Computer Concept & Programming

I semester, BScCSIT

Ambition Guru College

Compiled by :
Ankit Bhattarai

Unit 8

(5 hrs.)

File Handling in C

- Introduction, Basic Terminology Associated with Files, Types of Files, Streams and Files, Binary vs Text File ad File Buffering, File System Structures, Various Types of File Access Methods, Input and Output Operations on Files and Standard Devices, File Operations, Error Handling in Files, Command Line Arguments

Introduction: What is a File?

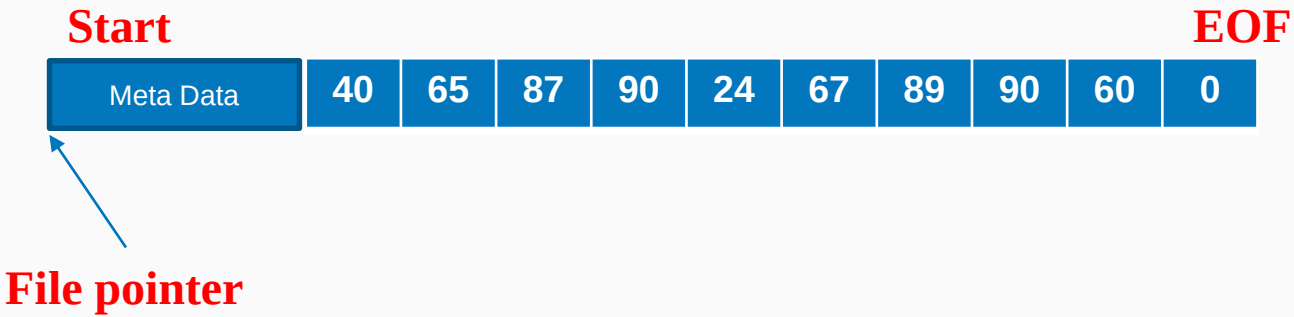
- A named collection of data, typically stored in a secondary storage (e.g., hard disk).
- Stored as sequence of bytes, logically contiguous (may not be physically contiguous on disk).

Examples:

- Records of all employees in an organization
- Document files created using Microsoft Word
- Video of a movie
- Audio of a music

- Non-volatile data storage
  - Can be used when power to computer is off

Introduction: How a File is Stored?



Note:

- The last byte of a file contains the end end-of-file character (EOF, with ASCII code 1A (Hex).
- While reading a file, the EOF character can be checked to know the end.

File Handling in C

Why file handling is needed in c program?

When problem involves large volume of data and in such situations the console oriented i/o operations pose two major problems.

- It becomes cumbersome and time consuming to handle large volume of data through terminals.
- The entire data is lost when either a program is terminated or the computer is turned off.

To solve these problems the concept of data file is introduced in which data can be stored on disks and read whenever necessary, without destroying data.

However, if we have a file containing all the data, we can easily access the contents of the file using few commands in C.

Introduction: Type of Files

I. **Text files:** Text file is a human readable sequence of characters and the words they form that can be encoded into computer-readable format such as ASCII. A text file is also known as ASCII file and can be read by any word processor. Text files stores information in consecutive characters. These characters can be interpreted as individual data items or as a component of strings or numbers.

II. **Binary files:** A binary files is made up of machine readable symbols that is made up of 0's and 1's.The binary file must be interpreted by the program that understand in advance exactly how it is formatted. Binary files are organized into blocks containing contiguous bytes of information. These blocks represents more complex data structures, such as array and structures.

- Text files
  - Contain ASCII code only
    - C-programs
- Binary files
  - Contain non-ASCII characters
    - Image, audio, video, executable, etc.

What type of file a .docx file produced by MS-Word?

Differences between text file and binary file.

Text File	Binary File
Text file is human readable because everything is stored in terms of text.	In binary file everything is written in terms of 0 and 1, therefore binary file is not human readable.
Every text file is terminated with EOF(End of File),whose ASCII value is 26.	There is no such special character present in the binary mode files to mark the end of file.
A newline (\n) character is converted into the carriage return-linefeed combination before being written to the disk.	In binary file, these conversions will not take place.
In text file, the text and characters are stored one character per byte. For example, the integer value 12345 will occupy 2 bytes in memory but it will occupy 5 bytes in text file.	In binary file, the integer value 12345 will occupy 2 bytes in memory as well as in file.
The extension for text file is .txt.	The extension for binary file is .dat.
File opening modes for text file are r, w, a, r+, w+ and a+.	File opening modes for binary file are rb,wb, ab, rb+,wb+ and ab+.

File Handling in C

Create a structure called student having name, age and rollno. Input a record store it in a file called student.txt and display the information.

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>

struct Student {
    char name[50];
    int age;
    int rollno;
};


int main() {
    struct Student s;

    FILE *fp;
    fp = fopen("student.txt", "w+");
    if (fp == NULL)
    {
        printf("File couldn't be opened");
        exit(0);
    }
```

File Handling in C

Create a structure called student having name, age and rollno. Input a record store it in a file called student.txt and display the information.

```
printf("Enter student name: ");
gets(s.name);

printf("Enter student age: ");
scanf("%d", &s.age);

printf("Enter student roll number: ");
scanf("%d", &s.rollno);

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

fclose(fp);

fp = fopen("student.txt", "r+");
if (fp == NULL) {
     printf("File couldn't be opened for reading\n");
        exit(1);
    }
```

File Handling in C

Create a structure called student having name, age and rollno. Input a record store it in a file called student.txt and display the information.

```
printf(" The information of student is \n");


fread(&s, sizeof(s), 1, fp);

printf("Name    : %s\n", s.name);
printf("Age     : %d\n", s.age);
printf("Roll No: %d\n", s.rollno);

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

Basic file operations in C

Basic file operations in C

The basic file operations in C are:

1. Naming a file
2. Opening a file
3. Reading data from file
4. Writing data to file and
5. Closing the file

Basic file operations in C

Opening and closing a file

- Before a program can write to a file or read from a file, the program must opened it.
- Opening a file establishes a link between program and the operating system .This provides the operating system the name of a file and the mode in which file is to be opened.
- While working with high level data file, we need buffer area where information is stored temporarily in the course of transferring data between computer memory and data file.
- The process of establishing connection between the program and the file is called opening a file.
- A structure named FILE is defined in the file stdio.h that contains all the information about file like name, status, buffer size, current position and of file status etc. A file pointer is a pointer to a structure of type FILE.
- Whenever a file is opened, a structure of type FILE associated with it and the file pointer that points to this structure identifies this file.

Basic file operations in C

The function **fopen()** is used to open the file. The buffer area is established by:

FILE *ptr_variable;

And file is opened by using the following syntax:

ptr_variable = fopen(“file_name”, ” file_opening_mode”);

Here, file_name is the name of the file we intend to open and opening mode specifies the purpose of opening file.

For example:

```
FILE *fptr1,*fptr2;
fptr1=fopen(“myfile.txt”,”w”);
fptr2=fopen(“testfile.dat”,”rb”);
```

The file that was opened using **fopen()** function must be closed when no more operations are to be performed on it. After closing the file the connection between file and the program is broken. On closing the file, all buffers associated with it are flushed and can be available for other files.

Basic file operations in C

Its syntax is:

fclose(ptr_variable);

for example:

fclose(fptr1);

fclose(fptr2);

Basic file operations in C

File Pointer

A file pointer is a pointer to a structure, which contains information about the file, including its name, current position of the file, whether the file is being read or written, and whether errors or end of the file have occurred. The user does not need to know the details, because the definitions obtained from stdio.h include a structure declaration called FILE. The only declaration needed for a file pointer is symbolized by

FILE *fptr;

This says that fptr is the file pointer that points to a FILE structure

Question: What is the significance of file pointer in file handling?

File opening modes

File opening modes

File opening modes specifies the way in which a file should be opened ln other words it specifies the purpose of opening a file. Let us discuss the file opening mode for text file.

File mode	Meaning of mode	During Inexistence of file
"r"	Opens an existing file for reading purpose only	If the file does not exist, fopen() returns NULL.
"w"	Opens a file for writing purpose	If the file exists, its contents are overwritten(contents are deleted first and written). If the file does not exist, it will be created.
"a"	Opens an existing file for appending purpose (i.e. addition of new content at the end of the existing file)	If the file does not exists, it will be created.
"r+"	Opens an existing file for both reading and writing purpose.	If the file does not exist, fopen() returns NULL.
"w+"	Opens file for reading and writing purpose.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
"a+"	Opens an existing file for both reading and appending purpose.	If the file does not exists, it will be created.

File opening modes

Note: The file opening modes in binary files are similar to text mode .Character b is added to each mode to indicate the binary mode. eg. rb, wb,ab,rb+,wb+,ab+.

E.g.

FILE *fptr;

fptr=fopen(“sample.txt”, r);

Here, file named sample.txt is opening for reading mode only.

Similarly,

FILE *fptr;

fptr=fopen(“hello.dat”,wb+);

Here, file named hello.dat is opening for both reading and writing.

Reading and writing data to a file

1. Character I/O functions

Character I/O functions

Using character I/O functions data can be read from a file or written to a file one character at a time.

fgetc(): It is used to read a character from a file.

Syntax: **char_variable=fgetc(file_ptr_variable);**

fputc(): It is used to write a character to file.

Syntax: **fputc(char_variable,file_ptr_variable);**

Character I/O functions

WAP to read a characters from keyboard and write character to a file.

OR

WAP to create a file named sample.txt and write some characters to a file until user hits the enter key.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
char ch;
FILE *fptr;
fptr=fopen("sample.txt","w");
if(fptr==NULL)
{
  printf("File cannot be openend");
  exit(1);
}
printf("Enter some characters");
while((ch=getchar())!='\n')
{
   fputc(ch,fptr);
}
fclose(fptr);
getch();
return 0;
}
```

Character I/O functions

WAP to open the file created in above example and read character stored in it and display it to the screen.

OR

WAP to read a character from the file name sample.txt and display it on screen.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
char ch;
FILE *fptr;
fptr=fopen("sample.txt","r");
printf("The character from file are\n");
if(fptr==NULL)
{
  printf("File cannot be openend");
  exit(1);
}
while((ch=fgetc(fptr))!=EOF)
{
  putchar(ch);
}
fclose(fptr);
getch();
return 0;
}
```

2. String Input/ Output functions

String Input/ Output functions

Using string I/O functions, data can be read from a file or written to a file in the form of array of characters.

fgets(): It is used to read a string from a file

syntax: fgets(string_variable,int_value,file_ptr_variable);

Here, int_value denotes the number of characters in string .The functions read a string from a file representing file_ptr_variable and stores in a variable string variable.

fputs():It is used to write a string to a file

syntax: fputs(string_variable,file_ptr_variable);

Here, contents in the string variable is written to a file representing a file_ptr_variable.

Character I/O functions

Write a program to create a file named “info.txt” and write “Welcome to Ambition” and display it.

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

int main() {
    FILE *fptr;
    fptr = fopen("info.txt", "w+");
    if (fptr == NULL) {
        printf("File cannot be opened.\n");
        exit(1);
    }

    fputs(“Welcome to Ambition", fptr);
    rewind(fptr);
    char ch;
    printf("Contents of the file:\n");
    while ((ch = fgetc(fptr)) != EOF) {
        putchar(ch);
    }

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

Character I/O functions

Write a program to create a file named “info.txt” and write “Welcome to Ambition” and display it.

(Alternative Solution)

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

int main() {
    FILE *fptr;
    char buffer[100]; // Buffer to hold the line

    fptr = fopen("info.txt", "w+");
    if (fptr == NULL) {
        printf("File cannot be opened.\n");
        exit(1);
    }
    fputs("Welcome to Ambition", fptr);
    rewind(fptr);
    printf("Contents of the file:\n");

    if (fgets(buffer, sizeof(buffer), fptr) != NULL) {
        printf("%s", buffer);  // Display the string
    }

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

Character I/O functions

WAP to write a string data to a file from the user.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>

int main()
{
FILE *fptr;
char str[50];
fptr=fopen("sample.txt","w");
if(fptr==NULL)
{
  printf("File cannot be opened");
  exit(1);
}
printf("Enter the strings");
while((strlen(gets(str))!=0))
{
  fputs(str,fptr);
}
fclose(fptr);
getch();
return 0;
}
```

3. Formatted Input /Output

Formatted Input /Output

These functions are used to read numbers, characters or string from a file or write them to a file in a format as per requirement.

fprintf(): It is formatted output function which is used to write integer, float ,char or string data to a file.

Syntax: fprintf(file_ptr_variable,"control_string",list_variables);

fscanf(): It is formatted input function which is used to read integer ,float, char or string data from a file.

Syntax: fscanf(file_ptr_variable,"format _specifier",&list_variables);

Formatted Input /Output

WAP to input age, gender, marks of a student in a file called “student.txt”. Now read these information from the file display them.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
char gender;
int age;
float marks;
FILE *fptr;
fptr=fopen("student.txt","w+");
if(fptr==NULL)
{
printf("File cannot be opened");
exit(1);
}
```

Formatted Input /Output

WAP to input age, gender, marks of a student in a file called “student.txt”. Now read these information from the file display them.

```
printf("Enter the gender\n");
scanf("%c",&gender);
printf("Enter the age\n");
scanf("%d",&age);
printf("Enter the marks\n") ;
scanf("%f",&marks);
fprintf(fptr,"%c\t%d\t%f",gender,age,marks);
rewind(fptr);
fscanf(fptr,"%c %d %f",&gender,&age,&marks);
printf("Gender=%c\tAge=%d\tMarks=%f",gender,age,marks);
fclose(fptr);
getch();
return 0;
}
```

Formatted Input /Output

WAP to open a new file and read name, address and telephone number of 10 employees from a user and write to a file.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
char name[20],address[20];
long int telno;
int i;
FILE *fptr;
fptr=fopen("employee.txt","w");
if(fptr==NULL)
{
printf("File cannot be opened");
exit(1);
}
for(i=0;i<10;i++)
{
printf("Enter the name,address and telno of employee
%d\n",i+1);
scanf("%s%s%ld",name,address,&telno);
fprintf(fptr,"%s\t%s\t%ld\n",name,address,telno);
}
fclose(fptr);
getch();
return 0;
}
```

4. Record I/O

4. Record I/O

Record i/o writes numbers to files in binary format.so that integers are stored in 2 bytes ,floating point numbers are stored in 4 bytes and so on.

- It permits reading or writing multiple data values in the form of arrays, structures and array of structures once.
- Thus the arrays, structures, array of structures etc. can be read from a file or written to a file as a single unit using record i/o.

There are functions fwrite and fread for writing and reading structure (record) in a file on a disk.

fwrite () used for writing entire block to a given file.

Syntax:fwrite(&ptr,size_of_array_or_structure,number_of_structure_or_array,fptr);

fread() is used to read an entire block from a given file.

Syntax: fread(&ptr, size_of_array_or_structure, number_of_structure_or_array, fptr);

4. Record I/O

where,

i) ptr is the address of an array or structure to be written

ii) Size_of_array_or_structure is an integer value that shows the size of structure or size of array which is being read or written.

iii) number_of_structure_or_array is an integer value that indicates number of arrays or structures to be written to file or read from file.

iv) fptr is a file pointer of a file opened in binary mode.

4. Record I/O

Illustration of fwrite( ) and fread( )

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct student
{
int roll;
char name[25];
float marks;
};
int main()
{
FILE *fptr;
char ch;
struct student st;
fptr = fopen("test.dat","wb+");
if(fptr == NULL)
{
printf("File cannot be opened");
exit(1);
}
```

4. Record I/O

Illustration of fwrite( ) and fread( )

```
do
{
printf("Enter the Rollno:\n");
scanf("%d",&st.roll);
printf("Enter the Name:\n");
scanf("%s",st.name);
printf("Enter the Marks:\n");
scanf("%f",&st.marks);
fwrite(&st,sizeof(st),1,fptr);
printf("Do you want to add another data (y/n):\n");
ch = getche();
}while(ch=='y' || ch=='Y');
printf("Data written successfully\n");
rewind(fptr);
printf("\nRoll\tName\tMarks\n");
while(fread(&st,sizeof(st),1,fptr)==1)
{
printf("\n%d\t%s\t%f",st.roll,st.name,st.marks);
}
fclose(fptr);
getch();
return 0;
}
```

Write a program to create structure for the following data for student (rollno, name, phone, address and semester).Read the 10 students by user and write only those students whose semester is 1 in file “student.txt”.

Write a program to create structure for the following data for student (rollno, name, phone, address and semester).Read the 10 students by user and write only those students whose semester is 1 in file “student.txt”.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct student
{
char name[20];
int rollno;
long int phone;
char address[20];
int semester;
};

int main()
{
int i;
struct student st[10];
FILE *fptr;
fptr=fopen("student.txt","w+");
if(fptr==NULL)
{
printf("File cannot be opened");
exit(1);
}
```

Write a program to create structure for the following data for student (rollno, Name, phone, address and semester).Read the 10 students by user and write only those students whose semester is 1 in file “student.txt”.

```
printf("Enter records of 10 students\n");
for(i=0;i<10;i++)
{
printf("Enter the name\n");
gets(st[i].name);
printf("Enter rollno\n");
scanf("%d",&st[i].rollno);
printf("Enter Phone number\n");
scanf("%ld",&st[i].phone);
printf("Enter the Address\n");
gets(st[i].address);
printf("Enter semester\n");
scanf("%d",&st[i].semester);
if(st[i].semester==1)
{
            fwrite(&st[i],sizeof(st[i]),1,fptr);
}
}
printf("Data written successfully");
fclose(fptr);
getch();
return 0;
}
```

Write a program to input name, address, faculty, program and GPA(in maximum 4.0) of 500 students and store them in ‘RESULT.DAT’ data file and display the records of those student whose faculty is ‘Engineering’ and GPA>3.5

Write a program to input name, address, faculty, program and GPA(in maximum 4.0) of 500 students and store them in ‘RESULT.DAT’ data file and display the records of those student whose faculty is ‘Engineering’ and GPA>3.5

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct student
{
char name[20];
char address[20];
char faculty[20];
char program[20];
float gpa;
};

int main()
{
struct student st[500];
int i;
FILE *fptr;
fptr=fopen("result.dat","wb+");
if(fptr==NULL)
{
 printf("File cannot be opened");
 exit(1);
}
```

Write a program to input name, address, faculty, program and GPA(in maximum 4.0) of 500 students and store them in ‘RESULT.DAT’ data file and display the records of those student whose faculty is ‘Engineering’ and GPA>3.5

```
printf("Enter records of 500 students");
for(i=0;i<500;i++)
{
printf("Enter the records of student
%d\n",i+1);

printf("Enter student name\n");
gets(st[i].name);

printf("Enter the address\n");
gets(st[i].address);

printf("Enter the faculty\n");
gets(st[i].faculty);

printf("Enter the Program\n");
gets(st[i].program);
```

Write a program to input name, address, faculty, program and GPA(in maximum 4.0) of 500 students and store them in ‘RESULT.DAT’ data file and display the records of those student whose faculty is ‘Engineering’ and GPA>3.5

```
do
{
   printf("Enter the gpa\n");
   scanf("%f",&st[i].gpa);
}while(st[i].gpa>4);
}

fwrite(&st,sizeof(st),500,fptr);
rewind(fptr);
printf("Information student of faculty Engineerig with GPA
greater than 3.5 are:\n");
printf("Name\tAddress\tFaculty\tProgram\tGPA\n");
fread(&st,sizeof(st),500,fptr);

for(i=0;i<500;i++)
{
if(st[i].gpa>3.5&&strcmp(st[i].faculty,"Engineering")==0)
{
printf("\n%s\t%s\t%s\t%s\t%f",st[i].name,st[i].address,st[i].
faculty,st[i].program,st[i].gpa);
}
}
getch();
return 0;
}
```

Write a program to read the name ,author and price of 500 books in a library from the file “library.dat”.Now print the name and price of those books whose price is above Rs.300.

Write a program to read the name ,author and price of 500 books in a library from the file "library.dat".Now print the name and price of those books whose price is above Rs.300.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct book
{
char name[20];
char author[20];
float price;
};
int main()
{

int i;
struct book b[500];
FILE *fptr;
fptr=fopen("library.dat","rb");
if(fptr==NULL)
{
  printf("file cannot be opened");
  exit(1);
}
```

Write a program to read the name ,author and price of 500 books in a library from the file “library.dat”. Now print the name and price of those books whose price is above Rs.300.

```
printf("Book which price is above 300 are");
printf("\nName\tprice");
fread(&b,sizeof(b),500,fptr);
for(i=0;i<500;i++)
{
  if(b[i].price>300)
  {
    printf("\n%s\t%f",b[i].name,b[i].price);
  }
}
fclose(fptr);
getch();
return 0;
}
```

Error handling in files

Error handling in files

- When working with files, it's crucial to check whether file operations like fopen(), fread(), fwrite(), etc. succeed. If not handled properly, they can crash your program.

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

int main() {
    FILE *fp;

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

    if (fp == NULL) {
        printf("Error opening file");
        exit(1);  // Exit with failure
    }

    // File operations go here...
    fclose(fp);
    return 0;
}
```

Note:

- fopen() returns NULL on failure.
- printf("message") prints the system error message.
- exit(1) exits the program if the file can't be opened.

Questions:

1. Differentiate between binary files and text files with suitable examples.
2. List various file operations supported in C programming.
3. Explain input and output operations on files using fprintf() and fscanf() with syntax.
4. List down the different file opening modes in C.
5. Write a program to input name, address, registration no, faculty and academic year of admission in university of ‘n’ number of students of AB University and append them in data file called ‘STUDENT.DAT’. Then display the records of those students by reading the records from ‘STUDENT.DAT’ data file who got admission in 2016.
6. Create a structure called goods that stores number, price, purchase date and quantity. WAP to store information of 100 goods in the file “good.dat”.
7. Write a C program to create a text file, write some content into it, and then read and display it using file operations (fopen(), fprintf(), fscanf(), fclose()).

THANK YOU

Any Queries ?