Programming in C (CS-1001)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

3 Credit

Mr. Rajat Kumar Behera - Associate Professor

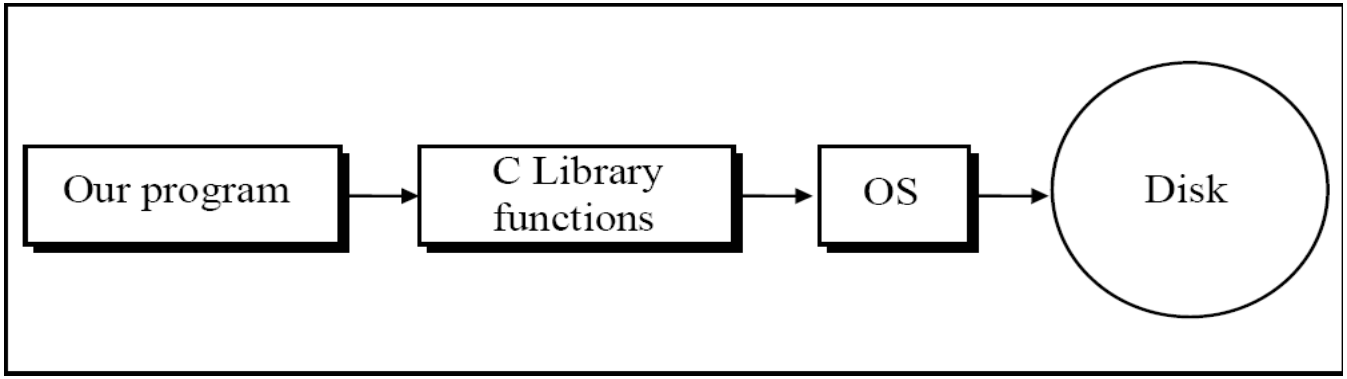
Course Contents

Sr #	Major and Detailed Coverage Area	Hrs
11	**File Handling**	2
	File operations - opening, closing, reading, writing.	

Introduction

Often it is not enough to just display the data on the screen. This is because if the data is large, only a limited amount of it can be stored in memory and only a limited amount of it can be displayed on the screen. It would be inappropriate to store this data in memory for one more reason. Memory is volatile and its contents would be lost once the program is terminated. So if we need the same data again it would have to be either entered through the keyboard again or would have to be regenerated programmatically. Obviously both these operations would be tedious. At such times it becomes necessary to store the data in a manner that can be later retrieved and displayed either in part or in whole. This medium is usually a 'file' on the disk.

Data Organization



File Operation and I/O Functions

There are different operations that can be carried out on a file. These are:

- Creation of a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Moving to a specific location in a file (seeking)
- Closing a file

There are 2 distinct ways to perform file operations in C. The first one is known as the low-level I/O and uses OS level system calls. The second method is referred as the high-level I/O operation and uses function in C's standard I/O library.

Sr #	Function Name	Operation
1	fopen()	Creates a new file or open an existing file for use
2	fclose()	Closes the file which has been opened for use
3	fgetc()	Reads a character from a file
4	fputc()	Writes a character to a file

File I/O Functions

Sr #	Function Name	Operation
5	fprintf()	Writes a set of values to a file
6	fscanf()	Reads a set of data values from a file
7	fseek()	Sets the position to a desired point in the file
8	ftell()	Gives the current position in the file (in terms of bytes from the start)
		position to the beginning of the file

Defining and Opening a File

If we wanted to store data in a file in the secondary memory, we must specify certain things about the file to the operating system. They include the following:

- File Name
- Data Structure
- Purpose

File name is a string that make up a valid file name for the operating system. It may contain two parts, a primary name and an optional period with the extensions. Examples: **input.data**, **student.c** or **text.out**

Defining and Opening a File

Data structure of a file is defined as **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.

When we open a file, we must specify what we want to do with the file. For example, we may write the data to the file or read the already existing data. So the general format for declaring and opening a file:

FILE *fp;
fp = **fopen**("filename", "mode");

The 1st statement declares the variable fp as a "pointer to the data type **FILE**". **FILE** is a structure defined in the I/O library. The second statement opens the file named filename and assigns an identifier to the **FILE** type pointer **fp**. This pointer is which contains all the information about the file is subsequently used as communication link between the system and the program.

The second statement also specifies the **purpose** of opening the file. The mode does the job.

File Mode and Description

Sr #	Mode	Description
1	r	Opens an existing text file for reading purpose
2	f	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
3	a	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
4	r+	Opens a text file for both reading and writing
5	w+	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
		both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

Examples

- FILE *fp1 = fopen("data.c","r");
- FILE *fp2 = fopen("data.c","a");
- FILE *fp3 = fopen("data.c","w+");
- FILE *fp4 = fopen("data.c","r+");
- FILE *fp5 = fopen("data.c","a+");
- FILE *fp6 = fopen("data.c","f");

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 is broken. To close a file, use the fclose( ) function. The prototype of this function is –

```
int fclose( FILE *fp );
```

The **fclose**() function returns zero on success, or EOF if there is an error in closing the file. This function actually flushes any data still pending in the buffer to the file, closes the file, and releases any memory used for the file. The EOF is a constant defined in the header file stdio.h.

Example 1

```
FILE *fp1;
fp1 = fopen("data.c","r");
.....

fclose(fp1);
```

Example 1

```
FILE *fp2;
fp1 = fopen("OutputData.dat","a");
.....

fclose(fp2);
```

Display contents of a file on screen

Code snippet

```
FILE *fp ;
char ch ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
  ch = fgetc ( fp ) ;
  if ( ch == EOF )
    break ;
  printf ( "%c", ch ) ;
}
fclose ( fp ) ;
```

Note

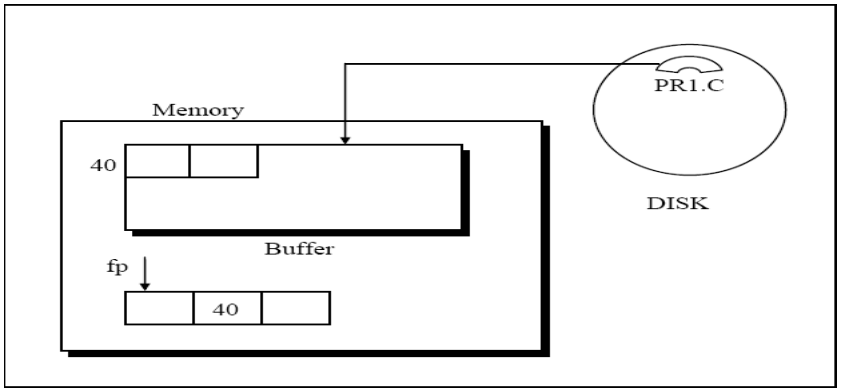
fgetc() reads the character from the current pointer position, advances the pointer position so that it now points to the next character, and returns the character that is read, which we collected in the variable ch.

Code snippet Explanation

fopen( ) performs three important tasks when you open the file in “r” mode:

- Firstly it searches on the disk the file to be opened.
- Then it loads the file from the disk into a place in memory called buffer.
- It sets up a character pointer that points to the first character of the buffer.

Diagrammatic Representation



Count chars, spaces, tabs and newlines in a file


```
FILE *fp ;
char ch ;
int nol = 0, not = 0, nob = 0, noc = 0 ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
  ch = fgetc ( fp ) ;
  if ( ch == EOF )
    break ;
  noc++ ;
 if ( ch == ' ' )
   nob++ ;
 if ( ch == '\n' )
   nol++ ;
 if ( ch == '\t' )
    not++ ;
}
fclose ( fp ) ;
```

```
//continuation of program
printf ( "\nNumber of characters = %d", noc ) ;
printf ( "\nNumber of blanks = %d", nob ) ;
printf ( "\nNumber of tabs = %d", not ) ;
printf ( "\nNumber of lines = %d", nol ) ;
```

Trouble in Opening a File

There is a possibility that when we try to open a file using the function fopen( ), the file may not be opened. While opening the file in “r” mode, this may happen because the file being opened may not be present on the disk at all. And you obviously cannot read a file that doesn’t exist. Similarly, while opening the file for writing, fopen( ) may fail due to a number of reasons, like, disk space may be insufficient to open a new file, or the disk may be write protected or the disk is damaged and so on. Crux of the matter is that it is important for any program that accesses disk files to check whether a file has been opened successfully before trying to read or write to the file. If the file opening fails due to any of the several reasons mentioned above, the fopen( ) function returns a value **NULL** (defined in “stdio.h” as #define NULL 0). This can be handled in the program as follows…

```
FILE *fp ;
fp = fopen ( "PR1.C", "r" ) ;
if ( fp == NULL )
{
  puts ( "cannot open file" ) ;
  exit(0);
}
```

Writing to a File


```
FILE *fp ;
char s[80];
fp = fopen ( "POEM.TXT", "w" ) ;
if ( fp == NULL )
{
  puts ( "Cannot open file" ) ;
  exit( ) ;
}
printf ( "\nEnter a few lines of text:\n" ) ;
while ( strlen ( gets ( s ) ) > 0 )
{
  fputs ( s, fp ) ;
  fputs ( "\n", fp ) ;
}
fclose ( fp ) ;
```

Note

- Each string is terminated by hitting enter. To terminate the execution of the program, hit enter at the beginning of a line. This creates a string of zero length, which the program recognizes as the signal to close the file and exit.
- We have set up a character array to receive the string; the fputs( ) function then writes the contents of the array to the disk. Since fputs( ) does not automatically add a newline character to the end of the string, we must do this explicitly to make it easier to read the string back from the file.

A File-copy Program


```
FILE *fs, *ft ;
char ch ;
fs = fopen ( "pr1.c", "r" ) ;
if ( fs == NULL )
{
 puts ( "Cannot open source file" ) ;
 exit( ) ;
}
ft = fopen ( "pr2.c", "w" ) ;
if ( ft == NULL )
{
  puts ( "Cannot open target file" ) ;
  fclose ( fs ) ;
  exit( ) ;
}
```

```
while ( 1 )
{
  ch = fgetc ( fs ) ;
  if ( ch == EOF )
    break ;
  else
    fputc ( ch, ft ) ;
}
fclose ( fs ) ;
fclose ( ft ) ;
```

A File-append Program


```
FILE *fs, *ft ;
char ch ;
fs = fopen ( "pr1.c", "r" ) ;
if ( fs == NULL )
{
 puts ( "Cannot open source file" ) ;
 exit( ) ;
}
ft = fopen ( "pr2.c", “a" ) ;
if ( ft == NULL )
{
  puts ( "Cannot open target file" ) ;
  fclose ( fs ) ;
  exit( ) ;
}
```

```
while ( 1 )
{
  ch = fgetc ( fs ) ;
  if ( ch == EOF )
    break ;
  else
    fputc ( ch, ft ) ;
}
fclose ( fs ) ;
fclose ( ft ) ;
```

Writing record to a file


```
FILE *fp ;
char another = 'Y' ;
struct emp
{
   char name[40] ;
   int age ;
   float bs ;
} ;
struct emp e ;
fp = fopen ( "EMPLOYEE.DAT", "w" ) ;
if ( fp == NULL )
{
   puts ( "Cannot open file" ) ;
   exit( ) ;
}
while ( 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) " ) ;
   another = getc( ) ;
}
fclose ( fp ) ;
```

Thank You

Home Work (HW)

- Write a program to read a file and display contents with its line numbers.
- Suppose a file contains student's records with each record containing name and age of a student. Write a program to read these records and display them in sorted order by name.
- Write a program to find the size of a text file without traversing it character by character.
- Write a program to copy one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.
- Write a program that merges lines alternately from two files and writes the results to new file. If one file has less number of lines than the other, the remaining lines from the larger file should be simply copied into the target file.

Home Work (HW)

- Given a list of names of students in a class, write a program to store the names in a file on disk. Make a provision to display the nth name in the list (n is data to be read) and to display all names starting with S.
- Given a text file, write a program to create another text file deleting the words “a”, “the”, “an” and replacing each one of them with a blank space.
- A hospital keeps a file of blood donors in which each record has the format:

  Name: 20 Columns
  Address: 40 Columns
  Age: 2 Columns
  Blood Type: 1 Column (Type 1, 2, 3 or 4)

  Write a program to read the file and print a list of all blood donors whose age is below 25 and blood is type 2.