Department of
Computer Science and Engineering

Title: File Operations in C

Structured Programming Lab

CSE 104

Green University of Bangladesh

1 Learning Objective

- To create, update, read, and delete the files stored on the local file system through C program
- To handle standard I/O in C using fprintf(), fscanf(), fread(), fwrite(), fseek() etc. with the help of examples

2 What is File Handling in C

A file is nothing but a source of storing information permanently in the form of a sequence of bytes on a disk. The contents of a file are not volatile like the C compiler memory. The various operations available like creating a file, opening a file, reading a file or manipulating data inside a file is referred to as file handling.

2.1 Why files are needed

There is a time when the output generated by compiling and running the program does not serve the purpose. If we want to check the output of the program several times, it becomes a tedious task to compile and run the same program multiple times. This is where file handling comes into play. Here are some of the following reasons behind the popularity of file handling:

- Reusability: It helps in preserving the data or information generated after running the program.
- Large storage capacity: Using files, we need not worry about the problem of storing data in bulk.
- Saves time: There are certain programs that require a lot of input from the user. We can easily access any part of the code with the help of certain commands.
- Portability: We can easily transfer the contents of a file from one computer system to another without having to worry about the loss of data.

3 Types of Files

When dealing with files, there are two types of files we should know about:

1. Text files
2. Binary files

3.1 Text files

Text files are the normal .txt files. We can easily create text files using any simple text editors such as Notepad. When we open those files, we will see all the contents within the file as plain text. We 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.

3.2 Binary files

Binary files are mostly the .bin files in our 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.

4 File Operations

In C, we can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

4.1 Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

```
FILE *fptr;
```

4.2 Opening a file - for creation and edit

Opening a file is performed using the fopen() function defined in the stdio.h header file. The syntax for opening a file in standard I/O is:

```
ptr = fopen("fileopen","mode");
```

For example,

```
fopen("E:\\cprogram\\newprogram.txt","w");
fopen("E:\\cprogram\\oldprogram.bin","rb");
```

Let's suppose the file **newprogram.txt** doesn't exist in the location E:\cprogram. The first function creates a new file named **newprogram.txt** and opens it for writing as per the mode 'w'. The writing mode allows us to create and edit (overwrite) the contents of the file. Now let's suppose the second binary file **oldprogram.bin** exists in the location E:\cprogram. The second function opens the existing file for reading in binary mode 'rb'. The reading mode only allows us to read the file, we cannot write into the file.

MODE	ELUCIDATION
r	We use it to open a text file in reading mode
w	We use it to open or create a text file in writing mode
a	We use it to open a text file in append mode
r+	We use it to open a text file in both reading and writing mode
w+	We use it to open a text file in both reading and writing mode
a+	We use it to open a text file in both reading and writing mode
rb	We use it to open a binary file in reading mode
wb	We use it to open or create a binary file in writing mode
ab	We use it to open a binary file in append mode
rb+	We use it to open a binary file in both reading and writing mode
wb+	We use it to open a binary file in both reading and writing mode
ab+	We use it to open a binary file in both reading and writing mode

Figure 1: Opening Modes in Standard I/O

4.3 Closing a File

The file (both text and binary) should be closed after reading/writing. Closing a file is performed using the fclose() function.

```
fclose(fptr);
```

Here, fptr is a file pointer associated with the file to be closed.

4.4 Reading and writing to a text file

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.

Example 1: Write to a text file

The following program takes a number from the user and stores in the file program.txt.

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

int main()
{
   int num;
   FILE *fptr;

   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }

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

   fprintf(fptr,"%d",num);
   fclose(fptr);

   return 0;
}
```

After we compile and run this program, we can see a text file program.txt created in C drive of our computer. When we open the file, we can see the integer we entered.

Example 2: Read from a text file

The following program reads the integer present in the program.txt file and prints it onto the screen. If we successfully created the file from Example 1, running this program will get us the integer we entered. Other functions like fgetchar(), fputc() etc. can be used in a similar way.

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

int main()
{
   int num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.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;
}
```

4.5 Reading and writing to a binary file

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.

4.5.1 Writing to a binary file

To write into a binary file, we need to use the fwrite() function. The functions take four arguments:

1. address of data to be written in the disk
2. size of data to be written in the disk
3. number of such type of data
4. pointer to the file where we want to write.

```
fwrite(addressData, sizeData, numbersData, pointerToFile);
```

Example 3: Write to a binary file using fwrite()

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

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.bin","wb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      num.n1 = n;
      num.n2 = 5*n;
      num.n3 = 5*n + 1;
       fwrite(&num, sizeof(struct threeNum), 1, fptr);
   }
   fclose(fptr);

   return 0;
}
```

In this program, we create a new file program.bin in the C drive. We declare a structure **threeNum** with three numbers - n1, n2 and n3, and define it in the main function as num. Now, inside the for loop, we store the value into the file using **fwrite()**. The first parameter takes the address of num and the second parameter takes the size of the structure **threeNum**. Since we're only inserting one instance of num, the third parameter is 1. And, the last parameter ***fptr** points to the file we're storing the data. Finally, we close the file.

4.5.2 Reading from a binary file

Function fread() also take 4 arguments similar to the fwrite() function as above.

```
fread(addressData, sizeData, numbersData, pointerToFile);
```

Example 4: Read from a binary file using fread()

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

struct threeNum
{
   int n1, n2, n3;
};

```

```c
int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
        printf("Error! opening file");

        // Program exits if the file pointer returns NULL.
        exit(1);
    }

    for(n = 1; n < 5; ++n)
    {
       fread(&num, sizeof(struct threeNum), 1, fptr);
       printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2, num.n3);
    }
    fclose(fptr);

    return 0;
}
```

In this program, we read the same file program.bin and loop through the records one by one. In simple terms, we read one threeNum record of threeNum size from the file pointed by *fptr into the structure num. we shall get the same records we inserted in Example 3.

5 File handling exercises

5.1 Problem 01: C program to count characters, words and lines in a text file

Pseudocode

Algorithm 1 Character, Word and Line Counter

1: Step 01: Open source file in r (read) mode.
2: Step 02: Initialize three variables characters = 0, words = 0 and lines = 0 to store counts.
3: Step 03: Read a character from file and store it to some variable say ch.
4: Step 04: Increment characters count.
5: Step 05: Increment words count if current character is whitespace character (space, tab, new line, null character)
6: Step 06: Increment lines count if current character is new line character i.e. if (ch == '\n' || ch == '\0').
7: Step 07: Repeat step 3-4 till file has reached end.
8: Step 08: Finally after file has reached end increment words and lines count by one if total characters > 0 to make sure we count last word and line.

Code

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

int main()
{
    FILE * file;
    char path[100];

    char ch;
    int characters, words, lines;


    /* Input path of files to merge to third file */
    printf("Enter source file path: ");
    scanf("%s", path);

    /* Open source files in 'r' mode */
    file = fopen(path, "r");

```

```c

    /* Check if file opened successfully */
    if (file == NULL)
    {
        printf("\nUnable to open file.\n");
        printf("Please check if file exists and you have read privilege.\n");

        exit(EXIT_FAILURE);
    }

    /*
     * Logic to count characters, words and lines.
     */
    characters = words = lines = 0;
    while ((ch = fgetc(file)) != EOF)
    {
        characters++;

        /* Check new line */
        if (ch == '\n' || ch == '\0')
            lines++;

        /* Check words */
        if (ch == ' ' || ch == '\t' || ch == '\n' || ch == '\0')
            words++;
    }

    /* Increment words and lines for last word */
    if (characters > 0)
    {
        words++;
        lines++;
    }

    /* Print file statistics */
    printf("\n");
    printf("Total characters = %d\n", characters);
    printf("Total words      = %d\n", words);
    printf("Total lines      = %d\n", lines);


    /* Close files to release resources */
    fclose(file);

    return 0;
}
```

Input

Suppose if data\file3.txt contains

```
I love programming.
Working with files in C programming is fun.
I am learning C programming at Bangladesh.
```

Output

```
Total characters = 106
Total words      = 18
Total lines      = 3
```

5.2 Problem 02: C program to count occurrences of a word in file

Pseudocode

Algorithm 2 Word Frequency Calculation

1: Step 01: Open source file in r (read) mode, store its reference to fptr.
2: Step 02: Input word to search from user, store it in word.
3: Step 03: Initialize a variable count = 0 to store total occurrences of word.
4: Step 04: Read a line from file and store it in str.
5: Step 05: Increment count by one, if an occurrence of word is found in str.
6: Step 06: Repeat step 4-5 till end of file.
7: Step 07: Finally close fptr to release all resources.

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BUFFER_SIZE 1000
int countOccurrences(FILE *fptr, const char *word);


int main()
{
    FILE *fptr;
    char path[100];
    char word[50];
    int wCount;

    /* Input file path */
    printf("Enter file path: ");
    scanf("%s", path);

    printf("Enter word to search in file: ");
    scanf("%s", word);

    /* Try to open file */
    fptr = fopen(path, "r");

    /* Exit if file not opened successfully */
    if (fptr == NULL)
    {
        printf("Unable to open file.\n");
        printf("Please check you have read/write previleges.\n");

        exit(EXIT_FAILURE);
    }
    wCount = countOccurrences(fptr, word);

    printf("'%s' is found %d times in file.", word, wCount);

    fclose(fptr);

    return 0;
}


/**
 * Returns total occurrences of a word in given file.
 */
int countOccurrences(FILE *fptr, const char *word)
{
    char str[BUFFER_SIZE];
    char *pos;

    int index, count;

    count = 0;

```

```c
    // Read line from file till end of file.
    while ((fgets(str, BUFFER_SIZE, fptr)) != NULL)
    {
        index = 0;

        // Find next occurrence of word in str
        while ((pos = strstr(str + index, word)) != NULL)
        {
            // Index of word in str is
            // Memory address of pos - memory
            // address of str.
            index = (pos - str) + 1;

            count++;
        }
    }

    return count;
}
```

Input

Suppose data\file3.txt contains

```
I love programming.
I am learning C programming at Green University.
Programming with files is fun.
Learning C programming at Green University is simple and easy.
```

Output

```
Enter file path: data/file3.txt
Enter word to search in file: Green
'Green' is found 2 times in file.
```

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a C program to read name and marks of n number of students and store them in a file.
2. Write a 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.
3. Write a C program to find and replace a word in a text file.

7 Lab Exercise (Submit as a report)

1. Write a 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.
2. Write a C program to count occurrences of all words in a text file.
3. Write a C program to convert uppercase to lowercase character and vice versa in a text file.

8 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.