CHAPTER TWELVE: C- Programming: Managing Files in C

Chapter Objectives

By the end of this chapter the learner should be able to

- Describe the concept of files in C
- Describe and use/implement the C program basic file operations
  - ✓ Define and open a file
  - ✓ Close a file
  - ✓ Read from a file
  - ✓ Write into a file
  - ✓ Close a file

12.1. Introduction

A file is a place on the disk where a group of related data is stored. C supports a number of functions that have the ability to perform basic file operations, which includes;

- Naming a file
- Opening a file
- Reading data from a file
- Writing data to a file
- Closing a file

There are two distinct ways to perform file operations in C

1. Low-level I/O and used the Unix system calls
2. High-level I/O operation and uses functions in C's standard I/O library
3. High Level I/O functions in C

Function	Operation
fopen()	Creates a new file for use / opens an existing file for use
fclose()	Closes a file that had been opened for use
getc()	Reads a character from a file
putc()	Writes a character to file
fprintf()	Writes a set of data values to a file

fscanf()	Reads a set of data values from a file
getw()	Reads an integer from a file
putw()	Writes an integer in a file
fseek()	Set the position to a desired point in a file
ftell()	Fives the current position in the file(in terms of bytes from the start)
rewind()	Sets the position to the beginning of the file

There are many other functions and not all are supported by all the C compilers

12.2. Defining and Opening aFile.

If we want to store data in a file in the secondary memory, there is need to specify the following things about the file;

- Filename: File name is a string of characters that make up a valid filename for the operating system. It may contain two parts a primary name and an optional period with the extension. Examples
  Input.data, Student.c, Text.out

- Data structure: Data structure of a file is defined a FILE in the library of standard I/O function definitions. Therefore all files should be declared as type FILE before they are used. FILE is a defined data type

- Purpose. When we open a file we must specify what we want to do with the file, example read from or write to it. Format
  FILE *fp;
  fp = **fopen**("filename", "mode");

  FILE *fp declares the variable fp as a "pointer to the data type FILE. FILE is a structure that is defined in the I/O library.

  fp = fopen('filename, "mode"); open the file named and assigns an identifier to the FILE type pointer fp. The fp pointer which contains all the information about the file is subsequently used as a communication link between the system and the program.
  The statement also specifies the purpose of opening the file. Options includes;

- r = open a file for reading only
- w – opens a file for writing only.
- a – opens a file for appending (or adding) data to it.

When opening a file;

- When the mode is ‘writing’, a file with the specified name is created if the file does not exist. The contents are deleted if the file already exists.
- When the purpose is ‘appending’, the file is opened with the current contents safe. A file with the specified name is created if the file does not exist.
- If the purpose is ‘reading’ and if it exists, then the file is opened with the current contents safe otherwise abd error occurs.

Example

FILE *p1, *p2;

p1 = fopen(“data”, “r”);

p2 = fopen(“results”, “w”);

- p1 is opened for reading and if it does not exist an error occurs.
- p2 is opened for writing and if it exists, its contents are deleted and the file opened a new file.

Additions modes includes;

r+ - existing file is opened to the beginning for both read and write.

w+ - same as w except both for reading and writing.

a+ - same as a except both for read and writing.

12.3. Closing a File

A file must be closed as soon as all operations on it have been completed. This ensures that all outstanding information associated with the file is flushed out from the buffers and all links to the file are broken. It also prevents any accidental misuse of the file. Format

- **fclose(file_pointer);**

example

FILE *p1, *p2;

p1 = fopen(“INPUT”, “w”);

p2 = fopen(“OUTPUT”, “r”);

……

…….

fclose(p1);

fclose(p2);

Once a file is closed its file pointer can be used for another file.

12.4. Input / Output Operations on Files

Once a file is opened, reading out of or writing into it is accomplished using the standard I/O routines mentioned earlier.

Getc and putc functions

These are analogous to getchar and putchar functions that handles one character at a time

Example;

- getc(c, fp1); writes the character contained in the character variable c to the file associated with FILE pointer fp1, when the file is opened with write mode.
- c = getc(fp2); reads a character from the file whose file pointer is fp2.

Prgram Example 12.1: Program to read data keyed in through the keyboard and write it into a file, then read the data from the file and display it on the screen

- The End-Of-File (EOF) marker is achieved through pressing ***<control-z>***

Data to enter: ***This is a program to test the file handling features of C program.***

```
#include <stdio.h>
#include<conio.h>
main()
{
FILE *f1;
char c;
printf("Data Input\n\n");
                                    /* open the INPUT file for writing */
f1 = fopen("INPUT", "w");
                                    /* get character from the Keyboard */
while ((c=getchar()) != EOF)
putc(c, f1);
                                    /* close the INPUT file */
fclose(f1);
printf("\nData Output \n\n");
                                    /* Open the INPUT file for reading */
```

```
f1 = fopen("INPUT", "r");
                                    /* Read a character from the INPUT file */
 while ((c=getc(f1)) !=EOF)
                                /* Display the character on the screen */
 printf("%c", c);
                                /* Close the INPUT file */
fclose(f1);

getch();
}
```

Testing for End-Of-File (EOF) condition is important, as attempting to read past the EOF might either cause the program to terminate with an error or result in an infinite loop situation.

12.5. The getw and putw Functions

getw and **putw** are integer-oriented functions. Format

putw(interger, fp);

getw(fp);

Program Example 12.2: Write a program to read integers from a file called DATA and write all odd integers into a file called ODD and all even integers into a file called EVEN.

Test data

111	222	333	000	121
444	555	666	232	343
777	888	999	454	-1

-1 is entered to terminate the reading and file is closed

```
#include <stdio.h>
#include<conio.h>
main()
{
FILE *f1, *f2, *f3;
```

```
int number, i;

printf ("Content of DATA file \n\n");
f1 = fopen("DATA", "w");
for (i = 1; i<= 30; i++)
{
 scanf ("%d", &number);

if (number == -1) break;
putw(number, f1);
}
fclose(f1);

f1 = fopen("DATA", "r");
f2 = fopen("ODD", "w");
f3 = fopen("EVEN", "r");

while((number = getw(f1)) != EOF)
{
  if(number  %2 == 0)
 putw(number, f3);
else
 putw(number, f2);
}

fclose(f1);
fclose(f2);
fclose(f3);

f2 = fopen("ODD", "r");
f3 = fopen ("EVEN", "r");

printf ("\n\nContents of ODD file\n\n");
  while ((number = getw(f2)) != EOF)
```

```
  printf("%4d", number);

 printf("\n\nContents of EVEN file\n\n");
    while ((number = getw(f3)) != EOF)
    printf("%4d", number);

 fclose(f2);
 fclose(f3);

 getch();
 }
```

12.6. The fprintf and fscanf Functions

fprintf and fscanf perform I/O operations on files.

Format for fprint()

fprintf(fp, "control string", list);

- fp is the file pointer associated with a file that has been opened for writing
- control string contains output specifications for the items in the list
- list may include variables, constants and strings

Example

fprintf(f1, "%s %d %f", name, age, 7.5);

fscanf Format

fscanf(fp, control string", list);

example

fscanf(f1, "%s %d", item, &quantity);

Program Example 12.3: Write a program to open a file named INVENTORY and store in it the following data;

Item Name	Number	Price	Quantity
AAA-1	111	17.50	115

BBB-2	125	36.00	75
C-3	247	31.75	104

Use file name ‘INVENTORY’

```
#include <stdio.h>
#include<conio.h>
main()
{
FILE *fp;
int  number, quantity,  i;
float  price, value;
char  item[10], filename[10];

printf("Input file name \n");
scanf ("%s", filename);
fp = fopen(filename, "w");
printf("Input Inventory data\n\n");
printf("Item name  Number  Price Quantity\n");

for (i =1; i <=3; i++)
{
fscanf(stdin, "%s    %d   %f   %d", item, number, &price, &quantity);
fprintf(fp, "%s    %d    %.2f  %d", item, number, price, quantity);
}

fclose(fp);
fprintf(stdout, "\n\n");

fp = fopen(filename, "r");
printf("Item name      Number    Price    Quantity    Value\n");
  for (i =1 ; i <=3; i++)
{
```

```
  fscanf(fp, "%s    %d    %f    %d", item, &number, &price, &quantity);
   value = price * quantity;
fprintf(stdout, "%-8s  %7d  %8.2f  %8d  %11.2f\n" ,
                item, number, price, quantity, value);
}
fclose(fp);
getch();
}
```

12.7. Error Handling during I/O Operations

Typical I/O operations errors include;

1. Trying to read beyond the end-of-file mark.
2. Device overflow
3. Trying to use a file that has not been opened
4. Trying to perform an operation on a file, when the file is opened for another type of operation
5. Opening a file with an invalid filename
6. Attempting to write to a write-protected file.

C has two status-inquiry library functions; **feof** and **ferror** that helps to detect I/O errors in files.

feof is used to detect end-of-file condition. It take the file pointer as its only argument and returns a nonzero integer value if all of the data from the specified file has been read and returns zero otherwise.. example if fp is a file pointer that has been opened then the statement ;

```
 if(feof(fp))
 printf("End ofdata");
```

would display "End of data" on reaching the end of file condition.

Ferror function reports the status of the file indicated. It also takes the file pointer as its only argument and returns a nonzero integer value if an error has been detected up to that point, during processing. It returns zero otherwise. Example

```
    if (ferror(fp) != 0)
    printf ("An error has occurred. \n");
```

or to check if file is opened

- when a file is opened using the **fopen** function, the file pointer is returned, thus if the file could not be opened the function returns zero.
  if (fp == NULL)
  printf("File could not be opened");

Program Example 12.4: Program to Illustrate error handling in file operations

Use file name 'TEST'

```
#include <stdio.h>
#include<conio.h>
main()
{
char   *filename;
FILE *fp1, *fp2;
int i, number;

fp1 = fopen("TEST", "w");
for (i = 10; i<= 100; i+=10)
    putw(i, fp1);

fclose(fp1);

printf("\nInput file name\n");

open_file:
scanf("%s", filename);
if((fp2 = fopen(filename, "r"))== NULL)
{
printf("Cannot open the file.\n");
printf("Type file name again.\n\n");
goto open_file;
}
else
  for (i = 1; i<= 20; i++)
```

```
{
  number = getw(fp2);
      if (feof(fp2))
      {
      printf("\nRan out of data.\n");
      break;
      }
      else
      printf("%d\n", number);
      }

      fclose(fp2);
      getch();
}
```

Chapter Review Questions

1. what do the following statements do?

   a). while ((c=getchar() != EOF)
   putc(c, fl);

   b). while ((m=getw(fl)) != EOF
   prinbtf ((“%5d”, m)

2. Write a program t copy the content from one file to another.
3. Write a program to append one file at the end of another
4. Write a program to compare two files and return 0 is they are equal and 1 if they are not.