Intro to File Processing in C

File handling

- Storing of data in a file using program

What can we do?

- Create a new file
- Opening an existing file
- Read from existing file
- Write to a file
- Moving data to a specific location on the file
- Closing the file

To work with files, we need to declare a pointer of type file. This declaration is needed for communication between the file and the program

```
FILE *fptr;
```

fopen()

FILE *fopen(**const char** * filename, **const char** * mode);

Filename: absolute path to file

fclose()

int fclose(FILE *stream)

Mode

File doesn’t exist:

"w", "a" modes can create file

"r" mode returns NULL

Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. Data is added to the end of the file.	If the file does not exist, it will be created.
ab	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.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exist, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

fscanf(),fgets(),fgetc()

int fscanf(FILE *stream, const char *charPointer[])

On success, this function returns the number of values read and on error or end of the file it returns EOF or -1.

char *fgets(char *str, int n, FILE *fp);

The function reads a string from the file pointed to by fp into the memory pointed to by str. On error or end of the file it returns NULL.

char* fgetc(FILE *stream) :

It reads a single character from the file and increments the file position pointer. To use this function file must be opened in read mode. On success, it returns the ASCII value of the character but you can also assign the result to a variable of type char. On failure or end of file, it returns EOF or -1.

fprintf(),fputs(),fputc()

int fprintf(FILE *stream, char *string[])

int fputs(const char *str, FILE *fp);

This function is used to print a string to the file. On success, it returns 0. On error, it returns EOF or   -1.

int fputc(char character , FILE *stream)

On error, it returns EOF i.e -1.

fread()

reads data from the given **stream** into the array pointed to, by **ptr**. This function is used mainly for binary files.

size_t fread(void *ptr, size_t size, size_t n, FILE *stream)

Parameters:

- **ptr** − the starting address of the memory block where data will be stored after reading from the file. Its minimum size is *size*n* bytes.
- **size** − This is the size in bytes of each element to be read.
- **n** − This is the number of elements, each one with a size of **size** bytes.
- **stream** − This is the pointer to a FILE object that specifies an input stream.

Return value:

The function returns the number of items of size "*size*" that were read successfully. If the return value is not equal to *n* then either end of file was reached or there was an error

fwrite()

writes data from the given **ptr** into the file pointed to by **stream**. This function is used mainly for binary files.

size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream)

Parameters

- **ptr** − it points to the block of memory which contains the data items to be written.
- **size** − This is the size in bytes of each element to be written.
- **n** − This is the number of elements, each one with a size of **size** bytes.
- **stream** − This is the pointer to a FILE object that specifies an output stream.

Return Value

The function returns the number of items of size "*size*" that were read successfully. If the return value is not equal to *n* then there was an error

Example:

float *f = 100.13;

fwrite(&f, sizeof(f), 1, fp);

fseek()

int fseek(FILE *stream, long int offset, int whence)

Parameters

stream − This is the pointer to a FILE object that identifies the stream.

offset − This is the number of bytes to offset from whence.

whence − This is the position from where offset is added. It is specified by one of the following constants −

SEEK_SET(Beginning of file), SEEK_CUR (Current position of the file pointer), SEEK_END (EOF)

THANK YOU!