



**«Московский государственный технический
университет имени Н.Э. Баумана»
(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 группы ИУ7-63Б: Фурдик Н. О.
(Фамилия И.О.)

Преподаватель: Рязанова Н.Ю.
(Фамилия И.О.)

Оглавление

Задание 1	2
Задание 1.2	4
Задание 2	6
Структура FILE	8
Список литературы	9

Задание 1

Проанализировать работу приведенных программ и объяснить результаты их работы.

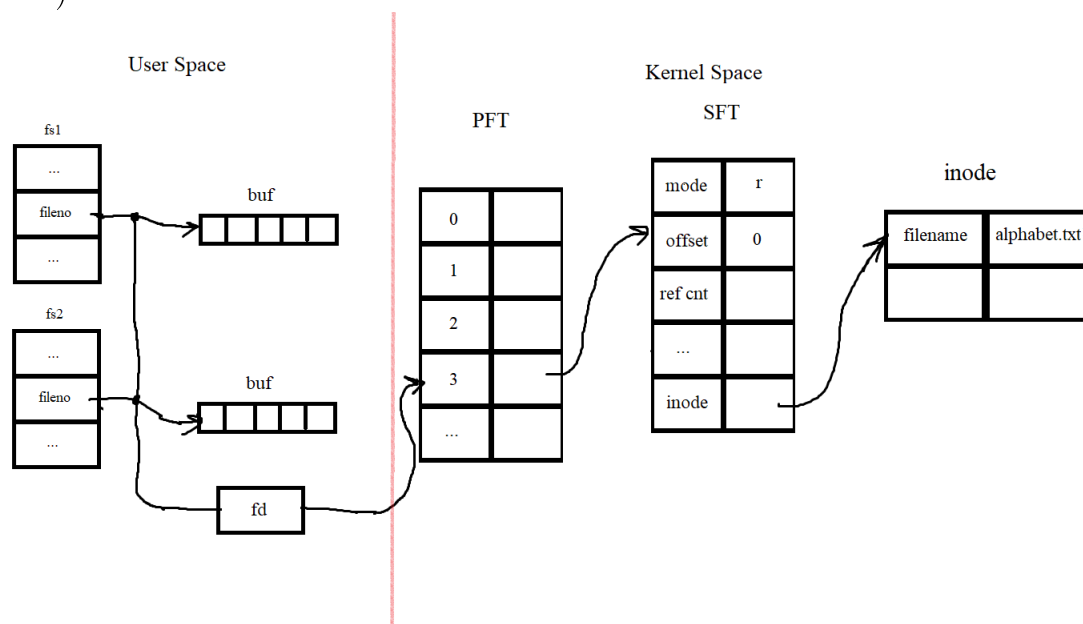
Листинг 1: Программа 1

```
1 #include <stdio.h>
2 #include <fcntl.h>
3
4 int main()
5 {
6     // have kernel open connection to file alphabet.txt
7     int fd = open("alphabet.txt", O_RDONLY);
8
9     // create two a C I/O buffered streams using the above connection
10    FILE *fs1 = fdopen(fd, "r");
11    char buff1[20];
12
13    setvbuf(fs1, buff1, _IOFBF, 20);
14
15    FILE *fs2 = fdopen(fd, "r");
16    char buff2[20];
17
18    setvbuf(fs2, buff2, _IOFBF, 20);
19
20    // read a char & write it alternatingly from fs1 and fs2
21    int flag1 = 1, flag2 = 2;
22
23    while(flag1 == 1 || flag2 == 1)
24    {
25        char c;
26
27        flag1 = fscanf(fs1, "%c", &c);
28
29        if (flag1 == 1)
30        {
31            fprintf(stdout, "%c", c);
32        }
33
34        flag2 = fscanf(fs2, "%c", &c);
35
36        if (flag2 == 1)
37        {
38            fprintf(stdout, "%c", c);
39        }
40    }
41
42    return 0;
43 }
```

Результаты работы:

```
schoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$ ./testCIO.exe
aubvcwdxyfzg
hijklmnopqrstschoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$
```

При первом вызове `fscanf()` буфер ввода `fs1` заполняется до конца, то есть первыми 20 символами, после чего меняется значение текущей позиции. Поскольку `fs1` и `fs2` ссылаются на одну и ту же запись в системной таблице открытых файлов, то при следующем вызове `fscanf()` буфер ввода `fs2` считывает последние 6 символов из файла. Результатом вывода будет являться строка, в которой символы поочередно выводятся из первого и второго буферов (поскольку в одном буфере 20 символов, а в другом 6, "вперемешку" будут выведены только первые 12 символов).

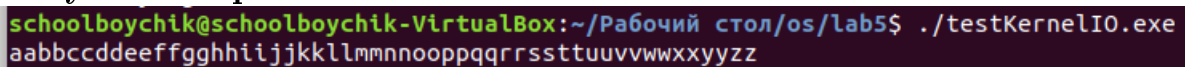


Задание 1.2

Листинг 2: Программа 2

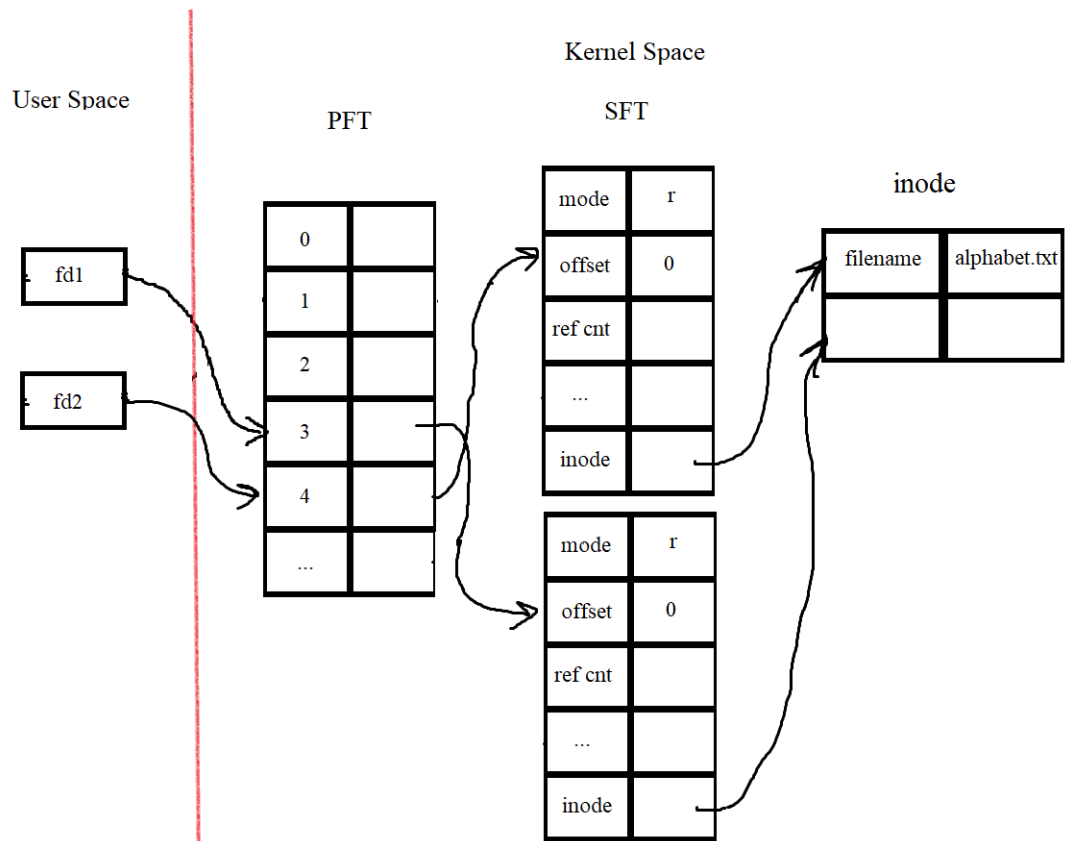
```
1 #include <fcntl.h>
2 #include <stdio.h>
3
4 int main()
5 {
6     char c;
7
8     // have kernel open two connection to file alphabet.txt
9     int fd1 = open("alphabet.txt",O_RDONLY);
10    int fd2 = open("alphabet.txt",O_RDONLY);
11    // read a char & write it alternatingly from connections fs1 & fd2
12
13    while(read(fd1,&c,1) == 1 && read(fd2,&c,1) == 1)
14    {
15        write(1,&c,1);
16
17        write(1,&c,1);
18    }
19
20    return 0;
21 }
```

Результаты работы:



```
schoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$ ./testKernelIO.exe
aabbccddeeffgghhiijjkkllmmnnooppqrrssttuuvvwwxxyyzz
```

Системный вызов `open()` вызывается 2 раза, каждый раз создавая новый файловый дескриптор (а также новую запись в системной таблице открытых файлов). Поскольку файловые дескрипторы разные, у каждого своя текущая позиция, из-за чего в результате с помощью `read()` и `write()` получается строка с дублирующимися символами.



Задание 2

Написать программу, которая открывает один и тот же файл два раза с использованием библиотечной функции `fopen()`. Для этого объявляются два файловых дескриптора. В цикле записать в файл буквы латинского алфавита поочередно передавая функции `fprintf()` то первый дескриптор, то – второй. Результат прокомментировать.

Листинг 3: Программа 3

```
1 #include <fcntl.h>
2 #include <stdio.h>
3
4 int main()
5 {
6     char alph[26] = "abcdefghijklmnopqrstuvwxyz";
7
8     FILE* f1 = fopen("p3.txt", "w");
9
10    if (f1 == NULL)
11    {
12        return -1;
13    }
14
15    FILE* f2 = fopen("p3.txt", "w");
16
17    if (f2 == NULL)
18    {
19        return -1;
20    }
21
22    for (int i = 0; i < 26; i++)
23    {
24        if (i % 2 == 0)
25            fprintf(f1, "%c", alph[i]);
26
27        if (i % 2 == 1)
28            fprintf(f2, "%c", alph[i]);
29    }
30
31    fclose(f1);
32    fclose(f2);
33
34    return 0;
35 }
```

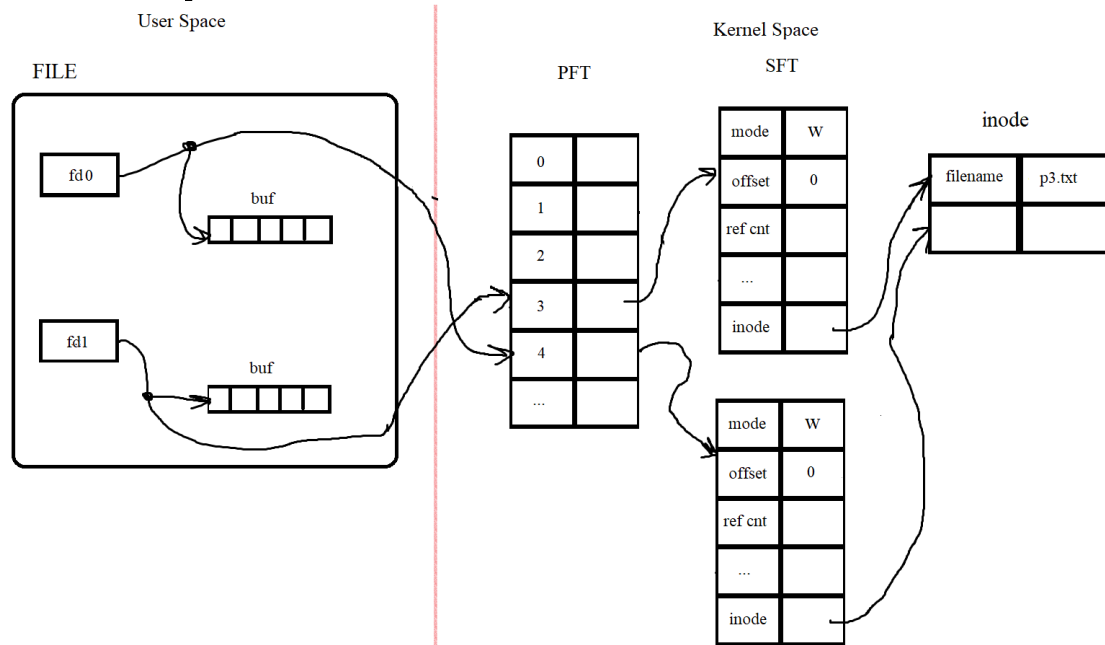
Результаты работы:

```

schoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$ ./p3.exe
schoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$ cat p3.txt
bdfhjlnprtvmxzs
schoolboychik@schoolboychik-VirtualBox:~/Рабочий стол/os/lab5$

```

`fprintf()` обеспечивает буферизованный вывод, а значит запись файл происходит только при вызове функций `fclose()`, `fflush()` или заполнении буфера ввода. Функция `fclose(f1)` закрывает файловый дескриптор и очистит поток, на который указывает `f1`. После ее выполнения в файл будут записаны данные из первого потока. Затем `fclose(f2)` проделает то же самое с `f2`, но так как оба потока открыты на запись, данные, записанные с помощью первого потока перезапишутся данными из второго потока.



Структура FILE

Ниже приведена структура FILE.

Листинг 4: Структура FILE

```
1 struct _IO_FILE {
2     int _flags;          /* High-order word is _IO_MAGIC; rest is flags. */
3     #define _IO_file_flags _flags
4
5     /* The following pointers correspond to the C++ streambuf protocol. */
6     /* Note: Tk uses the _IO_read_ptr and _IO_read_end fields directly. */
7     char* _IO_read_ptr;  /* Current read pointer */
8     char* _IO_read_end;  /* End of get area. */
9     char* _IO_read_base; /* Start of putback+get area. */
10    char* _IO_write_base; /* Start of put area. */
11    char* _IO_write_ptr;  /* Current put pointer. */
12    char* _IO_write_end;  /* End of put area. */
13    char* _IO_buf_base;   /* Start of reserve area. */
14    char* _IO_buf_end;    /* End of reserve area. */
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    #if 0
26    int _blksize;
27    #else
28    int _flags2;
29    #endif
30    _IO_off_t _old_offset; /* This used to be _offset but it's too small. */
31
32    #define __HAVE_COLUMN /* temporary */
33    /* 1+column number of pbase(); 0 is unknown. */
34    unsigned short _cur_column;
35    signed char _vtable_offset;
36    char _shortbuf[1];
37
38    /* char* _save_gptr; char* _save_egptr; */
39
40    _IO_lock_t *_lock;
41    #ifndef _IO_USE_OLD_IO_FILE
42    };
43
44    struct _IO_FILE_complete {
45        struct _IO_FILE _file;
46    #endif
47    #if defined _G_IO_IO_FILE_VERSION && _G_IO_IO_FILE_VERSION == 0x20001
48        _IO_off64_t _offset;
```

```

49 # if defined _LIBC || defined _GLIBCXX_USE_WCHAR_T
50 /* Wide character stream stuff. */
51 struct _IO_codecvt *_codecvt;
52 struct _IO_wide_data *_wide_data;
53 struct _IO_FILE *_freeres_list;
54 void *_freeres_buf;
55 size_t _freeres_size;
56 # else
57 void *__pad1;
58 void *__pad2;
59 void *__pad3;
60 void *__pad4;
61 size_t __pad5;
62 # endif
63 int _mode;
64 /* Make sure we don't get into trouble again. */
65 char __unused2[15 * sizeof (int) - 4 * sizeof (void *) - sizeof (size_t)];
66 #endif
67 };
68 #ifndef __cplusplus
69 typedef struct _IO_FILE _IO_FILE;
70 #endif
71 typedef struct _IO_FILE FILE;

```

Литература

1. Рязанова Н.Ю. - Курс лекций по "Операционным системам"[Текст], Москва 2020 год.
2. У. Ричард Стивенс, Стивен А. Раго - UNIX. Профессиональное программирование. 3-е изд. - Москва: Питер, 2018. - 944 с.