Воспоминания об stdio.

Евгений Линский

stdio

```
1 FILE* f1 = fopen("in.txt",....);// файл на диске
2 FILE* f2 = stdin; // можно читать с клавиатуры
3 FILE* f3 = stdout; // можно писать на экран
```

FILE — структура, описывающая абстракцию для ввода-вывода (файл на диске, клавиатура, экран). Что 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.

Текстовые и бинарные файла

1. На диске всегда байты, меняется только способ их интерпретации
2. Текстовый формат файла
   1. Интерпретируется как последовательность символов. Пример: число 100 записывается не как один байт, а как три символа '1' '0' '0' (3 байта).
   2. Есть спецсимволы: перевод строки, табуляция.
   3. Проблемы: разные кодировки, в том числе для спецсимволов (перевод строки '\n': Linux — 10, Windows — 10 13)
   4. Просто интерпретировать, но большой размер файла.
3. Бинарный формат файла
   1. Сложные форматы (bmp, wav, elf), для работы нужно описание. Пример: число 100 — как один байт.
   2. Еще пример. Заголовок: первые 4 байта ширина, вторые четыре байта высота. Данные: три байта RGB с выравниванием.
   3. Сложно интерпретировать, но компактный размер файла.

stdio

FILE — структура, описывающая абстракцию для ввода-вывода (файл на диске, клавиатура, экран).

```
FILE *f1 = fopen("in.txt",....);// файл на диске
FILE *f2 = stdin; // можно читать с клавиатуры
FILE *f3 = stdout; // можно писать на экран
```

stdin, stdout — глобальные переменные в стандартной библиотеке libc.

- getchar – fgetc(stdin), putchar – fputc(stdout) – символы
- gets, puts – строки (fgets, fputs)
- printf, scanf – форматный ввод (fprinf, fscanf)

Считать строку

```
char s[100];
gets(s);
scanf("%s", s);
```

Все ли хорошо?

Считать строку

```
char s[100];

while (ch != '\n' && i < 99 ) {
  s[i] = getchar();
  i++;
}
s[i] = 0;
```

```
gets(s); // Max size!!!
scanf("%s", s); // Max size!!!
fgets(s, 99, stdin);
scanf("%99s", s);
scanf_s("%99s", name, 100); // Microsoft version
```

printf/fprintf/sprintf

```
fprintf(stdout, ...); //printf
fscanf(stdin, ....); //scanf
char s1[] = "3  4";
sscanf(s1,"%d %d", &a, &b);
char s2[256];
sprintf(s2, "%d + %d = %d", a, b, c);
```

1. Все это небыстро, т.к. внутри функции нужно разобрать форматную строку
2. Технология: функция с переменным числом параметров (см. *va_arg*)

https://en.cppreference.com/w/cpp/io/c/fprintf

Переменное число аргументов в C (printf)

va_start, va_arg, va_end — макросы.

```
1  void simple_printf(const char* fmt, ...) {
2    va_list args;
3    //записать в args адрес следующего за fmt параметра на стеке
4    va_start(args, fmt);
5    while(*fmt != '\0') {
6      if(*fmt=='d') {
7        //достать со стека переменную типа int
8        int i = va_arg(args, int)
9        // здесь должен быть код, который
10       // выводит int на экран с помощью putc
11     }
12     fmt++;
13   }
14   va_end(args);
15 }
16 //Труднообнаруживаемые ошибки
17 printf("%s", 5);
18 printf("%d %d", 4); printf("%d", 4, 5);
```