C语言文件操作教学文档

1. 文件操作简介

在C语言中，文件操作是非常重要的一部分，特别是在处理外部数据和存储持久化数据时。C语言提供了几个标准库函数来读取和写入文件。

常见的文件操作步骤包括：

1. **打开文件**：指定文件路径和模式。
2. **文件读写**：读取或写入文件内容。
3. **关闭文件**：释放文件资源。

文件模式

文件模式决定了文件的打开方式。以下是一些常用模式：

- `"r"`：**只读模式。如果文件不存在，打开失败。**
- `"w"`：**写模式。如果文件不存在，会创建文件；如果存在，文件内容会被清空。**
- `"a"`：追加模式。如果文件不存在，会创建文件；如果存在，内容会追加到文件末尾。
- `"r+"`：读写模式。文件必须存在。
- `"w+"`：读写模式。如果文件存在，会被清空；如果不存在，会创建文件。
- `"a+"`：读写追加模式。文件不存在会创建文件，写操作追加到文件末尾。

2. 文件操作函数

2.1 打开文件 - `fopen`

```
FILE *fopen(const char *filename, const char *mode);
```

`fopen`用于打开文件，并返回文件指针，成功则返回`FILE *`，失败返回`NULL`。常见的打开方式：

```
FILE *file = fopen("example.txt", "r");
if (file == NULL) {
    printf("文件打开失败。\n");
}
```

2.2 关闭文件 - `fclose`

```
int fclose(FILE *stream);
```

关闭打开的文件，释放文件资源。成功关闭返回0，失败返回EOF。

```
fclose(file);
```

2.3 文件读取 - `fscanf`、`fgets`、`fgetc`

C语言提供多种读取文件内容的方式：

- `fscanf`：**从文件中读取格式化数据（类似于`scanf`）。**
- `fgets`：读取一行字符串。
- `fgetc`：读取一个字符。

示例代码

```
char buffer[100];
FILE *file = fopen("example.txt", "r");
if (file != NULL) {
    while (fgets(buffer, 100, file) != NULL) {
        printf("%s", buffer);
    }
    fclose(file);
}
```

假设`example.txt`文件内容如下：

```
Alice 20
Bob 22
Charlie 21
```

在每一行中，包含一个名字和一个年龄。我们可以用`fscanf`将名字和年龄读出并输出。以下是修改后的代码示例：

```
#include <stdio.h>

int main() {
    char name[50];
    int age;
    FILE *file = fopen("example.txt", "r");

    if (file != NULL) {
        while (fscanf(file, "%s %d", name, &age) != EOF) {
            printf("姓名: %s, 年龄: %d\n", name, age);
        }
        fclose(file);
    } else {
        printf("无法打开文件。\n");
    }

    return 0;
}
```

2.4 文件写入 - `fprintf`、`fputs`、`fputc`

写入数据到文件的几种常用函数：

- `fprintf`：**格式化写入数据（类似于`printf`）。**
- `fputs`：写入字符串。
- `fputc`：写入单个字符。

示例代码

```
 FILE *file = fopen("example.txt", "w");
if (file != NULL) {
    fprintf(file, "Hello, World!\n");
    fputs("This is a test.\n", file);
    fputc('A', file);
    fclose(file);
}
```

3. 文件操作的常见错误

3.1 文件打开失败

文件打开失败的原因通常包括文件不存在、路径错误或权限不足。在打开文件后应始终检查文件指针是否为`NULL`。

3.2 忘记关闭文件

如果忘记使用`fclose`关闭文件，可能会导致文件资源未释放。特别是在长时间运行的程序中，这种错误会导致资源泄漏。

3.3 错误使用文件模式

使用错误的文件模式可能会导致程序意外覆盖文件内容。例如，使用`"w"`模式打开文件会清空文件内容，因此需谨慎选择模式。

4. 文件读写示例

4.1 从文件读取整数并计算平均值

以下代码演示如何从文件中读取一系列整数并计算平均值：

```
 #include <stdio.h>

int main() {
    FILE *file = fopen("numbers.txt", "r");
    if (file == NULL) {
        printf("无法打开文件。\n");
        return 1;
    }

    int num, count = 0;
    double sum = 0;
```

```
    while (fscanf(file, "%d", &num) != EOF) {
        sum += num;
        count++;
    }

    fclose(file);

    if (count > 0) {
        printf("平均值: %.2f\n", sum / count);
    } else {
        printf("文件为空或没有有效数据。\n");
    }

    return 0;
}
```

4.2 向文件写入结构体数据

以下代码演示如何将一个结构体数组写入文件：

```
 #include <stdio.h>

struct Student {
    char name[50];
    int age;
};

int main() {
    struct Student students[] = {
        {"Alice", 20},
        {"Bob", 22},
        {"Charlie", 21}
    };

    FILE *file = fopen("students.txt", "w");
    if (file == NULL) {
        printf("无法打开文件。\n");
        return 1;
    }

    for (int i = 0; i < 3; i++) {
        fprintf(file, "姓名: %s, 年龄: %d\n", students[i].name, students[i].age);
    }

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

4.3 综合题：编写一个程序，将一组学生的姓名和成绩保存到文件中，并从文件中读取这些信息进行平均成绩计算。

```c
#include <stdio.h>

// 定义学生结构体，包含姓名和分数
struct Student {
    char name[50];  // 学生姓名
    int score;      // 学生分数
};

// 函数：将学生数据数组保存到文件中
void saveToFile(struct Student students[], int count, const char *filename) {
    FILE *file = fopen(filename, "w");  // 打开文件用于写入
    if (file == NULL) {  // 如果文件打开失败
        printf("无法打开文件。\n");
        return;  // 直接返回，不执行后续操作
    }

    // 遍历学生数组，将每个学生的姓名和分数写入文件
    for (int i = 0; i < count; i++) {
        fprintf(file, "%s %d\n", students[i].name, students[i].score);
    }

    fclose(file);  // 关闭文件
    printf("数据已保存到 %s\n", filename);  // 打印成功消息
}

// 函数：从文件中读取数据并计算平均成绩
double calculateAverageScore(const char *filename) {
    FILE *file = fopen(filename, "r");  // 打开文件用于读取
    if (file == NULL) {  // 如果文件打开失败
        printf("无法打开文件。\n");
        return -1;  // 返回错误标志
    }

    char name[50];  // 用于存储读取到的学生姓名
    int score;      // 用于存储读取到的学生分数
    int count = 0;  // 计数器，记录读取到的学生数量
    double sum = 0; // 累计分数

    // 循环读取文件中的每一行，直到文件末尾
    while (fscanf(file, "%s %d", name, &score) != EOF) {
        sum += score;  // 累加分数
        count++;       // 增加计数
    }

    fclose(file);  // 关闭文件

    // 如果没有读取到任何数据，则打印消息并返回0
    if (count == 0) {
        printf("文件中没有有效的数据。\n");
        return 0;
    }
```

```
    // 返回计算出的平均成绩
    return sum / count;
}

// 主函数
int main() {
    // 定义学生数组并初始化
    struct Student students[] = {
        {"Alice", 85},
        {"Bob", 78},
        {"Charlie", 92},
        {"David", 88},
        {"Eve", 90}
    };
    int count = sizeof(students) / sizeof(students[0]);  // 计算学生数量
    char *filename = "students.txt";  // 指定文件名

    // 调用函数将学生数据保存到文件中
    saveToFile(students, count, filename);

    // 调用函数从文件中读取数据并计算平均成绩
    double averageScore = calculateAverageScore(filename);
    // 如果计算平均成绩成功，则打印结果
    if (averageScore != -1) {
        printf("平均成绩: %.2f\n", averageScore);
    }

    return 0;  // 程序正常退出
}
```