COMPUTER PROGRAMMING

File Handling in C

Files

- **File – place on disc where group of related data is stored**
  - E.g. your C programs

- **High-level programming languages support file operations:**
  - *Naming*
  - *Opening*
  - *Reading*
  - *Writing*
  - *Closing*

Defining and Opening file

- *Filename (e.g. sort.c, input.data)*
- *Data structure (e.g. FILE)*
- *Purpose (e.g. reading, writing, appending)*

Filename

- **String of characters that make up a valid filename**
- **May contain two parts**
  - *Primary*
  - *Optional period with extension*
- *Examples:* ***a.out, prog.c, temp, text.out***

Different operations that can be performed on a file

1. Creation of a new file (fopen with attributes as “a” or “a+” or “w” or “w++”)
2. Opening an existing file (fopen)
3. Reading from file (fscanf)

Different operations that can be performed on a file

4. **Writing to a file (fprintf)**
5. **Moving to a specific location in a file (fseek, rewind)**
6. **Closing a file (fclose)**

The text highlighted in the brackets denotes the functions used for performing those operations.

General format for opening file

```
FILE *filepointer;

/*variable filepointer is pointer to type FILE*/
```

So, the file can be opened as:

```
filepointer = fopen("filename", "mode");

/*opens file with name filename ,  assigns identifier to fp */
```

e.g. `filePointer = fopen("fileName.txt", "w")`

General format for opening file

- **filepointer**
  - contains all information about file
  - Communication link between system and program
- **Mode can be**
  - r open file for reading only
  - w open file for writing only
  - a open file for appending (adding) data

Modes

- **Writing mode**
  - **if file already exists then *contents are deleted*,**
  - **else new file with specified name created**
- **Appending mode**
  - **if file already exists then file opened with contents safe**
  - **else new file created**
- **Reading mode**
  - **if file already exists then opened with contents safe**
  - **else error occurs.**

Modes

- **"r" – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.**
- **"w" – Searches file. If the file exists, its contents are overwritten. If the file doesn't exist, a new file is created. Returns NULL, if unable to open file.**

Modes

- **“a” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.**

Example…..

```
FILE *p1, *p2;

p1 = fopen(“data”,”r”);

p2= fopen(“results”, w”);
```

Closing a file

- **File must be closed as soon as all operations on it completed**

- **This Ensures**
  - **All outstanding information associated with file flushed out from buffers**
  - **All links to file broken**
  - **Accidental misuse of file prevented**

- **If want to change mode of file, then first close and open again**

Examples…..

Syntax: fclose(file_pointer);

Example:

```
FILE *p1, *p2;
p1 = fopen(“INPUT.txt”, “r”);
p2 =fopen(“OUTPUT.txt”, “w”);
……..
……..
fclose(p1);
fclose(p2);
```

Operations on files

Reading from a file

The file read operations can be performed using functions fscanf. Both the functions performed the same operations as that of printf and gets but with an additional parameter, the *file pointer.*

So, it depends on you if you want to read the file line by line or character by character.

```
FILE * filePointer;

filePointer = fopen(“fileName.txt”, “r”);

fscanf(filePointer, "%s %s %s %d", str1, str2, str3, &year);
```

Writing a file

The file write operations can be perfomed by the functions fprintf with similarities to read operations.

```
FILE *filePointer ;

filePointer = fopen(“fileName.txt”, “w”);

fprintf(filePointer, "%s %s %s %d", "We", "are", "in", 2012);
```

- **C provides several different functions for reading/writing**

- **fprintf() – write set of data values**
- **fscanf() – read set of data values**

fscanf() and fprintf()

- similar to scanf() and printf()
- in addition provide file-pointer
- given the following
  - file-pointer f1 (points to file opened in write mode)
  - file-pointer f2 (points to file opened in read mode)
  - integer variable i
  - float variable f
- Example:

```
fprintf(f1, "%d %f\n", i, f);
fprintf(stdout, "%f \n", f);      /*note: stdout refers to screen */
fscanf(f2, "%d %f", &i, &f);
```

- fscanf returns EOF when end-of-file reached

EXAMPLE:
FILE WRITING (FILES & STRUCTURE)

```
#include<stdio.h>
int main( )
{
        FILE *fp ;
        char another = 'Y' ;
        struct emp
        {
                char name[40] ;
                int age ;
                float bs ;
        } ;
        struct emp e ;

        fp = fopen ("record.txt", "w" ) ;

        while ( another == 'Y'  || another == 'y' )
        {
                printf ( "\nEnter name, age and basic salary: " ) ;
                scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;
                fprintf ( fp, "%s %d %f\n", e.name, e.age, e.bs ) ;
                printf ( "Add another record (Y/N) " ) ;
                fflush ( stdin ) ;
                scanf("%c",&another);
        }
        fclose ( fp ) ;
        return 0;
}
```

EXAMPLE:
FILE READING (FILES & STRUCTURE)

```
#include<stdio.h>
int main( )
{
	FILE *fp ;
	struct emp
	{
		char name[40] ;
		int age ;
		float bs ;
	} ;
	struct emp e ;

	fp = fopen ("record.txt", "r" ) ;

	while ( fscanf ( fp, "%s %d %f", e.name, &e.age, &e.bs ) != EOF )
		printf ( "\n%s %d %f", e.name, e.age, e.bs ) ;

	fclose ( fp ) ;
	return 0;
}
```

EXAMPLE:
COPY ONE FILE TO ANOTHER FILE

```
#include<stdio.h>
int main( )
{
	FILE *fp,*fp2 ;
	struct emp
	{
		char name[40] ;
		int age ;
		float bs ;
	} ;
	struct emp e ;

	fp = fopen ("record.txt", "r" ) ;
	fp2 = fopen ("copy.txt", "w" ) ;

	while ( fscanf ( fp, "%s %d %f", e.name, &e.age, &e.bs ) != EOF )
		fprintf ( fp2, "%s %d %f\n", e.name, e.age, e.bs ) ;

	fclose (fp) ;
	fclose (fp2) ;
	return 0;
}
```