

Семинар #9: Аргументы командной строки. Файлы.

Функции `sprintf` и `sscanf`.

На предыдущих занятиях мы прошли функции ввода/вывода из стандартных потоков: `printf` и `scanf`. А также функции для ввода/вывода из файлов: `fprintf` и `fscanf`. Функции `sprintf` и `sscanf` предназначены для ввода/вывода в строку или из строки. Пример использования можно посмотреть в файле `0sscanf.c`.

- **Задача #1: Конвертация переменных в строки и обратно**

```
char str1[10] = "79";  
char str2[10] = "435";
```

Конвертируйте эти 2 строки в числа, сложите и напечатайте результат

Аргументы командной строки

Программы могут принимать аргументы. Простейший пример – утилита `ls`. Если запустить `ls` без аргументов, то она просто напечатает содержимое текущей директории. Если же использовать эту программу с опцией `-l`: `ls -l`, то на экран выведется подробное описание файлов и папок в текущей директории. Другой пример - опция для компилятора `gcc -std=c99`.

- **Задача #2: `xxd`** - это простая программа, которая выводит на экран всё содержимое файла побайтово. Если, например, запустить программу следующим образом: `xxd a.out`, то она выведет на экран всё содержимое этого исполняемого файла. Часто используемые опции командной строки: `-h` (сокращение от help) и `-v` (сокращение от version).
 - Запустите `xxd` с аргументом – именем файла `hello.txt`. Этот файл содержит лишь строку `Hello`. `xxd` покажет вам содержимое этого файла в шестнадцатеричном виде и в виде ASCII.
 - Запустите `xxd` с опцией `-h`.
 - Запустите `xxd` с нужной опцией, чтобы вывод файла `hello.txt` был представлен в двоичном виде.
 - * Если файл большой, то весь вывод `xxd` не поместится на экран. Перенаправить вывод в нужный файл можно следующим образом: `xxd a.out > temp.txt`. После этого в файле `temp.txt` будет храниться всё, что было бы напечатано на экран.
 - * Создайте программу Hello World и скомпилируйте её в файл `a.out`. Сохраните вывод `xxd ./a.out` в отдельном файле `hw.txt`. Измените файл `hw.txt`, так чтобы программа печатала Hello MIPT. Создайте исполняемый файл из файла `hw.txt`, используя `xxd` с опцией `-r`.
- **Задача #3: `argc`**: Простейшая программа `1argc.c` печатает количество аргументов командной строки. Скомпилируйте эту программу и протестируйте её, запуская с разным количеством аргументов.
- **Задача #4: `argv`**: Простейшая программа `2argv.c` печатает аргументы командной строки. Скомпилируйте эту программу и протестируйте её, запуская с разным количеством аргументов.

```
#include <stdio.h>  
int main(int argc, char** argv)  
{  
    printf("Number of arguments = %d\n", argc);  
    for (int i = 0; i < argc; ++i)  
        printf("Argument #%d = %s\n", i, argv[i]);  
}
```

- **Задача #5: Сумма аргументов**: Создайте программу `sum`, которая будет печатать сумму всех аргументов. Например, при вызове `./sum 4 8 15 16 23 42` программа должна напечатать 108

Функции для работы с файлами

- **fopen:** Открывает файл для чтения/записи

Режимы открытия файла:

r	открыть существующий файл для чтения (read)
w	создать новый файл и открыть его для записи (write) если файл уже существует, то он удалится перед записью
a	открыть для записи в конец файла (append)
r+	открыть для чтения/записи, с начала файла
w+	создать новый файл и открыть его для чтения/записи
a+	открыть для чтения/записи в конец файла

Для бинарных файлов в Windows нужно добавить символ **b**.

- **fopen:** Закрывает файл
- **fprintf/fscanf:** Функции работают аналогично **printf/scanf**, только первым аргументом нужно передать файл (указатель на структуру **FILE**).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE* fp = fopen("myfile.txt", "w");
    if (fp == NULL)
    {
        printf("Error!\n");
        exit(1);
    }
    fprintf(fp, "Hello world!");
    fclose(fp);
}
```

- Создаём и открываем файл "myfile.txt" на запись.
- Проверяем получилось ли открыть файл, если нет, то пишем сообщение об ошибке и выходим. В дальнейших примерах эта проверка будет опускаться для экономии места.
- Если получилось открыть, то записываем в файл строку с помощью **fprintf**.
- Закрываем файл.

Задача #6: Скомпилируйте программу **3fprintf.c** и запустите. В результате выполнения программы должен появиться файл **myfile.txt** с содержимым **Hello world!**.

- **fwrite/fread** - **бинарное чтение/запись:** **fwrite** записывает некоторый участок памяти в файл без обработки. **fread** считывает данные из файла в память без обработки.

Пример. Записываем 4 байта памяти переменной **a** в файл **binary.dat**:

```
#include <stdio.h>
int main()
{
    int a = 287454020; // Это число = 0x11223344 в шестнадцатеричной системе
    FILE* fb = fopen("binary.dat", "w");
    fwrite(&a, sizeof(int), 1, fb);
    fclose(fb);
}
```

Задача #7: Печать в текстовом и бинарном виде:

В файле **4text_and_binary.c** содержится пример записи числа в текстовом и бинарном виде. Скомпилируйте эту программу и запустите. Должно появиться 2 файла (**number.txt** и **number.bin**). Изучите оба эти файла, открывая их в текстовом редакторе, а также с помощью утилиты **xxd**. Объясните результат.

Задача #8: Печать массива в бинарном виде:

Пусть есть массив из чисел типа `int`: `int array[5] = {111, 222, 333, 444, 555};`

Запишите эти числа в текстовый файл `array.txt`, используя `fprintf`. Изучите содержимое этого файла побайтово с помощью `xxd`.

Запишите эти числа в бинарный файл `array.bin`, используя `fwrite`. Изучите содержимое этого файла побайтово с помощью `xxd`.

- **fgetc** - посимвольное чтение из файла - возвращает ASCII код следующего символа из файла. Если символов не осталось, то она возвращает константу EOF равную -1.

Пример программы, которая находит количество цифр в файле:

```
#include <stdio.h>
int main()
{
    FILE* f = fopen("input.txt", "r");
    int c;
    int num_of_digits = 0;

    while ((c = fgetc(f)) != EOF)
    {
        if (c >= '0' && c <= '9')
            num_of_digits += 1;
    }
    printf("Number of digits = %d\n", num_of_digits);
    fclose(f);
}
```

Эта программа содержится в файле `5number_of_digits.c`

- **Задача #9:** Написать программу `symbolcount`, которая считает количество символов в файле. название файла должно передаваться через аргумент командной строки:

```
gcc -o symbolcount main.c
./symbolcount war_and_peace.txt
3332371
```

- **Задача #10:** Написать программу `linecount`, которая находит количество строк в файле.

- **Задача #11:** Написать программу `wordcount`, которая находит количество слов в файле. Слово это любая последовательность символов, разделённая *одним или несколькими* пробельными символами. Пробельные символы это пробел, перенос на новую строку (`\n`) либо табуляция (`\t`).

- **ftell:** Функция, которая возвращает текущее положение в файле. Например, если мы начали считать с начала файла и считали 10 символов, то эта функция вернёт 10.
- **fseek:** Функция, которая устанавливает положение в файле. Например, следующая строка устанавливает положение на 100-й символ:

```
fseek (fout, 100, SEEK_SET);
```

Следующие функции будут считывать всё начиная с 101-го символа.

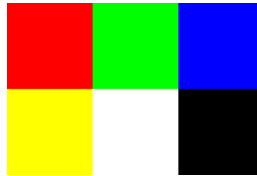
Работа с изображениями формата .ppm

Простейший формат для изображения имеет следующую структуру

```
P3
3 2
255
255 0 0
0 255 0
0 0 255
255 255 0
255 255 255
0 0 0
```

- В первой строке задаётся тип файла P3 - означает, что в этом файле будет храниться цветное изображение, причём значения пикселей будет задаваться в текстовом формате.
- Во второй строке задаются размеры картинки - 3 на 2 пикселя.
- Во третьей строке задаётся максимальное значение RGB компоненты цвета.
- Дальше идут RGB компоненты цветов каждого пикселя в текстовом формате.

Картинка имеет следующий вид:



- **Задача #12:** Написать программу, которая генерирует одноцветную картинку (500 на 500) в формате .ppm. Цвет должен передаваться через аргументы командной строки.
- **Задача #13: Белый шум:** Написать программу, которая случайное изображение в формате .ppm. Цвет каждого пикселя задаётся случайно.
- **Задача #14: Градиент:** Написать программу, которая генерирует градиентную картинку в формате .ppm. Два цвета должны передаваться через аргументы командной строки.
- **Задача #15: Черно-белая картинка:** Написать программу, которая считывает изображение в формате .ppm и сохраняет его в черно-белом виде. Файл изображения должен передаваться через аргументы командной строки. Читайте файл `russian_peasants_1909.ppm` и сделайте его черно-белым.