File Handling in C: A Complete Guide

What is File Handling?

File handling refers to reading from and writing to files on a storage device. It allows programs to:

- Store data permanently (beyond program execution)
- Process large datasets
- Share data between programs
- Save configuration/state information

When to Use File Handling

Appropriate Use Cases:

1. Data Persistence
   - Saving user preferences/settings
   - Storing application state between runs
   - Example: Game save files

2. Data Processing
   - Reading large datasets (CSV, logs)
   - Processing files line-by-line
   - Example: Analyzing server logs

3. Configuration Files
   - INI files, JSON configs
   - Example: config.json

4. Inter-Process Communication
   - Sharing data between programs
   - Example: Reports between tools

5. Binary Data Storage

- Saving images, audio
- Example: Image editing software

When NOT to Use File Handling

Poor Use Cases:

1. Frequent Small Writes Slow. Use buffering.
2. Real-Time Systems Unpredictable disk latency. Use RAM.
3. Highly Concurrent Access Race conditions. Use DBs.
4. Temporary Data Use memory variables.
5. Sensitive Data Use encryption or secure DBs.

File Handling Operations in C

1. Opening a File

```
FILE *file = fopen("data.txt", "r");
```

Modes:

- "r" Read
- "w" Write (overwrites)
- "a" Append
- "rb", "wb" Binary modes

2. Reading a File

```
char buffer[100];
while (fgets(buffer, sizeof(buffer), file)) {
    printf("%s", buffer);
}

int ch = fgetc(file);
```

3. Writing to a File

```
fprintf(file, "Name: %s, Age: %d", name, age);
fputs("Hello!", file);
fputc('A', file);
```

4. Binary Operations

```
struct Student s = {1, "Alice"};
fwrite(&s, sizeof(s), 1, file);
fread(&s, sizeof(s), 1, file);
```

5. Closing File

```
fclose(file);
```

Best Practices

Always check fopen() success

Close every opened file

Use binary mode for non-text data

Prefer fgets() over fscanf() for safety

Avoid plaintext for sensitive data

Alternatives to File Handling

- Databases (SQLite, MySQL) for structured data
- Memory-mapped files for speed
- Temporary files (tmpfile()) for short-lived data

Final Verdict

File Handling in C: A Complete Guide

Use File Handling When:

- You need persistent storage
- You are processing large datasets
- You store logs or configuration

Avoid File Handling When:

- You need speed (RAM)
- Multiple accesses to file (Use DB)
- Data is temporary (use variables)