Лекция 7. Обзор libc: stdio - II, stdlib

Евгений Линский

fopen

```
1  FILE*  f = fopen("in.txt", mode);
2  if( f == NULL ) {
3    // файл не открылся
4  }
5  fclose(f);
```

mode: r/w/a == читать/перезаписать/добавить в конец.
rt — в Windows при записи '\n' писать 10 13

1. Зачем делать fclose, если при закрытии программы все ресурсы ОС и так освободит?

fopen

```
1  FILE*  f = fopen("in.txt", mode);
2  if( f == NULL ) {
3    // файл не открылся
4  }
5  fclose(f);
```

mode: r/w/a == читать/перезаписать/добавить в конец.
rt — в Windows при записи '\n' писать 10 13

1. Зачем делать fclose, если при закрытии программы все ресурсы ОС и так освободит?
2. Число дескрипторов ограничено. На FILE тратится память.

fopen

```
1  FILE*  f = fopen("in.txt", mode);
2  if( f == NULL ) {
3    // файл не открылся
4  }
5  fclose(f);
```

mode: r/w/a == читать/перезаписать/добавить в конец.
rt — в Windows при записи '\n' писать 10 13

1. Зачем делать fclose, если при закрытии программы все ресурсы ОС и так освободит?
2. Число дескрипторов ограничено. На FILE тратится память.
3. Ограничения на работу с отрытым файлом (в Windows файл открытый на чтение нельзя удалить).

Текстовые файлы

```
int a = 0; int b = 0;
FILE* fin = fopen("in.txt", "r");
FILE* fout = fopen("out.txt", "w");
fscanf(fin, "%d %f", &a, &b);
fprintf(fout, "%d %f", a, b);
fclose(fout);
fclose(fin);
```

```
char s[256];
// считывает до whitespace ' ', '\n','\t'
fscanf(fin, "%s", s);
```

1. В чем проблема?

https://www.opennet.ru/base/sec/p49-14.txt.html

Текстовые файлы

```
int a = 0; int b = 0;
FILE* fin = fopen("in.txt", "r");
FILE* fout = fopen("out.txt", "w");
fscanf(fin, "%d %f", &a, &b);
fprintf(fout, "%d %f", a, b);
fclose(fout);
fclose(fin);
```

```
char s[256];
// считывает до whitespace ' ', '\n','\t'
fscanf(fin, "%s", s);
```

1. В чем проблема?
2. Не контролируется максимальное число считанных символов
   https://www.opennet.ru/base/sec/p49-14.txt.html

Текстовые файлы

```
1  int a = 0; int b = 0;
2  FILE* fin = fopen("in.txt", "r");
3  FILE* fout = fopen("out.txt", "w");
4  fscanf(fin, "%d %f", &a, &b);
5  fprintf(fout, "%d %f", a, b);
6  fclose(fout);
7  fclose(fin);
```

```
1  char s[256];
2  // считывает до whitespace ' ', '\n','\t'
3  fscanf(fin, "%s", s);
```

1. В чем проблема?
2. Не контролируется максимальное число считанных символов `https://www.opennet.ru/base/sec/p49-14.txt.html`
3. *fgets(s, 255, fin)* читает до '\n,' но не больше 255

Текстовые файлы

```
1  int a = 0; int b = 0;
2  FILE* fin = fopen("in.txt", "r");
3  FILE* fout = fopen("out.txt", "w");
4  fscanf(fin, "%d %f", &a, &b);
5  fprintf(fout, "%d %f", a, b);
6  fclose(fout);
7  fclose(fin);
```

```
1  char s[256];
2  // считывает до whitespace ' ', '\n','\t'
3  fscanf(fin, "%s", s);
```

1. В чем проблема?
2. Не контролируется максимальное число считанных символов `https://www.opennet.ru/base/sec/p49-14.txt.html`
3. *fgets(s, 255, fin)* читает до '\n,' но не больше 255
4. еще вариант *fscanf(fin, "%255s", s)*

printf/fprintf/sprintf

```
1  fprintf(stdout, ...); //printf
2  fscanf(stdin, ....); //scanf
3  char s1[] = "3  4";
4  sscanf(s2,"%d %d", &a, &b);
5  char s2[256];
6  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*)

Бинарные файлы

```
1  FILE* fin = fopen("in.txt", "r");
2  FILE* fout = fopen("out.txt", "w");
3  int array[100];
4  //размер одного элемента == 4, колво элементов == 100
5  fread(array, sizeof(int), 100, fin);
6  fwrite(array, sizeof(int), 50, fout);
7  fclose(fout);
8  fclose(fin);
```

А еще что можно?

1. fseek — переместиться на заданную позицию в файле (удобно пропускать ненужные поля в заголовке бинарного файла)
2. ftell — возвращает текущую позицию (как узнать размер файла?)

Ошибки

```
1  size_t res1 = fread(array, 100 * sizeof(int), 1, fin1);
2  size_t res2 = fread(array, sizeof(int), 100, fin2);
3  //int т.к. может быть EOF (обычно -1)
4  int res3 = fscanf(fin, "%d %f", &a, &b);
```

1. fread — число считанных **элементов**
2. fscanf — число считанных **элементов по формату**

Ошибки

```
1  while(!feof(fin)) {
2    fread(...)
3    if(ferror(fin)) {
4      ...
5    }
6  }
```

1. feof — возвращает индикатор конца файла
2. ferror — возвращает индикатор ошибки

Сначала *fread* перейдет чтением «за конец файла», а потом установит индикатор (не заранее).
ferror, feof — была ли при последней операции с файлом получена ошибка/достигнут конец файла.

fflush

```
1  int main() {
2    FILE* fout = fopen("...", "...");
3    ...
4    int a = 3; int b = 5;
5    fprintf(fout,  "%d %d", a, b);
6    int* array = malloc(10000000);
7    if(array == NULL) return -1
8    ...
9    fclose(fout);
10   return 0;
11 }
```

1. В чем проблема?

fflush

```
1  int main() {
2    FILE* fout = fopen("...", "...");
3    ...
4    int a = 3; int b = 5;
5    fprintf(fout,  "%d %d", a, b);
6    int* array = malloc(10000000);
7    if(array == NULL) return -1
8    ...
9    fclose(fout);
10   return 0;
11 }
```

1. В чем проблема?
2. "3 5" может осесть в буфере внутри FILE

fflush

```
1  int main() {
2    FILE* fout = fopen("...", "...");
3    ...
4    int a = 3; int b = 5;
5    fprintf(fout,  "%d %d", a, b);
6    int* array = malloc(10000000);
7    if(array == NULL) return -1
8    ...
9    fclose(fout);
10   return 0;
11 }
```

1. В чем проблема?
2. "3 5" может осесть в буфере внутри FILE
3. Можно вызвать *fflush(fout)* для принудительного сбрасывания буфера

Стандартная библиотека (libc)

1. `cplusplus.com`
2. `cppreference.com`
3. MSDN

Обзор:

- stdio.h — ввод/вывод (файл, клавиатура, экран)
- *stdlib.h* — работа с памятью, алгоритмы
- *string.h* — работа со строками и массивами
- math.h — математические функции
- *time.h* — время
- *assert.h*, *stdint.h*

stdlib

```
1  // array to integer
2  int res1 = atoi("abc"); // undef. behavior (usually 0)
3  int res2 = atoi("0");
```

```
1  // endPtr --- первый символ, на котором сломалось
2  long strtol(char *buffer, char **endPtr, int base);
3  char *end; char *ptr = "25a";
4  int N = strtol(ptr, &end, 10);
5  if (ptr == end) {
6
7  }
```

1. А еще можно с помощью?

stdlib

```
1  // array to integer
2  int res1 = atoi("abc"); // undef. behavior (usually 0)
3  int res2 = atoi("0");
```

```
1  // endPtr --- первый символ, на котором сломалось
2  long strtol(char *buffer, char **endPtr, int base);
3  char *end; char *ptr = "25a";
4  int N = strtol(ptr, &end, 10);
5  if (ptr == end) {
6
7  }
```

1. А еще можно с помощью?
2. sscanf

stdlib

time(NULL) — текущее время в секундах.

```
1    srand (time(NULL));
2    int r1 = rand();
3    srand (time(NULL));
4    int r2 = rand();
```

1. Что не так?

stdlib

time(NULL) — текущее время в секундах.

```
1    srand (time(NULL));
2    int r1 = rand();
3    srand (time(NULL));
4    int r2 = rand();
```

1. Что не так?
2. $r_i = rand(r_{i-1})$, $r_0 = srand()$; (псевдослучайная последовательность) srand и rand связаны через глобальную переменную.

stdlib

time(NULL) — текущее время в секундах.

```
1    srand (time(NULL));
2    int r1 = rand();
3    srand (time(NULL));
4    int r2 = rand();
```

1. Что не так?
2. $r_i = rand(r_{i-1})$, $r_0 = srand()$; (псевдослучайная последовательность) srand и rand связаны через глобальную переменную.
3. *time(NULL)* возвращается текущее время в секундах; выполнение программы займет меньше секунды.

stdlib

time(NULL) — текущее время в секундах.

```
1    srand (time(NULL));
2    int r1 = rand();
3    srand (time(NULL));
4    int r2 = rand();
```

1. Что не так?
2. $r_i = rand(r_{i-1})$, $r_0 = srand()$; (псевдослучайная последовательность) srand и rand связаны через глобальную переменную.
3. *time(NULL)* возвращается текущее время в секундах; выполнение программы займет меньше секунды.
4. Для отладки можно сделать *srand(3)* и всегда получать одну и ту же последовательность.

stdlib

```
void qsort (void* base, size_t num, size_t size,
            int (*compar)(const void*,const void*));

void* bsearch (const void* key, const void* base,
   size_t num, size_t size,
   int (*compar)(const void*,const void*));
```

```
ret = system("ls -l"); // return exit code
```

```
int* m = (int*) malloc(10000000);
if (m == NULL) {
  fprintf(stderr, "..."); //у stderr нет буфера
  exit( EXIT_FAILURE); //clean up: flushing buffers, etc
  // void abort (void); //without clean up
}
```