09 File Handling

Opening/Closing a file

```
#include <stdlib.h>
#include <stdio.h>
/* File pointer to hold reference to our file */
FILE * fPtr;
/*
* Open file in w (write) mode. "data/file1.txt"
is complete path to create file
*/
fPtr = fopen("data/file1.txt", "w");
/* fopen() return NULL if last operation
was unsuccessful */
if(fPtr == NULL)
```

```
{
    /* File not created hence exit */
    printf("Unable to create file.\n");
    exit(0);
}
/* Done with this file, close file
to release resource */
fclose(fPtr);
```

Reading from file

- `fgetc()` – Used to read single character from file.
- `fgets()` – Used to read string from file.
- `fscanf()` – Use this to read formatted input from file.
- `fread()` – Read block of raw bytes from file. Used to read binary files.

Reading

- Open a file using `fopen()` function and store its reference in a `FILE` pointer say `fPtr`.
- You must open file in `r` (read) mode or atleast mode that support read access.
- Read content from file using any of these functions `fgetc()`, `fgets()`, `fscanf()` or `fread()`. Finally, close the file using `fclose(fPtr)`.

Reading from file, char by char

```
do {
    /* Read single character from file */
    ch = fgetc(fPtr);
    /* Print character read on console */
    putchar(ch);
} while(ch != EOF); /* Repeat this
if last read character is not EOF */
```

Copying a file with source/destination as commandline arguments

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

int main(int argc, char* argv[]) {

    if (argc != 3) {
        printf("Invalid arguments\n");
        return 0;
    }

    char ch;

    FILE* s = fopen(argv[1], "r");
    FILE* d = fopen(argv[2], "w");

    /* fopen() return NULL if last operation was unsuccessful */
    if(s == NULL || d == NULL)
    {
        /* Unable to open file hence exit */
        printf("Unable to open file.\n");
```

```
        printf("Please check whether file exists and you have read privilege.\n");
        return 0;
    }

    /* File open success message */
    printf("File opened successfully. Reading file contents character by character. \n\n");

    do
    {
        /* Read single character from file */
        ch = fgetc(s);

        /* Print character read on console */
        putchar(ch);

        fputc(ch, d);

    } while(ch != EOF); /* Repeat this if last read character is not EOF */

    /* Done with this file, close file to release resource */
    fclose(s);
    fclose(d);

    return 0;
}
```

Reading line by line using `fgets()`

```
char * fgets(char * str, int num, FILE * stream);
```

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

#define BUFFER_SIZE 1000

int main() {
    /* File pointer to hold reference to our file */
    FILE * fPtr;

    char buffer[BUFFER_SIZE];
    int totalRead = 0;
    int total_chars = 0;

    /*
     * Open file in r (read) mode.
     * "data/file2.txt" is complete file path to read
     */
    fPtr = fopen("1.c", "r");

    /* fopen() return NULL if last operation was unsuccessful */
    if(fPtr == NULL)
    {
        /* Unable to open file hence exit */
        printf("Unable to open file.\n");
        printf("Please check whether file exists and you have read privilege.\n");
        return 0;
    }

    /* File open success message */
    printf("File opened successfully. Reading file contents line by line. \n\n");
```

```
    /* Repeat this until read line is not NULL */
    while(fgets(buffer, BUFFER_SIZE, fPtr) != NULL)
    {
        /* Total character read count */
        totalRead = strlen(buffer);
        total_chars += strlen(buffer);

        /* Print line read on cosole*/
        printf("%s", buffer);

    }

    printf("Total number of chars: %d", total_chars);

    /* Done with this file, close file to release resource */
    fclose(fPtr);

    return 0;
}
```

Home Work

Write a program which takes a file name as command line argument and prints the number of chars, words, lines and paragraphs in the file.