File Handling in C

Lab Sheet

Prasad Wilagama - Assistant Lecturer
University of Colombo School of Computing (UCSC)

17 July 2025

Objective

By the end of this lab, you will:

- Understand basic file handling concepts in C
- Write programs to create, read, write, append files
- Handle formatted input/output to files
- Check for errors and EOF in file operations
- Complete a mini project to copy contents of a file

Pre-requisites

Basic understanding of C programming, including variables, loops, and functions.

Important Notes

- Use your preferred C compiler (gcc recommended).
- Make sure your working directory contains the files you want to open.
- Read instructions carefully before coding.

Task 1: Opening a File

Goal: Open a file in read mode and check if it opened successfully.

Instructions

- Create a file named `data.txt` in your working folder and put some text inside.
- Write a C program to open `data.txt` for reading.
- If the file cannot be opened, print an error message.
- If opened successfully, print a success message.
- Close the file before the program ends.

Starter Code

```c
// Task 1: Open file and check

#include <stdio.h>

int main() {
    FILE *fp = fopen("data.txt", "r");
    if (fp == NULL) {
        printf("Error: Could not open file.\n");
        return 1;
    }
    printf("File opened successfully.\n");

    // TODO: Close the file here
    fclose(fp);

    return 0;
}
```

Task 2: Writing to a File

Goal: Write text data into a file.

Instructions

- Open a file named `output.txt` in write mode.
- Write the text "Hello UCSC students!" into the file.
- Write an additional line "Welcome to File Handling in C."
- Close the file properly.

Starter Code

```c
// Task 2: Write text to file

#include <stdio.h>

int main() {
    FILE *fp = fopen("output.txt", "w");
    if (fp == NULL) {
        printf("Error opening file for writing.\n");
        return 1;
    }

    fprintf(fp, "Hello UCSC students!\n");
    fputs("Welcome to File Handling in C.\n", fp);

    fclose(fp);
    printf("Data written successfully.\n");

    return 0;
}
```

Task 3: Reading from a File

Goal: Read the contents of a file line by line.

Instructions

- Open `output.txt` for reading.
- Use `fgets()` to read each line and print it to the console.
- Close the file after reading.

Starter Code

```c
// Task 3: Read file line by line

#include <stdio.h>

int main() {
    char line[100];
    FILE *fp = fopen("output.txt", "r");
    if (fp == NULL) {
        printf("Unable to open file for reading.\n");
        return 1;
    }

    while (fgets(line, sizeof(line), fp) != NULL) {
        printf("%s", line);
    }

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

Task 4: Appending Data to a File

Goal: Add new lines to an existing file without overwriting.

Instructions

- Open `output.txt` in append mode.
- Append the line: "This line is appended."
- Close the file properly.

Starter Code

```c
// Task 4: Append to file

#include <stdio.h>

int main() {
    FILE *fp = fopen("output.txt", "a");
    if (fp == NULL) {
        printf("Error opening file for appending.\n");
        return 1;
    }

    fputs("This line is appended.\n", fp);

    fclose(fp);
    printf("Data appended successfully.\n");
    return 0;
}
```

Task 5: Writing and Reading Formatted Data

Goal: Save structured data (name and marks) to a file and read it back.

Instructions

- Open a file `students.txt` in write mode.
- Write two lines containing a student name and their marks (e.g., `Alice 85`).
- Close the file.
- Reopen the file in read mode.
- Use `fscanf()` to read each name and marks, then print them.
- Close the file.

Starter Code

```c
// Task 5: Write and read formatted data

#include <stdio.h>

int main() {
    FILE *fp = fopen("students.txt", "w");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    fprintf(fp, "%s %d\n", "Alice", 85);
    fprintf(fp, "%s %d\n", "Bob", 90);
    fclose(fp);

    char name[20];
    int marks;
    fp = fopen("students.txt", "r");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    while (fscanf(fp, "%s %d", name, &marks) != EOF) {
        printf("Name: %s, Marks: %d\n", name, marks);
    }
    fclose(fp);

    return 0;
}
```

Task 6: Checking for EOF and File Errors

Goal: Handle end-of-file and file errors gracefully.

Instructions

- Open `students.txt` for reading.
- Read formatted data with error checking.
- Detect end-of-file and print a message.
- Detect read errors and print a message.
- Close the file.

Starter Code

```c
// Task 6: EOF and error handling

#include <stdio.h>

int main() {
    FILE *fp = fopen("students.txt", "r");
    if (fp == NULL) {
        printf("File not found.\n");
        return 1;
    }

    char name[20];
    int marks;

    while (1) {
        int ret = fscanf(fp, "%s %d", name, &marks);
        if (ret == EOF) break;
        if (ret != 2) {
            printf("File format error.\n");
            break;
        }
        printf("Name: %s, Marks: %d\n", name, marks);
    }

    if (feof(fp)) {
        printf("Reached end of file.\n");
    } else if (ferror(fp)) {
        printf("Error reading file.\n");
    }

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

Task 7: Mini Project - Copy File Contents

Goal: Copy contents from one file to another character by character.

Instructions

- Create a file `input.txt` with some text.
- Open `input.txt` in read mode.
- Open `copy.txt` in write mode.
- Copy all characters from `input.txt` to `copy.txt`.
- Close both files.
- Confirm by reading `copy.txt` or checking file content.

Starter Code

```c
// Task 7: Copy file contents

#include <stdio.h>

int main() {
    FILE *src = fopen("input.txt", "r");
    FILE *dest = fopen("copy.txt", "w");

    if (src == NULL || dest == NULL) {
        printf("Error opening files.\n");
        return 1;
    }

    int ch;
    while ((ch = fgetc(src)) != EOF) {
        fputc(ch, dest);
    }

    fclose(src);
    fclose(dest);

    printf("File copied successfully.\n");
    return 0;
}
```

Bonus Task: Count Lines in a File

Goal: Count total number of lines in a text file.

Instructions

- Open any text file in read mode.
- Read characters one by one.
- Increment a line counter on every newline character '\n'.
- Print the total line count.
- Close the file.

Starter Code

```c
// Bonus: Count lines in a file

#include <stdio.h>

int main() {
    FILE *fp = fopen("input.txt", "r");
    if (fp == NULL) {
        printf("Cannot open file.\n");
        return 1;
    }

    int lines = 0;
    int ch;
    while ((ch = fgetc(fp)) != EOF) {
        if (ch == '\n') lines++;
    }

    fclose(fp);
    printf("Total lines: %d\n", lines);
    return 0;
}
```

Summary

- Files store data persistently on disk.
- Use `fopen()` to open files in various modes.
- Always check if the file opened successfully.
- Use `fprintf()`, `fscanf()`, `fgets()`, `fputs()` for file I/O.
- Always close files with `fclose()`.
- Handle EOF and errors properly to avoid crashes.

Good luck with your coding!