acm.mipt.ru

олимпиады по программированию на Физтехе

Работа с файлами

Рассмотрим методы, с помощью которых можно создавать файлы и писать в них данные, и читать данные из существующих файлов.

Создание файлов

```
#include <stdio.h>
int main() {
    FILE *f;
    f = fopen("name.txt", "w");
    // ... работа с файлом
    fclose(f);
}
```

Эта программа создает файл "name.txt" в текущей директории. Если этот файл уже там присутствует, то она очищает его (обнуляет, то есть делает его пустым).

Первый аргумент у функции fopen – это путь к файлу (имя файла)

Второй аргумент – это спецификация режима, в котором будет открыт файл:

- **w+** – для записи и чтения; если указанного файла нет, то fopen создаст его, а если есть – обнулит его;
- **w** – для записи; если указанного файла нет, то fopen создаст его, а если есть – обнулит его;
- **r** – для чтения; если указанного файла нет, fopen не будет его создавать и вернет NULL;
- **r+** – для чтения и записи; если указанного файла нет, fopen не будет его создавать и вернет NULL;
- **a** – для дописывания; если указанного файла нет, fopen создаст его; указатель потока помещается в конец файла;
- **a+** – для чтения и дописывания; если указанного файла нет, fopen создаст его; указатель потока помещается в конец файла;

При успешном завершении fopen возвращают указатель FILE. Иначе возвращается NULL, а в глобальную переменную errno записывается код ошибки.

Чтение и запись

```
/* Program: files.c
   Build me with
     gcc -o files files.c
*/
#include <stdio.h>
int main() {
  FILE *f_in, *f_out;
  f_in = fopen("in.txt", "r");
  f_out = fopen("out.txt", "w+");
  if (f_in == NULL && f_out == NULL) {
      fprintf (stderr, "Can't open one of two files\n");
  }
  fscanf (f_in, "%d%d", &a, &b);
  fprintf (fout, "%d", a + b);
  fclose (f_in);
  fclose (f_out);
```

Вопросы и задания

- Что делает данная программа?
- Создайте файл "in.txt" с помощью команды bash:

  ```
  echo "12 14" > in.txt
  ```

- Что будет происходить с содержимым файла "out.txt" если эту программу запускать повторно
- Попробуйте f_in и f_out сделать одним и тем же файлом, который один раз открывается в режиме "r+".

Выдержка из страницы помощи по fopen

Наберите команду

```
bash$ man 3 fopen
```

и вы увидете описание функции fopen

Вот выдержка из этого описания, касаемая режимов открывания файлов (второй аргумент):

- **r** – Open text file for reading. The stream is positioned at the beginning of the file.
- **r+** – Open for reading and writing. The stream is positioned at the beginning of the file.
- **w** – Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
- **w+** – Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
- **a** – Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
- **a+** – Open for reading and appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.

Чтение и запись (примечание редактора) – корректная обработка ошибок

В коде, представленном в разделе **Чтение и запись** проанализируйте:

- печатается ли причина ошибки открытия файла? (Например, файла не существует).
- могут ли два файла не открыться по *разным* причинам?
- является ли хорошим стилем программирования сообщение с именем одного файла и системным сообщением об ошибке для абсолютно другого файла?

- f_in = NULL && f_out = NULL

Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

будет ли выражение истинным, если один файл удалось открыть, а второй - нет? Иногда fprintf(NULL, ...) или fscanf(NULL,...) приводит к аварийной остановке программы. Стоит ли выполнять эти функции с такими параметрами?

Некоторые стандартные функции языка C, если не могут сделать некоторое действие, например, открыть файл, пишут **номер ошибки** в глобальную переменную `errno` . Выводить в виде сообщения об ошибке открытия файла имя файла и номер ошибки - негуманно. Пользователи не обязаны заучивать коды ошибок. Более того, на разных операционных системах разные числа кодируют разные ошибки, и одна и та же ошибка (нет файла) кодируется в разных ОС разными числами.

Правильнее писать системное сообщение об ошибке. Для этого используют стандартную функцию языка C void perror(const char *s), которая печатает сначала переданную в параметре строку, потом через двоеточие сообщение об ошибке, номер которой хранится в `errno`.

Заметим, что если последняя вызываемая стандартная функция отработала без ошибок, она не обязана сбрасывать значение `errno` в состояние "без ошибок" и в этой переменной может хранится номер ошибки, который записала другая функция.

Не все функции изменяют значение `errno` в случае ошибки. Обычно это указывается в справке по функции.

Перепишем код чтения и записи в файл с диагностикой ошибок при открытии файлов.

```
/* Program: files.c
   Build me with
     gcc -o files files.c
*/
#include <stdio.h>
#include <error.h>       // чтобы заработал perror
int main() {
  FILE *f_in = NULL, *f_out = NULL;
  f_in = fopen("in.txt", "r");
  if (f_in == NULL) {
      perror("in.txt");  // печатаем ошибку открытия файла на чтение, быть может его нет; или файл есть, а у вас нет прав на чтени
      return 7;          // даже если тесты проверяющей системой не показаны, код возврата в тесте показан всегда
  }
  f_out = fopen("out.txt", "w+");
  if (f_out == NULL) {
      perror("out.txt"); // если файла нет, то его создадут. А если нет прав создать этот файл? Тогда получим диагностику, что не
      fclose(f_in);        // хороший тон - закрыть уже открытые нами потоки
      return 8;          // даже если тесты проверяющей системой не показаны, код возврата в тесте показан всегда
  }

  fscanf (f_in, "%d%d", &a, &b);
  fprintf (fout, "%d", a + b);

  fclose (f_in);
  fclose (f_out);
```

-- ArtemVoroztsov - 24 Mar 2005