



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ \_\_\_\_\_ «Информатика и системы управления»  
КАФЕДРА \_\_\_\_\_ «Программное обеспечение ЭВМ и информационные технологии»

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 по курсу «Операционные системы»

«Буферизованный и не буферизованный ввод-вывод»

Студент \_\_\_\_\_ Маслова Марина Дмитриевна  
Группа \_\_\_\_\_ ИУ7-63Б  
Оценка (баллы) \_\_\_\_\_  
Преподаватель \_\_\_\_\_ Рязанова Наталья Юрьевна

2022 г.

# 1 Структура FILE

Листинг 1.1 – Описание структуры FILE в файле /usr/include/bits/types/FILE.h

```
1 typedef struct _IO_FILE FILE;
```

Листинг 1.2 – Описание структуры \_IO\_FILE в файле /usr/include/bits/types/struct\_FILE.h

```
1 struct _IO_FILE
2 {
3     int _flags;          /* High-order word is _IO_MAGIC; rest is flags. */
4
5     /* The following pointers correspond to the C++ streambuf protocol. */
6     char *_IO_read_ptr;  /* Current read pointer */
7     char *_IO_read_end;  /* End of get area. */
8     char *_IO_read_base; /* Start of putback+get area. */
9     char *_IO_write_base; /* Start of put area. */
10    char *_IO_write_ptr;  /* Current put pointer. */
11    char *_IO_write_end;  /* End of put area. */
12    char *_IO_buf_base;   /* Start of reserve area. */
13    char *_IO_buf_end;    /* End of reserve area. */
14
15    /* The following fields are used to support backing up and undo. */
16    char *_IO_save_base; /* Pointer to start of non-current get area. */
17    char *_IO_backup_base; /* Pointer to first valid character of backup area */
18    char *_IO_save_end; /* Pointer to end of non-current get area. */
19
20    struct _IO_marker *_markers;
21
22    struct _IO_FILE *_chain;
23
24    int _fileno;
25    int _flags2;
26    __off_t _old_offset; /* This used to be _offset but it's too small. */
27
28    /* 1+column number of pbase(); 0 is unknown. */
29    unsigned short _cur_column;
30    signed char _vtable_offset;
31    char _shortbuf[1];
32
33    _IO_lock_t *_lock;
34 #ifndef _IO_USE_OLD_IO_FILE
35 };
```

## 2 Первая программа

Листинг 2.1 – Код первой программы. Один поток

```
1 #include <stdio.h>
2 #include <fcntl.h>
3
4 #define GREEN "\033[01;38;05;46m"
5 #define BLUE  "\033[01;38;05;33m"
6 #define CLEAR "\033[0m"
7
8 int main(void)
9 {
10     int fd = open("alphabet.txt", O_RDONLY);
11
12     FILE *fs1 = fdopen(fd, "r");
13     char buff1[20];
14     setvbuf(fs1, buff1, _IOFBF, 20);
15
16     FILE *fs2 = fdopen(fd, "r");
17     char buff2[20];
18     setvbuf(fs2, buff2, _IOFBF, 20);
19
20     int flag1 = 1, flag2 = 1;
21
22     while (flag1 == 1 || flag2 == 1)
23     {
24         char c;
25
26         if ((flag1 = fscanf(fs1, "%c", &c)) == 1)
27             fprintf(stdout, GREEN "%c" CLEAR, c);
28
29         if ((flag2 = fscanf(fs2, "%c", &c)) == 1)
30             fprintf(stdout, BLUE "%c" CLEAR, c);
31     }
32
33     fprintf(stdout, "\n");
34     return 0;
35 }
```

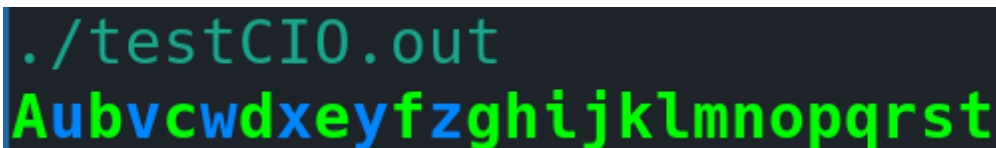


Рисунок 2.1 – Результат работы первой программы. Один поток

## Листинг 2.2 – Код первой программы. Два потока

```
1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <pthread.h>
4
5 #define GREEN "\033[01;38;05;46m"
6 #define BLUE  "\033[01;38;05;33m"
7 #define CLEAR "\033[0m"
8
9 struct args_struct { FILE * fs; char * color; };
10
11 void *read_buf(void *args)
12 {
13     struct args_struct *cur_args = (struct args_struct *) args;
14     FILE *fs = cur_args->fs;
15     char *color = cur_args->color;
16     int flag = 1;
17
18     while (flag == 1)
19     {
20         char c;
21         if ((flag = fscanf(fs, "%c", &c)) == 1)
22             fprintf(stdout, "%s%c" CLEAR, color, c);
23     }
24     return NULL;
25 }
26
27 int main(void)
28 {
29     int fd = open("alphabet.txt", O_RDONLY);
30
31     FILE *fs1 = fdopen(fd, "r");
32     char buff1[20];
33     setvbuf(fs1, buff1, _IOFBF, 20);
34     struct args_struct args1 = { .fs = fs1, .color = GREEN };
35
36     FILE *fs2 = fdopen(fd, "r");
37     char buff2[20];
38     setvbuf(fs2, buff2, _IOFBF, 20);
39     struct args_struct args2 = { .fs = fs2, .color = BLUE };
40
41     pthread_t td;
42     pthread_create(&td, NULL, read_buf, &args2);
43     read_buf(&args1);
44     pthread_join(td, NULL);
45     puts("");
46     return 0;
47 }
```

```
./testCI0ths.out  
Aubvwcdxeyzfg hijklmnopqrst
```

Рисунок 2.2 – Результат работы первой программы. Два потока

### Анализ результата

Системный вызов open

Вызовы fdopen

Вызовы setvbuf

fscanf и fprintf

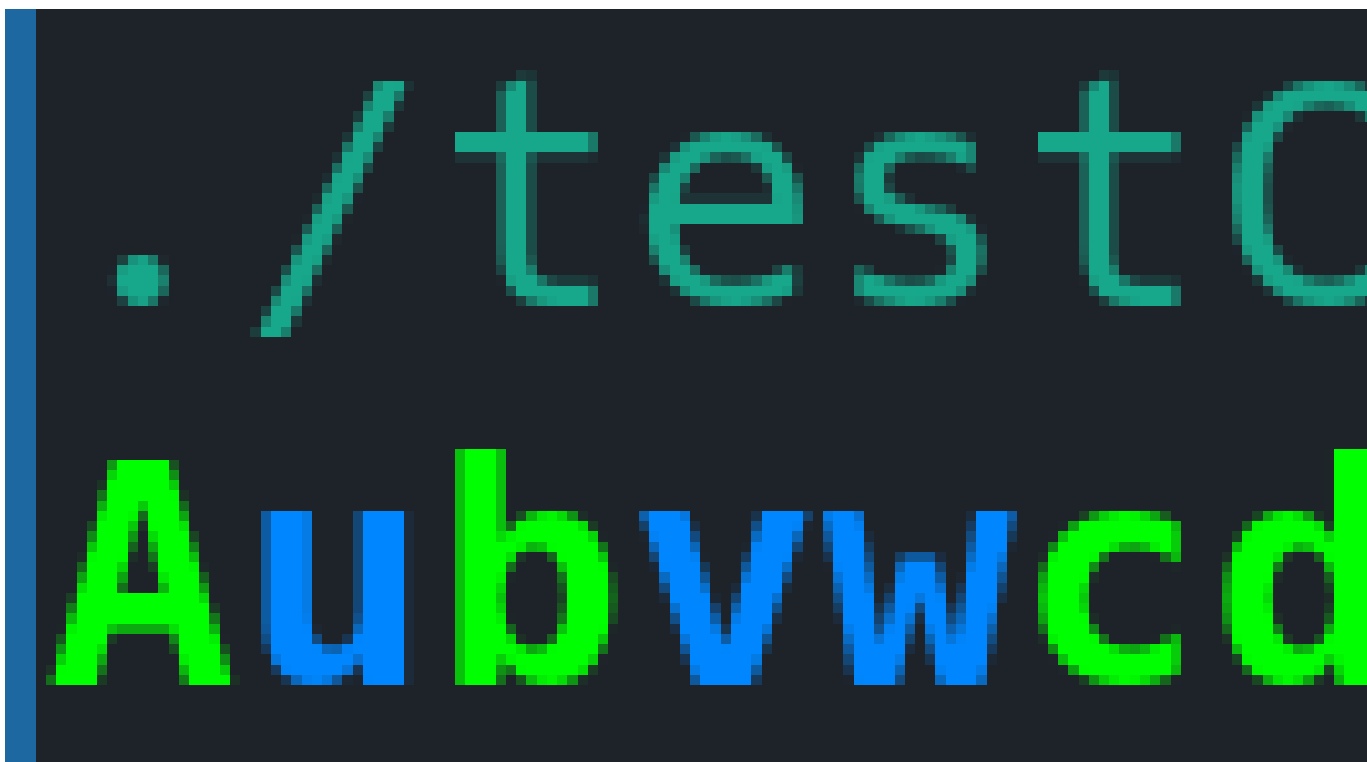
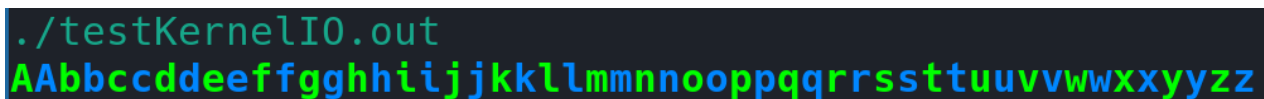


Рисунок 2.3 – Схема связей структур

### 3 Вторая программа

Листинг 3.1 – Код второй программы. Один поток

```
1 #include <fcntl.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 #define GREEN "\033[01;38;05;46m"
6 #define BLUE  "\033[01;38;05;33m"
7 #define CLEAR "\033[0m"
8
9 int main()
10 {
11     char c;
12     int fd1 = open("alphabet.txt", O_RDONLY);
13     int fd2 = open("alphabet.txt", O_RDONLY);
14     int flag1 = 1, flag2 = 1;
15
16     while(flag1 == 1 || flag2 == 1)
17     {
18         if ((flag1 = read(fd1, &c, 1)) == 1)
19             printf(GREEN "%c" CLEAR, c);
20
21         if ((flag2 = read(fd2, &c, 1)) == 1)
22             printf(BLUE "%c" CLEAR, c);
23     }
24
25     puts("");
26     return 0;
27 }
```



```
./testKernelIO.out
AAbbccddeeffgghhiijjkkllmmnnnooppqrrssttuuvvwwxxyyzz
```

Рисунок 3.1 – Результат работы второй программы. Один поток

### Листинг 3.2 – Код второй программы. Два потока

```
1 #include <fcntl.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <pthread.h>
5
6 #define GREEN "\033[01;38;05;46m"
7 #define BLUE  "\033[01;38;05;33m"
8 #define CLEAR "\033[0m"
9
10 struct args_struct { int fd; char * color; };
11
12 void *read_buf(void *args)
13 {
14     struct args_struct *cur_args = (struct args_struct *) args;
15     int fd = cur_args->fd;
16     char *color = cur_args->color;
17
18     int flag = 1;
19
20     while (flag == 1)
21     {
22         char c;
23         if ((flag = read(fd, &c, 1)) == 1)
24             printf("%s%c" CLEAR, color, c);
25     }
26
27     return NULL;
28 }
29
30 int main()
31 {
32     char c;
33     int fd1 = open("alphabet.txt", O_RDONLY);
34     struct args_struct args1 = { .fd = fd1, .color = GREEN };
35
36     int fd2 = open("alphabet.txt", O_RDONLY);
37     struct args_struct args2 = { .fd = fd2, .color = BLUE };
38
39     pthread_t td;
40     pthread_create(&td, NULL, read_buf, &args2);
41
42     read_buf(&args1);
43
44     pthread_join(td, NULL);
45     puts("");
46     return 0;
47 }
```

```
./testKernelIOths.out  
AAbbccddeffeghfigjkhiljkmnlmnoopqrpsqtuvrwsxtyzuvwxyz
```

Рисунок 3.2 – Результат работы второй программы. Два потока

### Анализ результата

Системный вызов open

Просто open, про два дескриптора

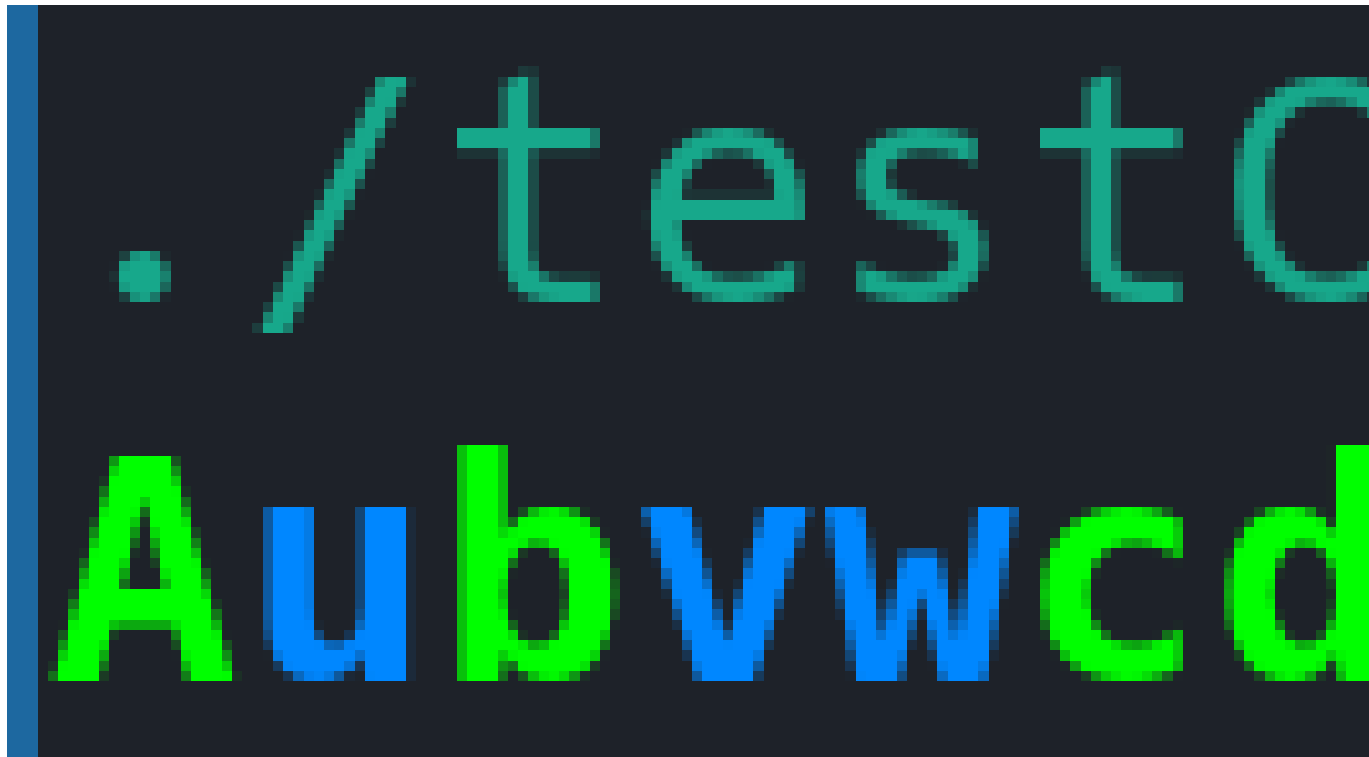


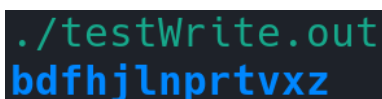
Рисунок 3.3 – Схема связей структур



## 4 Третья программа

Листинг 4.1 – Код третьей программы. Один поток

```
1 #include <stdio.h>
2 #include <sys/stat.h>
3
4 #define GREEN "\033[01;38;05;46m"
5 #define BLUE  "\033[01;38;05;33m"
6 #define CLEAR "\033[0m"
7
8 void fileInfo(FILE *fs)
9 {
10     struct stat statbuf;
11     stat("result.txt", &statbuf);
12     printf("\033[38;05;214minode: %ld\n", statbuf.st_ino);
13     printf("Общий размер в байтах: %ld\n", statbuf.st_size);
14     printf("Текущая позиция: %ld\n\n" CLEAR, ftell(fs));
15 }
16
17 int main(void)
18 {
19     FILE *fs1 = fopen("result.txt", "w");
20     fileInfo(fs1);
21
22     FILE *fs2 = fopen("result.txt", "w");
23     fileInfo(fs2);
24
25     for (char ch = 'a'; ch <= 'z'; ++ch)
26         fprintf(ch % 2 ? fs1 : fs2, "%s%c" CLEAR, ch % 2 ? GREEN : BLUE, ch);
27
28     fileInfo(fs1);
29     fclose(fs1);
30     fileInfo(fs1);
31
32     fileInfo(fs2);
33     fclose(fs2);
34     fileInfo(fs1);
35
36     return 0;
37 }
```



```
./testWrite.out
bdfhjlprtvxz
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
```

Рисунок 4.1 – Результат работы третьей программы. fs2 закрывается последним

```
./testWrite.out  
acegikmoqsuwy
```

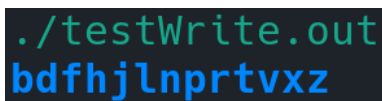
Рисунок 4.2 – Результат работы третьей программы. fs1 закрывается последним

```
inode: 4196184  
Общий размер в байтах: 0  
Текущая позиция: 0  
  
inode: 4196184  
Общий размер в байтах: 0  
Текущая позиция: 0  
  
inode: 4196184  
Общий размер в байтах: 0  
Текущая позиция: 247  
  
inode: 4196184  
Общий размер в байтах: 247  
Текущая позиция: -1  
  
inode: 4196184  
Общий размер в байтах: 247  
Текущая позиция: 247  
  
inode: 4196184  
Общий размер в байтах: 247  
Текущая позиция: -1
```

Рисунок 4.3 – Информация о состоянии открытых файлов

## Листинг 4.2 – Код третьей программы. Два потока

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <sys/stat.h>
4
5 #define GREEN "\033[01;38;05;46m"
6 #define BLUE  "\033[01;38;05;33m"
7 #define CLEAR "\033[0m"
8
9 struct args_struct { char begin; char * color; };
10
11 void *write_syms(void *args)
12 {
13     FILE *fs = fopen("resultths.txt", "w");
14
15     struct args_struct *cur_args = (struct args_struct *) args;
16     char begin = cur_args->begin;
17     char *color = cur_args->color;
18
19     for (char ch = begin; ch <= 'z'; ch += 2)
20         fprintf(fs, "%s%c" CLEAR, color, ch);
21
22     fclose(fs);
23     return NULL;
24 }
25
26 int main(void)
27 {
28     struct args_struct args1 = { .begin = 'a', .color = GREEN };
29     struct args_struct args2 = { .begin = 'b', .color = BLUE };
30
31     pthread_t td;
32     pthread_create(&td, NULL, write_syms, &args2);
33
34     write_syms(&args1);
35
36     pthread_join(td, NULL);
37     return 0;
38 }
```



```
./testWrite.out
bdfhjlnprtvxz
```

Рисунок 4.4 – Результат работы третьей программы.  
Последним вызывается fclose в вспомогательном потоке

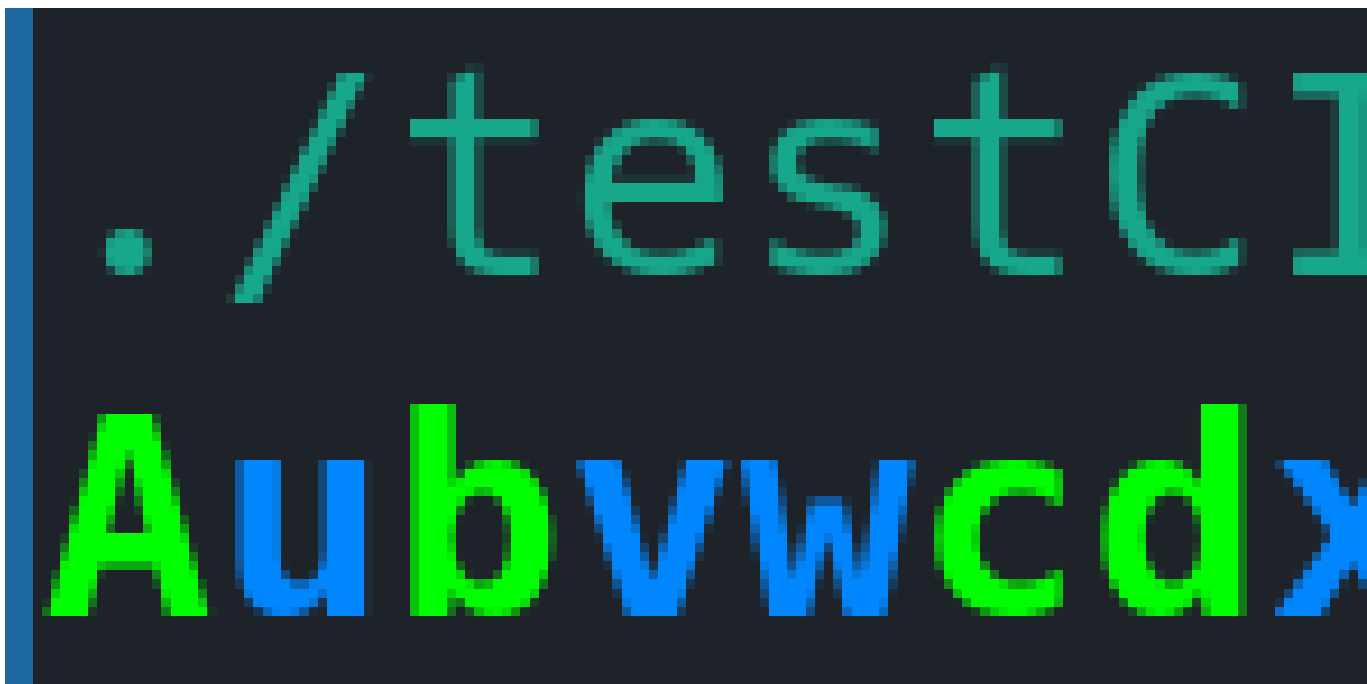
```
./testWrite.out  
acegikmoqsuwy
```

Рисунок 4.5 – Результат работы третьей программы.  
Последним вызывается `fclose` в главном потоке

### Анализ результата

Просто `open`, просто `fclose`

Перезапись



```
./testCI  
Aubvwcdx
```

Рисунок 4.6 – Схема связей структур