9. Ввод/вывод на С. Бинарные файлы

FILE, fopen, fclose, r/w, t/b, буферизация

FILE – структура, описывающая абстракцию для ввода-вывода. Внутри:

1) дескриптор – идентификатор (целое число) файла внутри ОС
2) промежуточный буфер – быстрее накопить буфер, а потом за один системный вызов записать его на диск, чем для каждого байта делать отдельный системный вызов
3) текущее положение в файле
4) индикатор ошибки – была ли ошибка при последней операции
5) индикатор конца файла – достигнут ли конец файла при последней операции

Напрямую с этими полями не работают, а используют функции stdio.

Бинарный формат файла:

1) сложные форматы (bmp, wav, elf), для работы нужно описание
2) пример: заголовок: первые 4 байта ширина, вторые 4 – высота
3) сложно интерпретировать, но компактный размер файла

FILE* f = fopen(“in.txt”, mode);
mode: rb/wb/ab == читать/перезаписать/добавить в конец
rt – в Windows при записи ‘\n’ писать 10 13

fopen: Для открытия бинарного файла используется функция fopen(). В режиме доступа (второй аргумент fopen) нужно обязательно добавить "b" к соответствующему режиму. Например:

* "rb": Открыть файл для чтения в бинарном режиме.
* "wb": Открыть файл для записи в бинарном режиме.
* "ab": Открыть файл для добавления в бинарном режиме.
* "r+b": Открыть файл для чтения и записи в бинарном режиме.
* "w+b": Открыть файл для чтения и записи (удаляя содержимое) в бинарном режиме.
* "a+b": Открыть файл для чтения и добавления в бинарном режиме.

fclose: Функция fclose() используется для закрытия файла. Она также работает одинаково как для текстовых, так и для бинарных файлов.

- t/b (текстовый/бинарный режим):

  Текстовый режим (без "b"):

  * Предназначен для работы с текстовыми файлами.
  * Символы новой строки могут преобразовываться (\n в Unix/Linux или \r\n в Windows).

  Бинарный режим (с "b"):

  * Предназначен для работы с бинарными файлами (данные без интерпретации).
  * Данные передаются как есть, без преобразований.

Буферизация – это техника, при которой данные временно хранятся в буфере (области памяти) перед тем, как они будут переданы на целевое устройство (например, на диск) или извлечены из него (например, из файла).

- Стандартная библиотека ввода/вывода в C (stdio.h) использует буферизацию для работы с файлами и стандартными потоками ввода/вывода.
- fopen (и fclose) управляют файловыми буферами.
- printf, scanf, fprintf, fscanf, fgets, fputs и т.д. используют буферы.

fread, fwrite, fseek, ftell, fflush

1. fread (чтение данных из файла):

• Функция fread читает блок данных из указанного файлового потока в буфер в памяти.

- size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

ptr: Указатель на буфер (область памяти), куда будут скопированы прочитанные данные, size: Размер одного элемента данных (в байтах), nmemb: Количество элементов, которые нужно прочитать, stream: Указатель на структуру FILE (файловый поток), из которого нужно читать. Возвращает количество успешно прочитанных элементов.

2. fwrite (запись данных в файл):

- Функция fwrite записывает блок данных из буфера в памяти в указанный файловый поток.
- size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
- Возвращает количество успешно записанных элементов.

3. fseek (изменение позиции в файле):

- Перемещает указатель текущей позиции в файле.
- Синтаксис: int fseek(FILE *stream, long int offset, int whence);
- whence: SEEK_SET (начало файла), SEEK_CUR (текущая позиция), SEEK_END (конец файла).
- Возвращает 0 при успехе, ненулевое значение при ошибке.

4. ftell (получение текущей позиции в файле):

- возвращает текущую позицию указателя в файле в байтах относительно начала файла.
- long int ftell(FILE *stream);
- Возвращает позицию в байтах или -1L в случае ошибки.

5. fflush (сброс буфера):

- Функция fflush сбрасывает буфер, связанный с указанным файловым потоком. Если буфер используется для записи, fflush гарантирует, что все данные из буфера будут записаны на диск или другое устройство. Если буфер используется для чтения, поведение fflush не определено (но в некоторых реализациях может сбросить буфер).

- int fflush(FILE *stream);

  * stream: Указатель на структуру FILE (файловый поток), для которого нужно сбросить буфер.

- Если stream равен NULL, fflush сбросит все буферы записи всех открытых файловых потоков.

- возвращает 0 в случае успеха. В случае ошибки возвращает EOF.

обработка ошибок, feof, ferror

Игнорирование ошибок может привести к непредсказуемому поведению программы, зависанию, потере данных или даже сбою. Правильная обработка ошибок позволяет:

- Сделать программу более надежной и устойчивой.

- Предотвратить потерю данных.

- Информировать пользователя о возникших проблемах.

- Корректно завершить программу в случае критической ошибки.

- Улучшить отладку и сопровождение кода.

feof (определение конца файла):

- feof проверяет, достигнут ли конец файла при чтении данных из файлового потока.

- int feof(FILE *stream);

  * stream: Указатель на структуру FILE (файловый поток).

- возвращает ненулевое значение (истина), если достигнут конец файла, и 0 (ложь), если нет.

ferror (проверка ошибки файлового потока):

- ferror проверяет, возникла ли ошибка при работе с файловым потоком.

- int ferror(FILE *stream);

  * stream: Указатель на структуру FILE (файловый поток).

- возвращает ненулевое значение (истина), если произошла ошибка, и 0 (ложь), если ошибки нет.

feof – возвращает индикатор конца файла

ferror – возвращает индикатор ошибки

ferror, feof – была ли при последней операции с файлом получена ошибка/достигнут конец файла