C Programming

Lesson 7: File Handling

Table of Contents

- **File Handling:**
  - Creation and Access
  - Text files and Data Files
  - Random access and Error Handling

Lesson Objectives

- **In this lesson, you will learn the method to process files and using functions to do it.**

Lesson Coverage

- **In this lesson, you will cover:**
  - Linked List: Read from file, build, save on file
  - Error Handling Functions: ferror(), perror()
  - File Positioning: fseek(), ftell(), rewind()
  - File Handling Overview:
    - Unformatted high-level disk I/O functions: fopen(), fclose()
    - Character Input/Output in files: getc(), putc()
    - String (line) Input/Output in Files: fgets(), fputs()
    - Formatted high-level disk I/O functions: fscanf(), fprintf()
    - Direct Input/Output: fread(),fwrite(), feof()

Sample Code

```
/*This program takes the contents of a text file and copies into another text file,
character by character */
void main(void)  {
        FILE *fileptr_read,*fileptr_write;
        char content;
        fileptr_read=fopen("pr1.c","r"); /* open file in read mode */
        if(fileptr_read==NULL)              {
                        puts("Cannot open source file");
                        exit(0);
        }
        fileptr_write=fopen("pr2.c","w"); /*open file in write mode*/
```

Sample Code

```
if(fileptr_write==NULL) {
                puts("Cannot open target file");
                fclose(fileptr_read);
                exit(0);
    }
    while(1) {
            content=getc(fileptr_read);
                if(content==EOF)
                    break;
                putc(content,fileptr_write);
    }
fclose(fileptr_read);
fclose(fileptr_write);
}
```

Functions Used

- **They are used to read and write a string of characters from / to a file.**
- **Functions used are as follows:**
  - fgets() :
    - It reads a line of text from a file.
    - The general format is as shown below:

```
char *fgets( char *s, int n, FILE *fp);
```

    - It reads character from the stream fp into the character array ‘s’ until a newline character is read, end-of-file is reached, or until n-1 characters are read.

Functions Used

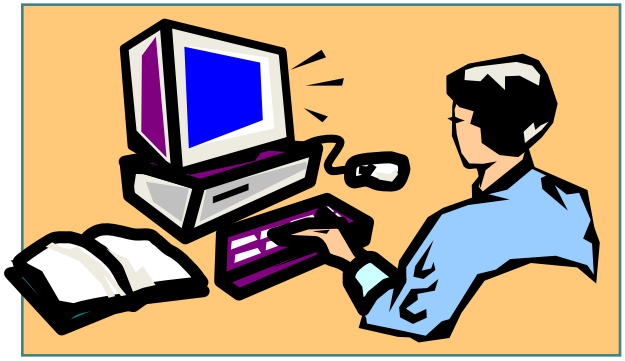
- **Functions used are as follows (contd.):**
  - fputs() :
    - It writes a line of text from a file.
    - The general format is as shown below:

```
int fputs(const char *s, FILE *fp);
```

    - It writes to the stream fp except the terminating null character of string ‘s’.
    - It returns the non-negative value, on success. It returns EOF, in case of an error.

Demo on String I/O in file

- **Demo fputs_fgets.c**



Sample Code

- **Let us see a program using String Input/Output functions:**

```
/* Receives strings from user and writes them to file. */

void main(void)
{
        FILE *fp;
        char data[80];
        fp=fopen("test.txt","w");
        if(fp==NULL)
        {
                puts("Cannot open file");
                exit(0);
        }
```

Sample Code

```
printf("Enter few lines of text \n ");
        while(strlen(gets(data)) >0)
        {
                fputs(data,fp);
                fputs("\n",fp);
        }
        fclose(fp);
}
```

Functions Used

- **They provide facilities to read and write a certain number of data items of specified size.**

- **Functions used are as follows:**
  - int fread(ptr,size,nitems,fp) :
    - It reads into buffer ptr the nitems of data items of size size from the stream fp.
    - It returns the number of items read, on success.
    - It returns EOF, if any error occurs.
  - int fwrite(ptr,size,nitems,fp) :
    - It appends at the most nitems item of data of size size in the file, from the array which ptr points to.
    - It returns the number of items written, on success.
    - It returns EOF if an error is encountered.

Usage of Error Handling Functions

- **Let us see some of the Error Handling Functions:**
  - int feof(FILE *fp);
    - It returns true (non-zero) if the end of the file pointed to by fp has been reached; otherwise it returns zero.
  - int ferror(FILE *fp);
    - It returns a non-zero value if the error indicator is set for the stream fp; otherwise it returns zero.
  - void perror(const char *s);
    - It writes the string ‘s’ followed by a colon and a space to the standard error output stderr, and then an implementation-defined error message corresponding to the integer in errno, terminated by a newline character.

Sample Code

```
FILE *fp,*fpr; char another='Y';
struct emp
{
        char name[40];
        int age;
        float bs;
};
struct emp e; fp=fopen("emp.dat","w");
if(fp==NULL)
{ puts("Cannot open file");
exit(0); }
while(another=='Y')
{
        printf("\n enter name , age basic salary\n");
        scanf("%s%d%f",&e.name,&e.age,&e.bs);
        fwrite(&e,sizeof(e),1,fp);
```

Sample Code

```
printf("Add another record (Y/N)");
fflush(stdin);another=getchar();
}
fclose(fp);
fpr=fopen("emp.dat","r");
if(fpr==NULL)
{
        puts("Cannot open file");
        exit(0);
}
while(fread(&e,sizeof(e),1,fpr)==1)
        printf("%s %d %f \n",e.name,e.age,e.bs);
fclose(fpr);
```

Functions Used

- long ftell(FILE *fp);
  - It returns the current value of the file position indicator associated with fp.
- void rewind(FILE *fp);
  - It resets the current value of the file position indicator associated with fp to the beginning of the file.
  - It allows a program to read through a file more than once without having to close and open the file again.

Summary

- **In this lesson, you have learnt:**
  - The getc() and putc() functions can be used for character I/O.
  - The main() function takes two arguments called argv and argc.
  - fread() and fwrite() functions provide facilities to read and write a certain number of data items of specified size.

Review Question

- **Question 1: stderr, stdin and stdout are FILE pointers.**
  - True/False

- **Question 2: The structure of FILE is defined in ___ header file.**

- **Question 3: A File written in text mode can be read back in the binary mode.**
  - True/False

Review Question: Match the Following

1. fseek(fp,n,SEEK_CUR)	1. sets cursor back from current position by n bytes
2. fseek(fp,-n,SEEK_CUR)	2. sets cursor ahead from current position by n bytes
3. fseek(fp,0,SEEK_END)	3. sets cursor to the beginning of the file
4. fseek(fp,0,SEEK_SET)	4. sets cursor to the end of the file

Lab Session

- **Lab 7**