Работа с файлами в языке C происходит через стандартную библиотеку, которая предоставляет функции для открытия, чтения, записи и закрытия файлов. Основная структура, используемая для работы с файлами, — это указатель на структуру `FILE`. С его помощью можно управлять файлами на уровне операционной системы.

Основные функции для работы с файлами

1. `fopen` — открытие файла
2. `fclose` — закрытие файла
3. `fread` — чтение данных из файла
4. `fwrite` — запись данных в файл
5. `fseek` — перемещение указателя внутри файла
6. `fprintf` — запись форматированных данных в файл
7. `fscanf` — чтение форматированных данных из файла
8. `feof` — проверка конца файла
9. `ferror` — проверка ошибок при работе с файлом

Открытие файла: `fopen`

Функция `fopen` используется для открытия файла. Она возвращает указатель типа `FILE*`, который затем используется для всех последующих операций с файлом.

Синтаксис:

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

- `filename` — имя файла, который нужно открыть.
- `mode` — режим открытия файла:
  - `"r"` — открыть для чтения (файл должен существовать).
  - `"w"` — открыть для записи (создаёт новый файл или очищает существующий).
  - `"a"` — открыть для добавления данных в конец файла (создаёт файл, если он не существует).
  - `"r+"` — открыть для чтения и записи.
  - `"w+"` — открыть для чтения и записи (создаёт новый файл или очищает существующий).
  - `"a+"` — открыть для чтения и добавления данных.

Пример:

```
FILE *file = fopen("example.txt", "r");
if (file == NULL) {
    printf("Ошибка при открытии файла!\n");
}
```

Закрытие файла: `fclose`

После завершения работы с файлом его нужно закрыть, чтобы освободить ресурсы. Для этого используется функция `fclose`.

Синтаксис:

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

Пример:

```
fclose(file);
```

Чтение из файла: `fread`

Функция `fread` используется для чтения бинарных данных из файла.

Синтаксис:

```
size_t fread(void *ptr, size_t size, size_t count, FILE *stream);
```

- `ptr` — указатель на буфер, куда будут записаны данные.
- `size` — размер одного элемента.
- `count` — количество элементов для чтения.
- `stream` — указатель на файл.

Пример:

```
FILE *file = fopen("example.txt", "rb");
if (file != NULL) {
    char buffer[100];
    fread(buffer, sizeof(char), 100, file);
    printf("%s\n", buffer);
    fclose(file);
}
```

Запись в файл: `fwrite`

Функция `fwrite` используется для записи бинарных данных в файл.

Синтаксис:

```
size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream);
```

Пример:

```
FILE *file = fopen("output.txt", "wb");
if (file != NULL) {
    char data[] = "Hello, World!";
    fwrite(data, sizeof(char), sizeof(data), file);
    fclose(file);
}
```

Форматированная запись и чтение: `fprintf` и `fscanf`

Функции `fprintf` и `fscanf` позволяют записывать и читать форматированные данные, аналогично тому, как это делается с функциями `printf` и `scanf` для консоли.

Запись:

```
FILE *file = fopen("data.txt", "w");
if (file != NULL) {
    fprintf(file, "Name: %s, Age: %d\n", "Alice", 25);
    fclose(file);
}
```

Чтение:

```
FILE *file = fopen("data.txt", "r");
if (file != NULL) {
    char name[50];
    int age;
    fscanf(file, "Name: %s, Age: %d", name, &age);
    printf("Name: %s, Age: %d\n", name, age);
    fclose(file);
}
```

Пример программы

Пример программы, которая читает файл построчно и выводит его содержимое на экран:

```
#include <stdio.h>

int main() {
    FILE *file = fopen("example.txt", "r");
    if (file == NULL) {
        printf("Не удалось открыть файл.\n");
```

```
        return 1;
    }

    char line[256];
    while (fgets(line, sizeof(line), file)) {
        printf("%s", line);
    }

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

Этот код открывает файл `example.txt`, читает его построчно с помощью `fgets`, выводит каждую строку на экран и затем закрывает файл.

Завершение работы

Работа с файлами в C важна для взаимодействия программ с внешними данными. Важно не забывать проверять успешность открытия файлов и корректно закрывать их после завершения работы.