FILE HANDLING IN C

File : named collection of data, typically stored in a secondary storage (eg. hard disk)

Stored as seq. of bytes, logically contiguous.

Every file - starting of file
seq. of bytes (actual data)
end of file

Allows only sequential access of data by a pointer.

Metadata (info about the file) can be maintained before the stream of actual data.

The last byte of file contains EOF character with ASCII code 1A (Hex)

Header file used : <stdio.h>

"r" : Opens a file for reading
"w" : Creates a file for writing (overwrites if data present)
"w+" : Creates file for reading & writing (")
"a" : Opens a file for appending - writing on the end of the file.
"rb" : Read a binary file (read as bytes)
"wb" : Write into a binary file (overwrites if data present)

If a file that does NOT exist is opened for writing or appending, it is created as a new.

File Opening error:

- Trying to read a file that doesn't exist.
- Trying to read a file that doesn't have permission.
- If there is an error, fopen() returns NULL

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

File to Refer : fopen.c

```
int main(){
  FILE *fptr;
  char filename[] = "file2.dat";
  fptr = fopen(filename, "w");   //OR  fptr = fopen("file2.dat", "w");

  if (fptr == NULL){
     printf("Error in creating file");
     exit(-1); }
  else{
     fprintf(fptr, "This is a test file \n");
     printf("file created");
     fclose(fptr);}
}
```

Important

Reading from a file

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

① fgetc fgetc(fptr)

- Reads a single character from a file

File To Refer : fgetc.c

```
while (fscanf(fptr, "%s", buf) == 1){
    printf("%s", buf); }
char buf[100]; // is before while loop
```

fscanf.c

string & var arg. list to take input from a file

③ fgets

fgets(str, size, fptr)

fgets.c

↓

Stores ~~three~~ (size-1) characters from fptr file to str.

File opening errors:

- Trying to read a file that doesn't exist.
- Trying to read a file that doesn't have permission.
- If there is an error, fopen() returns NULL

File to Refer : fopen.c

```
#include<stdio.h>
#include<stlib.h>
int main(){
  FILE *fptr;
  char filename[] = "file2.dat";
  fptr = fopen(filename,"w");   //OR fptr = fopen("file2.dat","w");
  if (fptr == NULL){
     printf("Error in creating file");
     exit(-1);}
  else{
     fprintf(fptr,"This is a test file\n");
     printf("file created");
     fclose(fptr);}
}
```

Important

Reading from a file

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

① fgetc fgetc(fptr)

File To Refer : fgetc.c

- Reads a single character from a file

fscanf

fscanf.c

- Use formatted string & var arg. list to take input from a file

③ fgets

fgets.c

fgets(str, size, fptr)

↓

Stores ~~three~~ (size-1) characters from fptr file to str

④ getc

getc is equivalent to fgetc except that it is a micro.

Writing into a file

① fputc

```
filecopy (FILE *fpin, FILE *fpout) {
    int c;
    while ((c=fgetc(fpin)!=EOF){
        fputc(c, fpout); }
```

② fprintf

```
fprintf(fptr, "example");
        ↓
    file pointer
```

eg. `fprintf(fptr, "Name: %s Age: %d", "Alice", 30);`

③ fputs

```
fputs ("file handling", fptr);
```

fprintf	fputs
When we need to write formatted strings that may include various datatypes & require specific formatting (eg. given above)	When we simply want to write a string to a file w/o any additional formatting.

④ putc

putc is equivalent to fputc except that its a micro.

NOTE:

Using stdout and stdin with fprintf and fscanf resp. behaves as printf and scanf.

eg.
```
fprintf(stdout, "Hello");
fscanf(stdin, "%d", &i);
fprintf(stdout, "%d\n", i);
```

OUTPUT:
```
Hello
15
15
```

Structured Input/Output for Files - Do if time is left!

① fwrite()

② fread