Module 13 – part 2 – File Handling

BITS Pilani
Pilani Campus

Dr. Jagat Sesh Challa
Department of Computer Science & Information Systems

Module Overview

- File and File Handling
- File Handling with Command Line Arguments

innovate achieve lead

BITS Pilani

Pilani Campus

File and File Handling

File

- Applications require information to be read from or written to memory device (**DISK**) which can be accomplished using **files.**

- All files are **administered by the operating system** which allocates and deallocates disk blocks.

- The operating system also controls the transfer of data between programs and the files they access.

File handling Basics

innovate achieve lead

- A file has to ***opened*** before data can be read from or written to it.
- When a file is opened, the operating system associates a ***stream*** with the file.
- A **common *buffer*** and a ***file position indicator*** are maintained in the memory for a function to know how much of the file has already been read.
- The stream is disconnected when the file is closed.

fopen: Opening a File

- `FILE *fp;` *Defines file pointer*
- `fp = fopen("foo.txt", "r");` *Only reading permitted*

- If the call is successful, **`fopen`** returns a pointer to a structure typedef'd to FILE.

- The pointer variable, **`fp`**, assigned by **`fopen`** acts as a file handle which will be used subsequently by all functions that access the file.

File Opening Modes

- **r** – Reading a file
- **w** – Writing a file
- **a** – Appending to the end of an existing file

When used with "**w**" or "**a**", `fopen` creates a file if it does not find one.

`fopen` will fail if the file does not have the necessary permissions (r, w and a) or if the file does not exist in case of "**r**".

File Opening Modes: Extended

- Database applications often need both read and write access for the same file, in which case, you must consider using the "**r+**", "**w+**" and "**a+**" modes.

Mode	File opened for
r	Reading only
r+	Both reading and writing
w	Writing only
w+	Both reading and writing
a	Appending only
a+	Reading entire file but only appending permitted

File Read/Write Functions

The standard library offers a number of functions for performing read/write operations on files.

1. Character-oriented functions (**`fgetc`** and **`fputc`**)
2. Line-oriented functions (**`fgets`** and **`fputs`**)
3. Formatted functions (**`fscanf`** and **`fprintf`**)

All of these functions are found in the standard library **`stdio.h`**.

fclose: Closing a file

- Operating systems have a limit on the number of files that can be opened by a program.
- Files must be closed with the **`fclose`** function:
- **`fclose(fp);`**
- Closing a file frees the file pointer and associated buffers.

Example 1

```
int main(){
  FILE *fp; char buf1[80], buf2[80];
  fputs("Enter a line of text: \n", stdout);
  fgets(buf1, 79, stdin);
  fp = fopen("foo", "w");
  fputs(buf1, fp);
  fclose(fp);
  fp = fopen("foo", "r");
  fgets(buf2, 79, fp);
  fputs(buf2, stdout);
  fclose(fp);
  return 0;
}
```

```
if (fp == NULL){
  fputs("Error", stdout);
  return 0;
}
```

OUTPUT

```
Enter a line of text:
Hello. This is file handling.
```

main.c | foo

```
1  Hello. This is file handling.
```

```
Hello. This is file handling.
```

Example 2:

Read the numbers in the “data.txt”, compute their sum and write the sum to “output_file.txt”.

data.txt:

45
34
67
46
99
103
183
59
30
56

Example 2: Code


```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   char toRead[100];
   FILE * fp = fopen("data2.txt", "r");
   int sum = 0;
   while(fgets(toRead,100,fp) != NULL) {
       printf( "%s\n" , toRead) ;
       sum += atoi(toRead);
   }
   fclose(fp);
   FILE * fp2 = fopen("output_file.txt", "w");
   char char_sum[50];
   sprintf(char_sum,"%d",sum);
   fputs(char_sum,fp2);
   fclose(fp2);
}
```

Program to manipulate the file order offset pointer

```
int main(){
   FILE *fp; int c, x;
   char buf[80] = "A1234 abcd"; char stg[80];
   fp = fopen("foo", "w+");
   fputs(buf, fp);
   rewind(fp);
   c = fgetc(fp);
   fputc(c, stdout);
   printf("\n");
   fscanf(fp, "%d", &x);
   fprintf(stdout, "%d", x);
   printf("\n");
   fgets(stg, 4, fp);
   fputs(stg, stdout);
   printf("\n");
   return 0;
}
```

Rewind function sets the file position to the beginning of the file of the given stream.

main.c | foo

```
1  A1234 abcd
```

```
jagat@Prithvi:~/cp$ ./a.out
A
1234
 ab
```

Program to read text from terminal, save it in a file and then reading again


```
int main(){
   FILE *fp; char buf[80];
   fp = fopen("foo", "w+");
   fputs("Enter a few lines, [Ctrl-d] to exit\n",stdout);
   while(fgets(buf, 80, stdin)!= NULL)
       fputs(buf, fp);
   rewind(fp);
   fputs("Reading from foo... \n", stdout);
   while(fgets(buf, 80, fp)!= NULL)
      fputs(buf, stdout);
   return 0;
 }
```

OUTPUT

```
Enter a few lines, [Ctrl-d] to exit
Line 1 of file hadling.
Line 2 of file handling.
```

main.c | foo

```
1  Line 1 of file hadling.
2  Line 2 of file handling.
```

```
Reading from foo...
Line 1 of file hadling.
Line 2 of file handling.
```

Practice problems

- Write a C program which prints itself! [Hint: use file handling]

- Extend the above to program such that comments are not printed. For simplicity, consider single line comments beginning with //

- Given a file f1, write a program to create a new file f2 which contains the contents of f1 double-spaced (i.e., each space replaced by two spaces, each new line by two new lines, and each tab (\t) by two tabs. Rest of the content remains unchanged).

innovate achieve lead

BITS Pilani

Pilani Campus

Example of File Handling with Command Line Arguments

Command Line Arguments

innovate achieve lead

- Values can be passed from command line when the C programs are executed.

- To handle command line arguments, **`main()`** function is modified as:

  - **`int main(int argc, char *argv[])`**
  - **`argc`** – number of arguments passed
  - **`argv[]`** – is a pointer array to the arguments passed

Example 1:

```
int main(int argc, char *argv[])
{
 printf("Number of arguments: %d \n", argc);
 int i = 0;
 while(i < argc){
   printf("Argument %d: %s \n",i, argv[i]);
   i++;
 }
 return 0;
}
```

```
amitesh@Prithvi:~$ gcc test1.c
amitesh@Prithvi:~$ ./a.out hello world
Number of arguments is 3
Argument 0: ./a.out
Argument 1: hello
Argument 2: world
```

Example 2: To display the content of one file

innovate achieve lead

```
#include <stdio.h>
int main(int argc, char *argv[]){
  FILE *fp; char ch;
  fp = fopen(argv[1], "r");
  if (fp == NULL){
     printf("Error");
     return(0);}
  ch = fgetc(fp);
  while (ch != EOF){
     printf("%c", ch);
     ch = fgetc(fp);}
  fclose(fp);
}
```

Example 3: To display the content of more than one file

innovate achieve lead

```
#include <stdio.h>
int main(int argc, char *argv[]){
  int nof = argc-1;
  while (nof > 0){
     FILE *fp; char ch;
     fp = fopen(argv[nof],"r");
     ch = fgetc(fp);
        while (ch != EOF){
           printf ("%c", ch);
           ch = fgetc(fp);}
     fclose (argv[nof]);}
     nof--;
}
```

innovate achieve lead

BITS Pilani

Pilani Campus

Thank you

Q & A