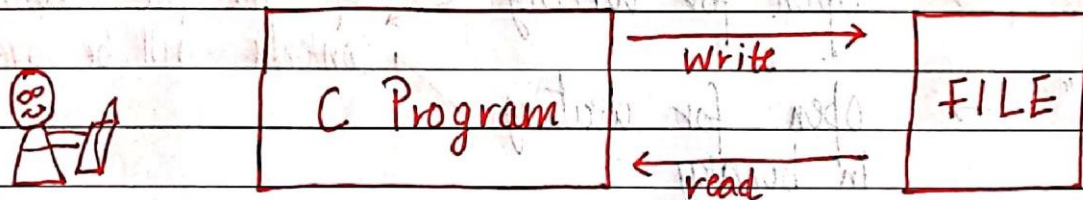


Chapter 10 - File I/O

The Random Access Memory is volatile and its content is lost once the program terminates. In order to persist the data forever we use files.

A file is data stored in a storage device.

A C program can talk to the file by reading content from it and writing content to it.



Programmer

FILE pointer

The "FILE" is a structure which needs to be created for opening the file.

A file pointer is a pointer to this structure of the file.

FILE pointer is needed for communication between the file and the program.

A FILE pointer can be created as follows:

```
FILE *ptr;  
ptr = fopen("filename.ext", "mode");
```


File opening modes in C

C offers the programmers to select a mode for opening a file. Following modes are primarily used in C File I/O

"r" → open for reading

If the file does not exist, fopen returns NULL

"rb" → open for reading in binary

"w" → open for writing

If the file exists, the contents will be overwritten

"wb" → open for writing in binary

"a" → open for append

If the file does not exist, it will be created

Types of files

There are two types of files:

1. Text files (.txt, .c)
2. Binary files (.jpg, .dat)

Reading a file

A file can be opened for reading as follows:

```
FILE * ptr;
ptr = fopen("Harry.txt", "r");
int num;
```


Let us assume that "Harry.txt" contains an integer
We can read that integer using:

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

\Rightarrow fscanf is file counterpart of scanf

This will read an integer from file in num variable.

Quick Quiz: Modify the program above to check whether the file exists or not before opening the file.

CLOSING the file

It is very important to close the file after read or write. This is achieved using fclose as follows:

```
fclose(ptr);
```

This will tell the compiler that we are done working with this file and the associated resources could be freed.

Writing to a file

We can write to a file in a very similar manner like we read the file

```
FILE *fptr;  
fptr = fopen("Harry.txt", "w");
```



```
int num = 432;  
fprintf(fptr, "%d", num);
```

```
fclose(fptr);
```

`fgetc()` and `fputc()`

`fgetc` and `fputc` are used to read and write a character from/to a file

```
fgetc(ptr)
```

⇒ used to read a character from file

```
fputc('c', ptr);
```

⇒ used to write character 'c' to the file

EOF: End of file

`fgetc` returns EOF when all the characters from a file have been read. so we can write a check like below to detect end of file.

```
while (1) {
```

```
    ch = fgetc(ptr);
```

```
    if (ch == EOF) {
```

```
        break;
```

```
    }
```

```
    // code
```

```
}
```

⇒ When all the content of a file has been read, break the loop!