Practical-11

To study about operations of file handling and Management

Practical-11.1

- Write a program to create a file and perform basic I/O operations on the same

File Management

- A File can be used to store a large volume of persistent data.
- File management functions
    1. Creation of a file
    2. Opening a file
    3. Reading a file
    4. Writing to a file
    5. Closing a file

How to Create a File

- A file is nothing but space in a memory where data is stored.
- Syntax:
  - FILE *fp;
  - fp = fopen (“d:\\p1.txt", “r");

fopen() Function

- fopen is a standard function which is used to open a file
- If the file is not present on the system, then it is created and then opened.
- If a file is already present on the system, then it is directly opened using this function
- fp is a file pointer which points to the type file

File opening Mode

Mode	Meaning
r	Open text file in read mode • If file exists, the marker is positioned at beginning. • If file doesn't exist, error returned.
w	Open text file in write mode • If file exists, it is erased. • If file doesn't exist, it is created.
a	Open text file in append mode • If file exists, the marker is positioned at end. • If file doesn't exist, it is created.

Example

- **Mode = “r”** – open for reading, this mode will open the file for reading purpose only
- This mode cannot create a new file and **open() returns NULL**

```
FILE * file;
if (file = fopen("hello.txt", "r"))
     {
 printf("File opened in read mode");
     }
 else
   {
  printf("The file is not present!”);
  Printf(“cannot create a new file using
r mode");
  fclose(file);
    }
```

Example

- **Mode = “w”** – open for writing only, this mode will open the file if present in the current directory for writing only
- If the file is not present in the current directory, the program will create a new file and open it for writing.
- If file is present and that contains some text in it, the contents will be overwritten.

```
FILE * file;
if (file = fopen("hello.txt", “w"))
     {
 printf("File opened in write mode
           or new file created");
     }
 else
   {
  printf("The file is not present!”);
   fclose(file);
    }
```

Example

- **Mode = “a”** – open for append only, this mode will open the file if present in the current directory for writing only
- If the file is not present in the current directory, the program will create a new file and open it for writing.
- If we open a file that contains some text in it, the contents will not be overwritten.

```
FILE * file;
if (file = fopen("hello.txt", “a"))
      {
printf("File opened in append mode or
new file created");
      }
 else
   {
printf("The file is not present!”);
fclose(file);
     }
```

Example- r+ Mode

- **Mode = “r+”** – open for reading and writing both, this mode will open the file for both reading and writing purposes
- This mode cannot create a new file and **open() returns NULL**, if we try to create a new file using this mode.
- If we open a file that contains some text in it and write something, the contents will be overwritten.

Example

```
FILE * file;
if (file = fopen("hello.txt", "r+"))
{
 printf("File opened successfully in read and write
both");
  }
else
printf("The file is not present! cannot create a new
file using r+ mode");
fclose(file);
```

W+ Mode

- **Mode = “w+”** – open for writing and reading, this mode will open the file if present in the current directory for writing and reading operation both.
- If the file is not present in the current directory, the program will create a new file and open it for reading and writing.
- If we open a file that contains some text in it, the contents will be overwritten.

a+ Mode

- Mode = “a+” − open for read and append, this mode will open the file if present in the current directory for both reading and writing.
- If the file is not present in the current directory, the program will create a new file and open it for reading and writing.
- If we open a file that contains some text in it, the contents will not be overwritten;
- instead, the new text will be added after the existing text in the file.

Practical-11.1

- **Write a program to create a file** and perform basic I/O operations on the same.
- Start
- Declare file pointer
- Open file in read mode and close
- Open file in write and close file
- Open file in append mode and Close file
- End

Reading Data from an existing file

- We can read content of a file in c using the fscanf() and fgets() and fgetc() functions

fscanf()

- **fscanf()**
- The fscanf() function is used to read character set i.e strings from the file.
- It returns the EOF, when all the content of the file are read by it.
- **Syntax**

```
int fscanf(FILE *stream, const char *charPointer[])
```

- FILE *stream: the pointer to the opened file.
- const char *charPointer[]: string of character.

Example

```
FILE * fp;
char str[500];
file = fopen(“d:\\hello.txt","r"))
while(fscanf(fp,"%s",str)!=EOF)
        {
            printf("%s",str);
        }
 else     printf("Error!”);
 fclose(fp);
```

OUTPUT:
GanpatUniversity

fgets()

- The fget() function in C is used to read string from the stream.
- Syntax

```
char* fgets(char *string, int length, FILE *stream)
```

- Parameter
- char *string: It is a string which will store the data from the string.
- int length: It is an int which gives the length of string to be considered.
- FILE *stream: It is the pointer to the opened file.

Example

```
FILE * file;

char str[500];

file = fopen(“d:\\hello.txt","r"))

printf("%s", fgets(str, 50,file));

fclose(file);
```

OUTPUT:
Ganpat University

fgetc()

The fgetc() function in C is used to return a single character from the file.

It gets a character from the file and returns EOF at the end of file.

Syntax

```
char* fgetc(FILE *stream)
```

Parameter

`FILE *stream: It is the pointer to the opened file.`

Example

```
FILE * fp;
char str;
file = fopen(“d:\\hello.txt", "r"))
while((str=fgetc(file))!=EOF)
 {
     printf("%c",str);
 }
fclose(file);
```

OUTPUT:
Ganpat University

Writing Data to a file in C

- We can write data to a file in C using the fprintf(), fputs(), fputc() functions.
- All are used to write contents to a file.

fprintf()

- The fprintf() function is used to write data to a file. It writes a set of characters in a file.
- Syntax
- `int fprintf(FILE *stream, char *string[])`
- Parameters
- FILE for *stream: It is the pointer to the opened file.
- char *string[]: It is the character array that we want to write in the file.

Example

```
FILE *fp;
   fp = fopen("d:\\file.txt", "w");
 //opening file
   fprintf(fp, “Ganpat University\n");
//writing data into file
   fclose(fp);
```

fputc()

- The fputc() function is used to write a single character to the file.
- Syntax
- `int fputc(char ch , FILE *stream)`
- Parameters
- char character : It is the character that we want to write in the file.

  FILE for *stream: It is the pointer to the opened file.

Example

```
FILE * file;

file = fopen(“d:\\hello.txt", "w“)

fputc('T', file);

fclose(file);
```

fputs()

- The fputs() function in C can be used to write to a file. It is used to write a line (character line) to the file.
- Syntax
- `int fputs(const char *string, FILE *stream)`
- Parameters
- Constant char *string[]: It is the character array that we want to write in the file.
- FILE for *stream: It is the pointer to the opened file.

Example

- FILE *fp;
- clrscr();
- fp=fopen("d:\\hello.txt","w");
- fputs("hello c programming",fp);
- fclose(fp);

Example: write into file

```
FILE *fp;

char s1[20]=“Ganpat university”;

fp=fopen(“p1.txt”,w);

If(fp==NULL)

{

printf(“File does not exist”);

 }

else

Printf(“File is open”);

If(strlen(s1)>0)

 {

    fputs(s1,fp);

    fputs(“\n”,fp);

}

fclose(fp);
```

Example-Input and output operation on file

```
FILE *fp; char ch;

fp = fopen(“d:\\file.txt", "w");

printf("Enter data...");

while( (ch = getchar()) != EOF)

 {

      putc(ch, fp);

  }

fclose(fp);
```

```
fp = fopen(" d:\\file.txt ", "r");

 while( (ch = getc(fp)! = EOF)

 printf("%c",ch);

// closing the file pointer

 fclose(fp);
```

Practical-11.1

- Write a program to create a file and perform basic I/O operations on the same.

Practical-11.2

- Write a program to illustrate the use of fputc(), fputs(), and fgets()