Unit 10: Data File Handling

Er.Nipun Thapa

File Handling in C

- In programming, we may require some specific input data to be generated several numbers of times. Sometimes, it is not enough to only display the data on the console.
- The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again. However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling in C.

File Handling in C

- File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program. The following operations can be performed on a file.
  - Creation of the new file
  - Opening an existing file
  - Reading from the file
  - Writing to the file
  - Deleting the file

Different Types of Files in C

- When dealing with files, there are two types of files you should know about:
  - Text files
  - Binary files

Different Types of Files in C

1. Text files

- Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.
- When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.
- They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

Different Types of Files in C

2. Binary files

- Binary files are mostly the **.bin** files in your computer.
- Instead of storing data in plain text, they store it in the binary form (0's and 1's).
- They can hold a higher amount of data, are not readable easily, and provides better security than text files.

C File Handling Operations

- In C, you can perform four major operations on files, either text or binary:
  - Creating a new file
  - Opening an existing file
  - Closing a file
  - Reading from and writing information to a file

C File Handling Operations

- Creation of a new file (**fopen with attributes as “a” or “a+” or “w” or “w++”)**
- Opening an existing file (**fopen**)
- Reading from file (**fscanf or fgets**)
- Writing to a file (**fprintf or fputs**)
- Moving to a specific location in a file (**fseek, rewind**)
- Closing a file (**fclose**)

Functions in File Operations:

File operation	Declaration & Description
fopen() – To open a file	Declaration: FILE *fopen (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=fopen ("filename", "'mode"); Where, fp – file pointer to the data type "FILE". filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.
fclose() – To close a file	Declaration: int fclose(FILE *fp); fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below. fclose (fp);
fgets() – To read a file	Declaration: char *fgets(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. fgets (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer
fprintf() – To write into a file	Declaration: int fprintf(FILE *fp, const char *format, ...);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or fprintf (fp, "text %d", variable_name);

Functions for file handling

No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

Opening or creating file

For opening a file, **fopen** function is used with the required access modes. Some of the commonly used file access modes are mentioned below.

File opening modes in C:

- **"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.
- **"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.
- **"r+" –** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- **"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.
- **"a+" –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which 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.

Opening or creating file

Syntax :

```
FILE *fptr;
ptr = fopen("fileopen","mode");
```

Example:

```
FILE *filePointer;
```

So, the file can be opened as

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

LAB 9. QN1: Program to Open a File, Write in it, And Close the File

```
# include <stdio.h>
# include <string.h>
int main( )
{

   FILE *filePointer ;
   filePointer = fopen(“ncc.c", "w") ;
   if ( filePointer == NULL )
   {
      printf( “ncc.c file failed to open." ) ;
   }
   else
   {
      printf("The file is now opened.\n") ;
   }
   fputs("We love ncc",filePointer);
      fclose(filePointer) ;
   return 0;
}
```

LAB 9. QN2 : Program to Open a File, Read from it, And Close the File

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

int main( )
{
   FILE *filePointer ;
   char dataToBeRead[50];
  // in read mode using "r" attribute
   filePointer = fopen("oic.c", "r") ;
   if ( filePointer == NULL )
   {
      printf( "oic.c file failed to open." ) ;
   }
```

```
else
   {
      printf("The file is now opened.\n") ;
      // Read the dataToBeRead from the file
      // using fgets() method
      while( fgets ( dataToBeRead, 50, filePointer ) != NULL )
      {
         // Print the dataToBeRead
         printf( "%s" , dataToBeRead ) ;
       }
      // Closing the file using fclose()
   }
      fclose(filePointer) ;

   return 0;
}
```

LAB 9. QN 3 : Read from a text file

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num;
  FILE *fptr;
  if ((fptr = fopen("oic.txt","r")) == NULL){
     printf("Error! opening file");
     // Program exits if the file pointer returns NULL.
     exit(1);
  }

  fscanf(fptr,"%d", &num);

  printf("Value of n=%d", num);
  fclose(fptr);

  return 0;
}
```

4. C program to read name and marks of n number of students and store them in a file.

```
#include <stdio.h>
int main()
{
  char name[50];
  int marks, i, num;

  printf("Enter number of students: ");
  scanf("%d", &num);

  FILE *fptr;
  fptr = (fopen("D:\\student.txt", "w"));
  if(fptr == NULL)
  {
     printf("Error!");
     exit(1);
  }

 for(i = 0; i < num; ++i)
  {
    printf("For student%d\nEnter name: ", i+1);
    scanf("%s", name);

    printf("Enter marks: ");
    scanf("%d", &marks);

    fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
  }

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

5. C program to read name and marks of n number of students from and store them in a file. If the file previously exits, add the information to the file.

```
#include <stdio.h>
int main()
{
  char name[50];
  int marks, i, num;

  printf("Enter number of students: ");
  scanf("%d", &num);

  FILE *fptr;
  fptr = (fopen("D:\\student.txt", “a"));
  if(fptr == NULL)
  {
     printf("Error!");
     exit(1);
  }

 for(i = 0; i < num; ++i)
  {
    printf("For student%d\nEnter name: ", i+1);
    scanf("%s", name);

    printf("Enter marks: ");
    scanf("%d", &marks);

    fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
  }

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

6. C program to write all the members of an array of structures to a file using fwrite(). Read the array from the file and display on the screen.

```
#include <stdio.h>
struct student
{
  char name[50];
  int height;
};
int main(){
   struct student stud1[5], stud2[5];
   FILE *fptr;
   int i;

   fptr = fopen("D:\\file.txt","wb");
   for(i = 0; i < 5; ++i)
   {
      fflush(stdin);
      printf("Enter name: ");
      gets(stud1[i].name);
      printf("Enter height: ");
      scanf("%d", &stud1[i].height);
   }

   fwrite(stud1, sizeof(stud1), 1, fptr);
   fclose(fptr);

   fptr = fopen("D:\\file.txt", "rb");
   fread(stud2, sizeof(stud2), 1, fptr);
   for(i = 0; i < 5; ++i)
   {
      printf("Name: %s\nHeight: %d", stud2[i].name,
stud2[i].height);
   }
   fclose(fptr);
}
```

C Source Code: Reading File Contents

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

int main()
{
 FILE *fptr;
 char ch;

 /* Opening file in read mode */
 fptr = fopen("D:\\student.txt","r");
 if(fptr==NULL)
 {
  printf("Can't open file. Make sure file exits.\n");
  exit(1);
 }
```

```
do
{
        ch = fgetc(fptr);
        putchar(ch);

}while(ch!=EOF);
fclose(fptr);

return 0;
}
```

End-Of-File (EOF)

- EOF is special character that indicates the the end of file has been reached.
- This character can be generated from the keyboard by typing ctrl+z;
- When we are creating a file, the special character EOF , is inserted after the last character of the file by the OS.
- Thus, the last point of file is detected using EOF while reading data from file.
- It’s necessity arises from the fact that we may not know in advance up to where we have data

Read file using End-Of-File (EOF) Example

```
#include<stdio.h>
void main( )
{
    FILE *fp ;
    char ch ;
    fp = fopen("D:\\student.txt","r") ;
    while ( 1 )
    {
        ch = fgetc ( fp ) ;
        if ( ch == EOF )
        break ;
        printf("%c",ch) ;
    }
    fclose (fp ) ;
}
```

Example 10: Define a structure for Vehicle Owner having data members name, address, telephone number, vehicle number and license number. Take the data for ten owners, write them in file "Own.txt". Read the data from the file and display them.

```
#include <stdio.h>
#define SIZE 10
void main()
{
      FILE *fp;
      struct vehicle_owner
      {
      char name[20];
      char address[20];
      long int phone_no;
      int vehicle_no;
      int license_no;
      };
      struct vehicle_owner vehicle[SIZE], v[SIZE];
      int i;
      clrscr();
      fp=fopen("C:\\Own.txt","w");
      if(fp==NULL)
      {
      printf("\nCannot create file.");
      exit();
      }
      for(i=0;i<SIZE;i++)
      {
      printf("\n Enter information about vehicle owner %d",i+1);
      printf("\n Enter name :\t");
```

```
        gets(vehicle[i].name);
        printf("\n Enter address:\t");
        gets(vehicle[i].address);
        printf("\n Enter telephone no:\t");
        scanf("%ld", &vehicle[i].phone_no);
        printf("\n Enter vehicle no:\t");
        scanf("%d", &vehicle[i].vehicle_no);
        printf("\n Enter license no:\t");
        scanf("%d", &vehicle[i].license_no);
        fprintf(fp,"%s\t%s\t%ld\t%d\t%d\n",vehicle[i].name,
        vehicle[i].address, vehicle[i].phone_no, vehicle[i].vehicle_no,
        vehicle[i].license_no);
        fflush(stdin);
        }
        fclose(fp);
        fp=fopen("C:\\Own.txt","r");
        for(i=0;i<SIZE;i++)
        {
        fscanf(fp,"%s %s %ld %d %d",&v[i].name,&v[i].address,&v[i]
        .phone_no,&v[i].vehicle_no,&v[i].license_no);
        printf("%s\t%s\t%ld\t%d\t%d\n",v[i].name,v[i].address,v[i].phone
        _no,v[i].vehicle_no,v[i].license_no);
        }
        fclose(fp);
getch();
}
```

the following words

Example 11: Given a text file, create another text file deleting the following words "three "bad", and "time".

```
#include <stdio.h>
void main()
{
      FILE *fp,*fpp;
      char c[10];
      fp=fopen("C:\\test.txt",“r");
      clrscr();
      if(fp==NULL)
      {
      printf("Cannot open file");
      exit();
      }
```

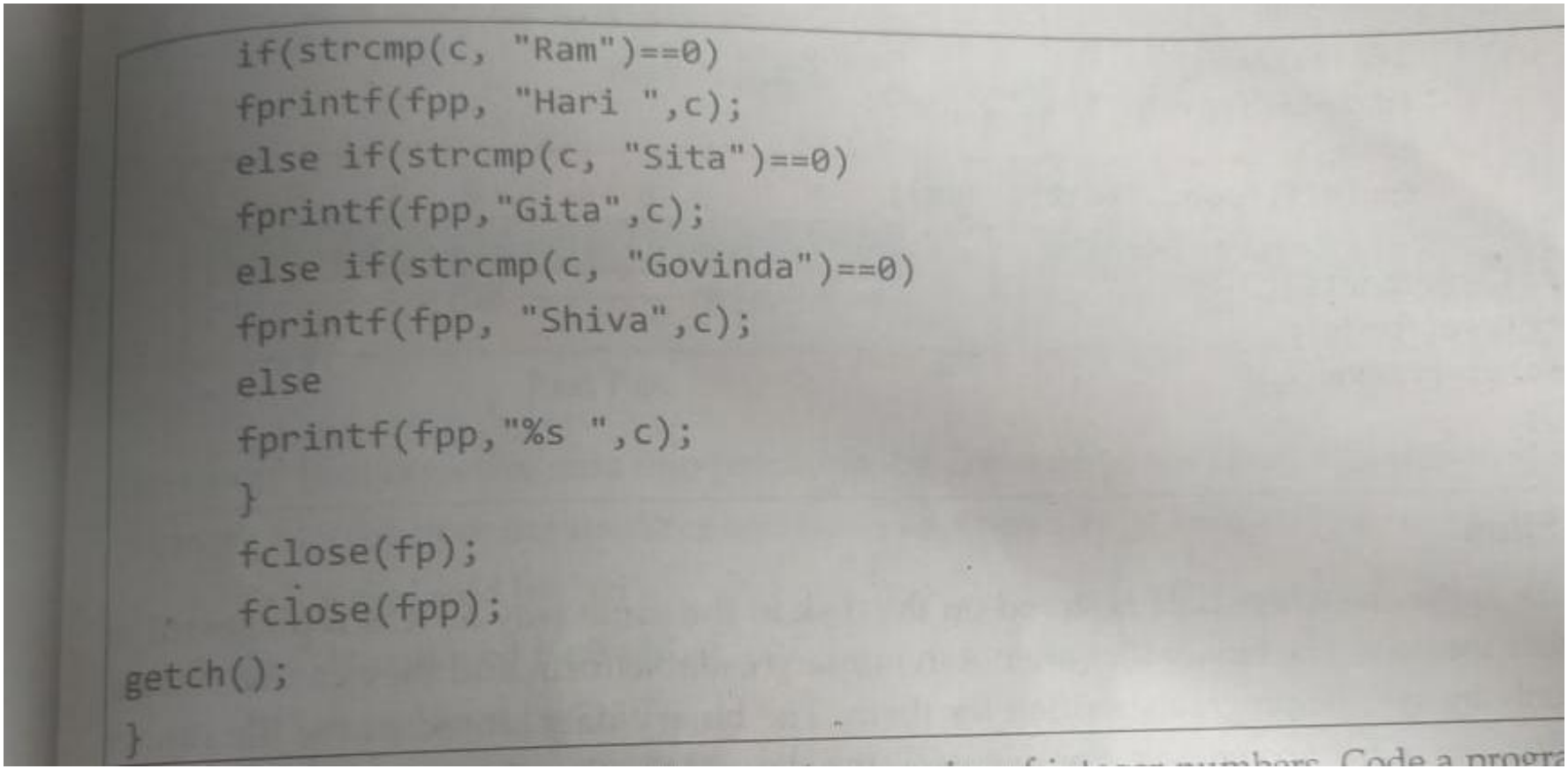
```
        fpp=fopen("C:\\hello.txt","w");
        if(fpp==NULL)
        {
        printf("Cannot create file");
        exit();
        }
        while(fscanf(fp,"%s",&c)!=EOF)
        {
        if((strcmp(c,"three")!=0)&&(strcmp(c,"bad")!=0)&&(strcmp(c,"time
        ")!=0))
        {
        fprintf(fpp,"%s ",c);
        }
        }
        fclose(fp);
        fclose(fpp);
getch();
}
```

Some text file is given, create another text file replacing the following words

```
getch();
}
```

Example 12: Some text file is given, create another text file replacing the following words "Ram" to "Hari", "Sita" to "Gita", and "Govinda" to "Shiva".

```
#include <stdio.h>
void main()
{
      FILE *fp,*fpp;
      char c[10];
      fp=fopen("C:\\test.txt","r");
      clrscr();
      if(fp==NULL)
      {
      printf("Cannot open file");
      exit();
      }
      fpp=fopen("C:\\hello.txt","w");
      if(fpp==NULL)
      {
      printf("Cannot create file");
      exit();
      }
      while(fscanf(fp,"%s",&c)!=EOF)
      {
```



```
        if(strcmp(c, "Ram")==0)
        fprintf(fpp, "Hari ",c);
        else if(strcmp(c, "Sita")==0)
        fprintf(fpp,"Gita",c);
        else if(strcmp(c, "Govinda")==0)
        fprintf(fpp, "Shiva",c);
        else
        fprintf(fpp,"%s ",c);
        }
        fclose(fp);
        fclose(fpp);
getch();
}
```

Finished Unit 10