# Introduction to Scientific and Engineering Computation

(BIL 102E)

**LECTURE 13**

**Files**

## Handling Files

- In our programs so far, we have only used the standard input, which is normally the keyboard, for receiving data from the outer world, and the standard output, which is normally the screen, for sending data to the outer world.
- When we exit the program all the information contained in the program (values of variables) is lost.
- This means that every time we start our computer we have to re-enter any necessary input data, re-calculate any necessary calculations, and print out the results.
- However, one of the most important characteristics of computers is their ability to store data for later use.
- The data, in general, is stored in **files** on hard-disks, floppy-disks, magnetic-tapes, zip-drives, CD-ROMS etc.

- Files can also contain program source codes (called *source code files*) and machine language instructions (called *executable files*).
- Files that contain data to be processed by programs are called *data files*.
- Data files consist of units of data called *records*.
- There are two kinds of *data files*:
  1. *Sequential Files*
  2. *Direct (Random) Access Files*

## The FILE structure

The FILE structure is defined in stdio.h and is used to access files. In order to reach files we first define a pointer to FILE. For example, **FILE** *fptr;

## Opening a file:

To be processed, each file should be opened first. **fopen**() function is used for this purpose. The syntax is

*file_ptr* = fopen(*file_name*, *mode*);

Here *file_ptr* is a pointer to FILE structure, *file_name* is a string consisting of the path and name of the file to be opened, and *mode* is another string that determines the opening mode of the file. After the file is opened it can be reached via *file_ptr*. If fopen is not success for any reason (for example file does not exist or the disk is full etc) it returns NULL value.

The mode parameter is made by a combination of the characters r (read), w (write), b (binary), a (append), and + (update).

For example,

"r" → opens existing text file for reading
"w" → creates a text file for writing
"a" → opens an existing text file for appending
"r+" → opens an existing text file for reading or writing
"w+" → creates a text file for reading and writing
"a+" → opens or creates a text file for appending
"rb" → opens an existing binary file for reading
"wb" → creates a binary file for writing
"ab" → opens an existing binary file for appending
"r+b" → opens an existing binary file for reading or writing
"w+b" → creates a binary file for reading and writing
"a+b" → opens or creates a binary file for appending

## Closing a file

It is a good programming practice to close any files opened before the termination of the program. It is also a requirement that a file has to be closed before it can be opened again (perhaps in another mode).

This is done by using the **fclose**() function.

The syntax is as follows:

**fclose**(file_ptr);

Here, file_ptr is a pointer to a FILE that has been opened.

## Reading and Writing Text Files

**fgetc**(file_ptr) and **fputc**(ch, file_ptr) functions can be used to read and write characters to text files. Here, ch is the character to written to the file and file_ptr is a pointer to an opened FILE. **fgetc**() returns the character read from the file.

**fgets**(str, n, file_ptr) and **fputs**(str, file_ptr) functions can be used to read and write strings to text files. Here, str is a pointer to a string (or a character array), file_ptr is a pointer to an opened FILE and n is an integer that shows the maximum number of characters to be read from the file.

It is also possible to use **fprintf**() and **fscanf**() for formatted input/output. These are very similar to **printf**() and **scanf**() functions, except that they require a FILE pointer as the first argument.

## Example

This program creates a file named try.txt and writes "This text is written in a file" in it.

```
#include <stdio.h>
main(){
  FILE *fptr;
  fptr = fopen("try.txt","w");
  if(fptr == NULL){
    printf("Problem in opening the file!");
       return -1;
  }

  fprintf(fptr,"This text is written in a file");
  fclose(fptr);
  return 0;
}
```

Open the file in write-only mode.

Check if any problem has occured while openning the file.

This file reads the data previously written in try.txt.

```c
#include <stdio.h>
main(){
  FILE *fptr;
  char buffer[255];
  fptr = fopen("try.txt","r");
  if(fptr == NULL){
    printf("Problem in opening the file!");
       return -1;
  }

  fgets(buffer,250,fptr);
  printf("%s\n",buffer);
  fclose(fptr);
  return 0;
}
```

Open the file in read-only mode.

Read at most 250 characters from the file.

```
This text is written in a file
```

## End of file

When reading data from files, it is necessary to test whether the end of the file has been reached or not.

If **fgetc**() or **fscanf**() functions are used they return an EOF character when the end of the file has been reached.

If **fgets**() is used it returns a NULL pointer.

It is also possible to use the **feof**() (end of file) function to test whether the end of the file has been reached or not.

## Example

Write a program that reads 5 integers from the keyboard and stores them in separate lines in a file called *Integers.txt*. If an error occurs in opening the file an appropriate message should be displayed.

```c
#include <stdio.h>
main(){
  FILE *fIntegers;
  int i,k;
  fIntegers=fopen("Integers.txt","w");
  if(fIntegers==NULL){
    printf("Problem in openning the file!");
    return -1;
  }
  for(i=0;i<5;i++) {
    printf("Enter a number:\n");
    scanf("%d",&k);
    fprintf(fIntegers,"%6d\n",k);
  }
  fclose(fIntegers);
  return 0;
}
```

```
Enter a number:
5
Enter a number:
15
Enter a number:
344
Enter a number:
12
Enter a number:
2443
```

If the numbers

`5, 15, 344, 12, 2443`

are entered from the keyboard then the file *Integers.txt* will consist of the following 5 lines:

```
     5
    15
   344
    12
  2443
```

## Example

Write a program that reads 5 Integers from the first five records of a file called Integers.txt, and then prints them on screen. The numbers should be assumed to be in the same format as in the the previous example. If an error occurs in opening the file an appropriate message should be displayed.

```
#include <stdio.h>
main(){
  FILE *fIntegers;
  int i,k;
  fIntegers=fopen("Integers.txt","r");
  if(fIntegers==NULL){
    printf("Problem in openning the file!\n");
    return -1;
  }
  for(i=0;i<5;i++) {
    fscanf(fIntegers,"%d",&k);
    printf("%6d\n",k);
  }
  fclose(fIntegers);
  return 0;
}
```

If the file is corrupted or it contains less than 5 records this program fails. A better programming example would be checking for such conditions.

## Example (A program that saves student informatıon)

```
#include <stdio.h>
/* ... We have defined some of the functions
   used by thıs program in the previous lectures */
void SaveStudent(FILE *f, student *stud){
  fprintf(f,"%-15s",stud->individual.name);
  fprintf(f,"%-15s",stud->individual.surname);
  fprintf(f,"%-15s\n",stud->department);
}
main()
{
  int i,j, num_studs=0;
  student *studs;
  FILE *fstuds;
  printf("Enter the number of students : ");
  scanf("%d",&num_studs);
  studs=malloc(num_studs * sizeof(student));
  if(studs == 0) {
    printf("Memory Allocation Error!\n");
       return 1;
  }    /* Continues from next page */
```

```
for(i=0;i<num_studs;i++) {
        printf("Information on Student %d:\n",i+1);
        ReadStudent(studs+i);
  }
  for(i=0;i<num_studs-1;i++)
         for(j=i+1; j<num_studs; j++)
                studsort2_surname(studs+i, studs+j);
  /* Save the result */
  fstuds=fopen("Students.txt","w");
  if(fstuds==NULL){
    printf("Problem in openning the file!");
       return -1;
  }
  fprintf(fstuds,"%-15s%-15s%-15s\n","Name",
                       "Surname","Department");
  for(i=0;i<num_studs;i++)
         SaveStudent(fstuds,studs+i);

  return 0;
}
```

### A sample execution of the program

```
Enter the number of students : 3
Information on Student 1:
Name : Ahmet
Surname : Zaimoglu
Department : Computer
Information on Student 2:
Name : Mehmet
Surname : Halacoglu
Department : Textile
Information on Student 3:
Name : Ahmet
Surname : Aksin
Department : Mining
```

### Students.txt file after the execution

```
Name           Surname        Department
Ahmet          Aksin          Mining
Mehmet         Halacoglu      Textile
Ahmet          Zaimoglu       Computer
```

## Example

In an experiment, values of the temperature in a container is observed at several times during a day and the results of observations are entered from the keyboard to be saved in a file, whose name is entered by the user.

Write a program called TemperatureReader that first reads the name of the output file and then asks the user the time and the corresponding temperature repetitively until the temperature entered is –99. The time should be read as a four character string in the hhmm format. The program should allow integer temperatures between –99 and 250. As the information is being entered from the keyboard, it should be written to the output file in a suitable tabular format. In order to increase the readability of the output file, the columns in the output file should be preceded by suitable labels such as *Time* and *Temp*.

```
#include <stdio.h>
main(){
 FILE *ftemp;
 char filename[250];
 char tm[5];
 int temp;

 printf("Enter the name of the file:\n");
 scanf("%250s",filename);
 ftemp=fopen(filename,"w");
 if(ftemp==NULL){
   printf("Problem in openning the file!");
        return -1;
 }
 fprintf(ftemp,"Time \t Temp \n ---- \t ----\n");
```

(continues in the next slide)

```
 do {
    printf("Please enter time:\n");
     scanf("%4s",tm);
    do {
          printf(“Please enter temperature:\n");
          scanf("%d",&temp);
    } while(temp<-99 || temp>250);
    if (temp!= -99)
          fprintf(ftemp,"%4s \t %4d\n",tm,temp);
 } while(temp!= -99);

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

A sample run of the program:

```
Please enter the name of output file:
Experiment.txt
Please enter time:
0910
Please enter temperature:
45
Please enter time:
0930
Please enter temperature:
55
Please enter time:
0950
Please enter temperature:
640
Please enter temperature:
64
Please enter time:
1020
Please enter temperature:
-99
```

*Experiment.txt* file after the execution:

```
Time Temp
---- ----
0910   45
0930   55
0950   64
```

A program that prints the contents of a text file.

```
#include <stdio.h>
main(){
  FILE *ftemp;
  char filename[250],buffer[80];
  int flag=0;

  printf("Enter the name of the file:\n");
  scanf("%250s",filename);
  ftemp=fopen(filename,"r");
  if(ftemp==NULL){
    printf("Problem in openning the file!");
            return -1;
  }
  while(!feof(ftemp)) {
    flag=fgets(buffer,80,ftemp);
    if(flag != NULL)
            printf("%s",buffer);
  }
  fclose(ftemp);
}
```