

File Handling

It is possible to handle files using the `C` language. Let's see how.

Opening and Closing Files

First of all, we need to open a file in order to work with it, and close it when we're done. That is why, the `stdio` library provides two functions — `fopen()` and `fclose()` meaning *file open* and *file close*.

Opening a File

To open a file, use the `fopen()` function, passing the name of the file (or the path) and the mode of opening it. There are various modes:

| Mode | Meaning |
|-----------------|----------------------|
| <code>r</code> | Read only |
| <code>w</code> | Write only |
| <code>a</code> | Append only |
| <code>r+</code> | Read and write only |
| <code>w+</code> | Write and read only |
| <code>a+</code> | Append and read only |

The difference between `w` and `a` is that when we open in writing mode, any new content written to the file will overwrite the existing content. It must be used with caution. When opening in append mode, any new content written to the file will be *added* where the cursor points on the file (usually at the the last character).

Now, we first create a file pointer that points to a file, and the opened file will be stored in it. It can be done like this

```
FILE *fp;  
fp = fopen("sample.txt", "r");
```

In the following code, we created a file pointer `fp` which is pointing to the file `sample.txt` in reading mode.

Now, it is important to check if the file exists or not. We may not always given existing file names. TO do that, check if the pointer is pointing to a file. If not, it points to `NULL`.

```
if (fp == NULL)
    printf("File does not exist!");
```

Closing a File

It is important to close and put back things wherever they belong to when we are doing working with them. When we open a file, it is like we bring it to workplace. Once we're done working with it, it must be kept back or *closed*.

To close a file, use the function `fclose()` and pass the file pointer as the argument.

```
fclose(fp);
```

It is important to close files, so that:

- Flushes new data to disk
- Prevents data loss and memory leak
- Releases system resources

Accessing File Content

To read or write int files, we know what mode we want to use. But how do we read or write into a file? To do so, the library has provided several functions for

- Writing
 - `fprintf`: Used to format a string just like in `printf` and write it into a file.
 - `fputs`: Used to write a string into a file.
 - `fputc`: Used to write a character into a file.
- Reading
 - `fscanf`
 - `fgets`

- `fgetc`

Here is a snippet showing how they can be used:

```
// Writing
fprintf(fp, "Hello world! %d", 10);
fputs("Hello world!", fp);
fputc('A', fp);

// Reading
fscanf(fp, "%d", &n);
fgets(str, 50, fp);
ch = fgetc(fp);
```

Checking End of File `EOF`

When a file keeps reading data, it reaches the final character at some point, and hence, end of the file. It won't read any data beyond that.

To check the end of file, try reading the next line and see if it is `EOF` which is a keyword to check the end of file.

```
while (ch != EOF) {
    printf("%c", ch);
    ch = fgetc(fp);
}

// or

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