./helloworld > hello.txt
```

The contents of `hello.txt` will, not surprisingly, be

```
Hello, world!
```

Say we have another program called `greet`, similar to `helloworld`, that greets you with a given `name`:

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

int main() {
    // Print the greeting.
    printf("Hello, %s!", name);
    return EXIT_SUCCESS;
}
```

Instead of reading from the keyboard, we can redirect `stdin` to read from a file using the `<` tool:

```
# Write a file containing a name.
echo Kamala > name.txt
# This will read the name from the file and print out the greetir
./greet < name.txt
# ==> Hello, Kamala!
# If you wanted to also write the greeting to a file, you could c
```

Note: these redirection operators are in `bash` and similar shells.

The Real Deal

The above methods only worked for the most basic of cases. If you wanted to do bigger and better things, you will probably want to work with files from within C instead of through the shell.

To accomplish this, you will use a function called `fopen`. This function takes two string parameters, the first being the file name and the second being the mode.

The mode are basically permissions, so `r` for read, `w` for write, `a` for append. You can also combine them, so `rw` would mean you

After you have a `FILE` pointer, you can use basically the same IO commands you would've used, except that you have to prefix them with `f` and the first argument will be the file pointer. For example, `printf`'s file version is `fprintf`.

Here's a program called `greetings` that reads a from a file containing a list of names and write the greetings to another file:

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

int main() {
    // Create file pointers.
    FILE *names = fopen("names.txt", "r");
    FILE *greet = fopen("greet.txt", "w");

    // Check that everything is OK.
    if (!names || !greet) {
        fprintf(stderr, "File opening failed!\n");
        return EXIT_FAILURE;
    }

    // Greetings time!
    char name[20];
    // Basically keep on reading untill there's nothing left.
    while (fscanf(names, "%s\n", name) > 0) {
        fprintf(greet, "Hello, %s!\n", name);
    }

    // When reached the end, print a message to the terminal to i
    if (feof(names)) {
        printf("Greetings are done!\n");
    }

    return EXIT_SUCCESS;
}
```

```
Kamala
Logan
Carol
```

Then after running `greetings` the file `greet.txt` will contain:

```
Hello, Kamala!
Hello, Logan!
Hello, Carol!
```

Forum Donate

Learn to code — free 3,000-hour curriculum

- What is a Linked List?
- Full Stack Career Guide
- JavaScript Array Length
- SQL Temp Table
- Pandas Count Rows
- Python Dict Has Key
- String to Array in Java
- Python Merge Dictionaries
- Center Text Vertically CSS
- Install Java in Ubuntu
- Python Sort Dict by Key
- Sets in Python
- HTML Form Basics
- Python End Program
- Python List to String
- Python Import from File
- Copy a Directory in Linux
- What's a Greedy Algorithm?
- Python Ternary Operator
- Smart Quotes Copy/Paste
- Kotlin vs Java
- Comments in YAML
- Python XOR Operator
- Exit Function in Python
- Parse a String in Python
- Reactive Programming Guide
- Edit Commit Messages in Git

Mobile App

Download on the App Store

GET IT ON Google Play

Our Charity