



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»*

**О Т Ч Е Т**

**по лабораторной работе № 0 5**

**Дисциплина: *Операционные системы***

Студент

**ИУ7И-66Б**

(Группа)

**Нгуен Ф. С.**

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

**Рязанова Н. Ю.**

(Подпись, дата)

(И.О. Фамилия)

*Москва, 2021*

## Структура `_IO_FILE` (/usr/include/x86\_64-linux-gnu/bits/types/struct\_FILE.h)

```
struct _IO_FILE
{
    int _flags;          /* High-order word is _IO_MAGIC; rest is flags. */

    /* The following pointers correspond to the C++ streambuf protocol. */
    char *_IO_read_ptr;  /* Current read pointer */
    char *_IO_read_end;  /* End of get area. */
    char *_IO_read_base; /* Start of putback+get area. */
    char *_IO_write_base; /* Start of put area. */
    char *_IO_write_ptr; /* Current put pointer. */
    char *_IO_write_end; /* End of put area. */
    char *_IO_buf_base; /* Start of reserve area. */
    char *_IO_buf_end; /* End of reserve area. */

    /* The following fields are used to support backing up and undo. */
    char *_IO_save_base; /* Pointer to start of non-current get area. */
    char *_IO_backup_base; /* Pointer to first valid character of backup area */
    char *_IO_save_end; /* Pointer to end of non-current get area. */

    struct _IO_marker *_markers;

    struct _IO_FILE *_chain;

    int _fileno;
    int _flags2;
    __off_t _old_offset; /* This used to be _offset but it's too small. */

    /* 1+column number of pbase(); 0 is unknown. */
    unsigned short _cur_column;
    signed char _vtable_offset;
    char _shortbuf[1];

    _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

## I. Первая программа

```
#include <stdio.h>

#include <fcntl.h>

#define OK 0
#define BUF_SIZE 20

#define FILE_NAME "./atoz.txt"

int main(void)
{
    int fd = open(FILE_NAME, O_RDONLY);

    FILE *fs1 = fdopen(fd, "r");
    char buff1[BUF_SIZE];
    setvbuf(fs1, buff1, _IOFBF, BUF_SIZE);

    FILE *fs2 = fdopen(fd, "r");
    char buff2[BUF_SIZE];
    setvbuf(fs2, buff2, _IOFBF, BUF_SIZE);

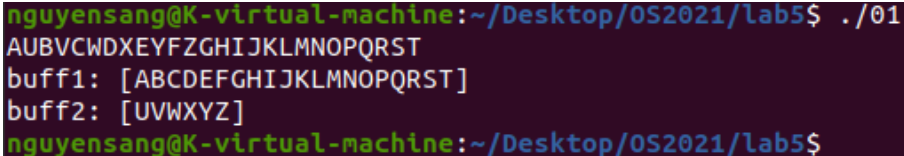
    int flag1 = 1, flag2 = 2;
    while (flag1 == 1 || flag2 == 1)
    {
        char c;

        if ((flag1 = fscanf(fs1, "%c", &c)) == 1)
        {
            fprintf(stdout, "%c", c);
        }

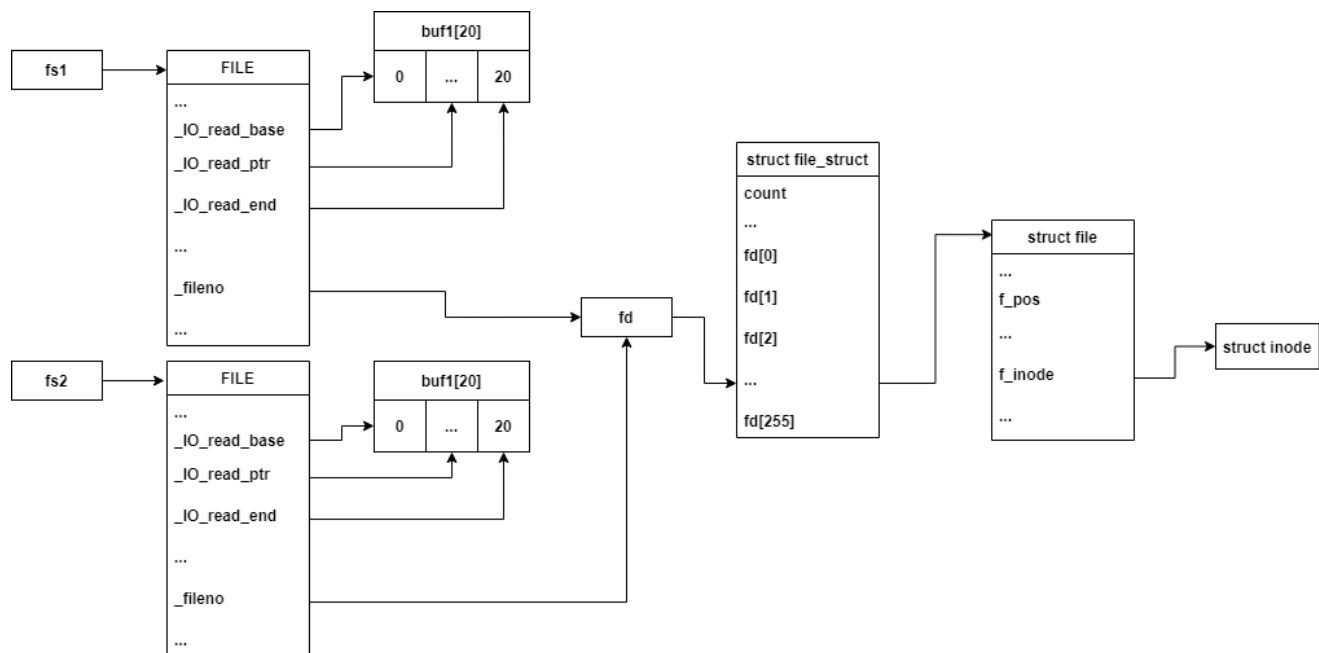
        if ((flag2 = fscanf(fs2, "%c", &c)) == 1)
        {
            fprintf(stdout, "%c", c);
        }
    }

    fprintf(stdout, "\nbuff1: [%s]\nbuff2: [%s]\n", buff1, buff2);

    return OK;
}
```

A terminal window with a dark purple background. The prompt is 'nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5\$'. The command './01' has been executed. The output is 'AUBVCWDXEYFZGHIJKLMNOPQRST' on the first line, 'buff1: [ABCDEFGHIJKLMNOPQRSTUVWXYZ]' on the second line, and 'buff2: [UVWXYZ]' on the third line. The prompt is repeated at the bottom.

```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$ ./01
AUBVCWDXEYFZGHIJKLMNOPQRST
buff1: [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
buff2: [UVWXYZ]
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$
```



## Анализ работы

1. Функция **open()** создает новый файловый дескриптор файла "atoz.txt", запись в системной таблице открыт файлов. Эта запись регистрирует смещение в файле и флаги состояния файла;
2. функция **fdopen()** создает указатели на структуру **FILE**. Поле `_fileno` содержит дескриптор, который вернула функция **open()**;
3. функция **setvbuf()** задает размер буфера в 20 байт и меняет тип буферизации на полную;
4. при первом вызове функции **fscanf()** в цикле (для **fs1**), **buff1** будет заполнен полностью -- первыми 20 буквами алфавита. **f\_pos** в структуре **struct\_file** открытого файла увеличится на 20;
5. при втором вызове **fscanf()** в цикле (для **fs2**) буфер **buff2** будет заполнен оставшимися 6 символами (начиная с **f\_pos**);
6. в цикле выводятся символы из **buff1** и **buff2**.

## II. Вторая программа

```
#include <fcntl.h>
#include <unistd.h>

#define OK 0
#define FILE_NAME "./atoz.txt"

int main(void)
{
    int fd1 = open(FILE_NAME, O_RDONLY);
    int fd2 = open(FILE_NAME, O_RDONLY);
    int rc1, rc2 = 1;

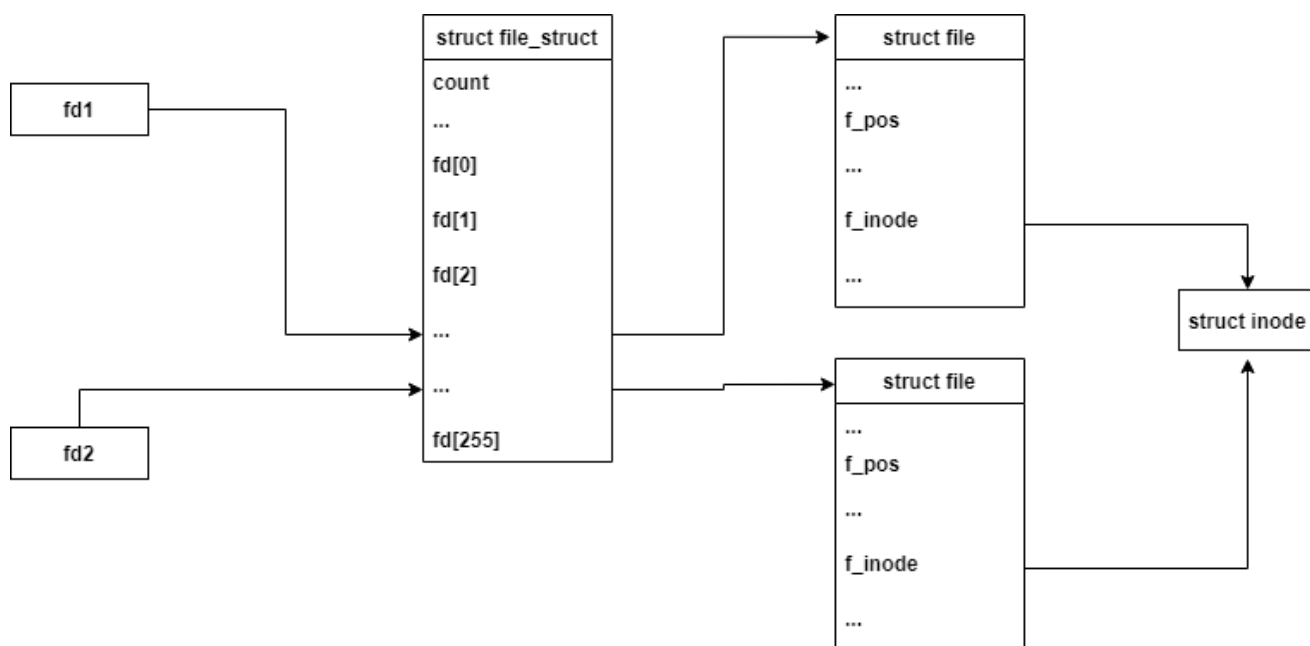
    while (rc1 == 1 || rc2 == 1)
    {
        char c;

        rc1 = read(fd1, &c, 1);
        if (rc1 == 1)
        {
            write(1, &c, 1);
        }

        rc2 = read(fd2, &c, 1);
        if (rc2 == 1)
        {
            write(1, &c, 1);
        }
    }

    return OK;
}
```

```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$ ./02
AABBCCDDEEFFGGHHIIJJKLLMMNNOOPPQQRRSSTTUUVVWXXYYZZnguyensang@K-virtual-machine
:~/Desktop/OS2021/lab5$
```



## Анализ работы

1. Функция **open()** создает файловые дескрипторы, два раза для одного и того же файла, поэтому в программе существует две различные **struct file**, но ссылающиеся на один и тот же **struct inode**;
2. Из-за того что структуры разные, посимвольная печать просто дважды выведет содержимое файла в формате 'aabbcc...';

### III. Третья программа

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

#define OK 0
#define FILE_NAME "./out.txt"

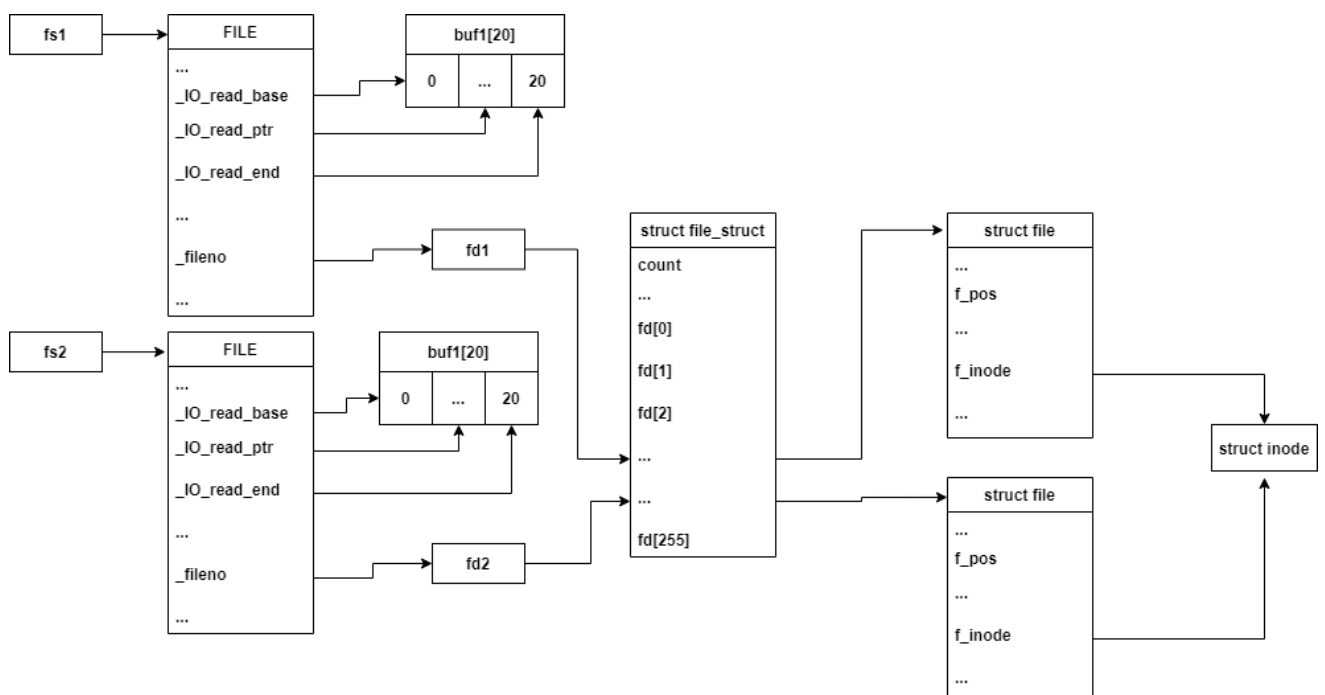
int main()
{
    FILE *f1 = fopen(FILE_NAME, "w");
    FILE *f2 = fopen(FILE_NAME, "w");

    for (char c = 'a'; c <= 'z'; c++)
    {
        if (c % 2)
        {
            fprintf(f1, "%c", c);
        }
        else
        {
            fprintf(f2, "%c", c);
        }
    }

    fclose(f2);
    fclose(f1);

    return OK;
}
```

```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$ gcc prog_03.c -o 03
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$ ./03
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$ cat out.txt
acegikmoqsuwynguyensang@K-virtual-machine:~/Desktop/OS2021/lab5$
```



## **Анализ работы**

1. Файл открывается на запись два раза, с помощью функции `fopen()`. Для этого объявляются два файловых дескриптора.;
2. По умолчанию используется полная буферизация, при которой запись в файл из буфера произойдет либо при заполнении буфера, либо при вызове `fclose()`, либо при завершении процесса;
3. В цикле записываются в файл буквы латинского алфавита поочередно передавая функции `fprintf()` то первый дескриптор, то – второй;
4. Из-за того `f_pos` независимы для каждого дескриптора файла, запись в файл будет производится с самого начала;
5. Таким образом, информация записанная при первом вызове `fclose()` будет потеряна в результате второго вызова `fclose()`;