Programming Languages 1
Lesson 6 - Guide

File handling

In a C program one can work with files. It is possible to read something form a file or to write something into a file. There are two different ways to do this. One of them is based on formatted input/output to manage simple text files. This lesson restricted to only this.

In a C program files can be referred by a special file descriptor, a `FILE` type pointer (so the name of the type is `FILE` with capital letters). A file descriptor variable obtain value in an opening process. All files have to be opened first by the special function of the `stdio.h` header file. For example:

```
// The "apple.txt" file is opened for reading.
FILE *f1=fopen("apple.txt","r");
// The "banana.txt" file is opened for writing.
FILE *f2=fopen("banana.txt","w");
// The "cherry.txt" file is opened for appending.
FILE *f3=fopen("cherry.txt","a");
```

Thus the value provided by the `fopen` function is assigned to a `FILE` pointer. Later the name of the file never appears, just the file descriptor variable. If the opening process was not successful the `fopen` returns by a `NULL` pointer value. In case of writing, when the file does not exists the system creates a new empty file, otherwise all the previous file content will be erased. Appending means writing to the end of the file keeping the original content. If the first parameter of the `fopen` is just a filename, than the file have to be in the same directory as the executable program, but path and filename can be also used.

If the file is opened for writing, one can use the `fprintf` function to write something into the file. The `fprintf`, works in a similar way, than `printf`, just is has a new first parameter, a file descriptor.

```
fprintf(f2, "%d bananas", 3+4);
```

The `"7 bananas"` text is written into the file referred by f2.

In order to read file content into variable(s), the `fscanf` function can be used. For example to read the next integer number from the file referred by `f1`, into the `num` integer variable, can be written as

```
fscanf(f1,"%d", &num);
```

After each reading/writing operation the system always know the actual position in the given file for the next reading/writing operation.

After we finish all the necessary input/output file operations, files must be closed by the `fclose` procedure.

```
fclose(f3);
```

Sometimes (especially during read) we have to know, whether we reached the end of the input file or not (is there any more value to read). The feof function is used to help the decision. Its parameter is a file descriptor and the function returns return by 0 (false) if we are not at the end of file, else it returns by 1 (true). (Don't forget the `!` operator means the logical negation.)

```
while(!feof(f1)){…} //iteration until there is further content in f1
```

The next short C program copies all the characters from "abc.txt" to the end of "def.txt" in a character-by-character way.

```
#include<stdio.h>
int main(){
  char c;
  FILE *source,*destination;
  source=fopen("abc.txt","r");
```

```
  if(source==NULL) return 1; //error
  destination=fopen("def.txt","a");
  while(!feof(source)){
    fscanf(source,"%c",&c);
    fprintf(destination,"%c",c);
  }
  fclose(source);
  fclose(destination);
  return 0;
}
```

Further readings:

- Brian W. Kernighan, Dennis M. Ritchie: *The C programming language*, Prentece Hall (2012)
- https://www.geeksforgeeks.org/basics-file-handling-c/
- https://www.programiz.com/c-programming/c-file-input-output