



**«Московский государственный технический  
университет  
имени Н.Э. Баумана (национальный исследовательский  
институт)»  
(МГТУ им. Н.Э. Баумана)**

---

**О т ч ё т**

**п о л а б о р а т о р н о й   р а б о т е   5**

**Дисциплина:** Операционные системы

**Тема:** Буферизованный и не буферизованный ввод-вывод.

Студентка гр. ИУ7и-66Б

\_\_\_\_\_  
(Подпись, дата)

**Нгуен Т.Т.**  
(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

**Рязанова Н.Ю.**  
(И.О. Фамилия)

Москва, 2021г

## Содержание

<b>I. Задание.....</b>	<b>3</b>
<b>II. Программа:.....</b>	<b>3</b>
1. Первая программа: .....	3
Код: .....	3
Результат работы программы: .....	4
Объяснить результаты их работы: .....	4
2. Вторая программа:.....	5
Код: .....	5
Результат работы программы: .....	5
Объяснить результаты их работы: .....	6
3. Третья программа:.....	6
Код: .....	6
Результат работы программы: .....	7
Объяснить результаты их работы: .....	7

## Задача: анализ особенностей работы функций ввода-вывода в UNIX/Linux

### I. Задание

В лабораторной работе анализируется результат выполнения трех программ. Программы демонстрируют открытие одного и того же файла несколько раз. Реализация открытия файла в одной программе несколько раз выбрана для простоты. Такая ситуация возможна в системе, когда один и тот же файл несколько раз открывают разные процессы. Но для получения ситуаций аналогичных тем, которые демонстрируют приведенные программы надо было бы синхронизировать работу процессов. При выполнении асинхронных процессов такая ситуация вероятна и ее надо учитывать, чтобы избежать потери данных или получения неверного результата при выводе в файл.

Проанализировать работу приведенных программ и объяснить результаты их работы.

### II. Программа:

#### 1. Первая программа:

Код:

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

/*
On my machine, a buffer size of 20 bytes
translated into a 12-character buffer.
Apparently 8 bytes were used up by the
stdio library for bookkeeping.
*/

int main()
{
    // have kernel open connection to file alphabet.txt
    int fd = open("alphabet.txt", O_RDONLY);

    // create two C I/O buffered streams using the above connection
    FILE *fs1 = fdopen(fd, "r");
    char buff1[20];
    setvbuf(fs1, buff1, _IOFBF, 20);

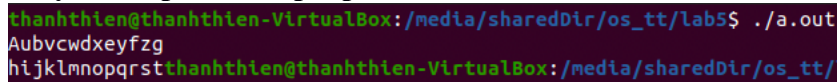
    FILE *fs2 = fdopen(fd, "r");
    char buff2[20];
    setvbuf(fs2, buff2, _IOFBF, 20);
}
```

```

// read a char & write it alternately from fs1 and fs2
int flag1 = 1, flag2 = 2;
while(flag1 == 1 || flag2 == 1)
{
    char c;
    flag1 = fscanf(fs1, "%c",&c);
    if (flag1 == 1) {
        fprintf(stdout,"%c",c);
    }
    flag2 = fscanf(fs2, "%c",&c);
    if (flag2 == 1) {
        fprintf(stdout,"%c",c);
    }
}
return 0;
}

```

### Результат работы программы:



```

thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$ ./a.out
Aubvcwdxeyfzg
hijklmnopqrstthanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$ █

```

### Объяснить результаты их работы:

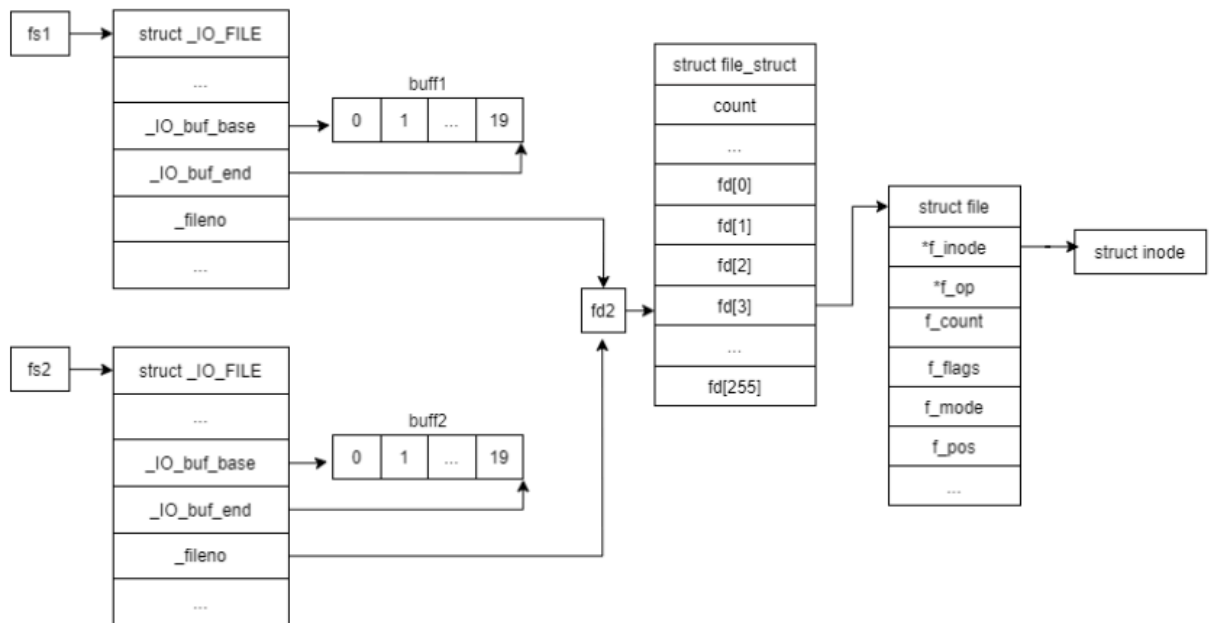
Системный вызов `open()` создает новый файловый дескриптор для открытого файла только для чтение, возвращая индекс в массиве файловых объектов `fd_array` структуры `files_struct`.

Вызов `fdopen()` создает структуры типа `FILE` (`fs1` и `fs2`), которые ссылаются на дескриптор открытого файла, созданный ранее.

Для дескрипторов `fs1` и `fs2` помощью `setbuf` задаем соответствующие буферы и тип буферизации `_IOFBF` (Fully buffered).

В цикле сначала при первом вызове `fscanf(fs1,...)`, в `buff1` заполнен 20 первыми символами, указатель `f_pos` установится на следующий за последним записанным в буфер символ (20), потом при первом вызове `fscanf(fs2,...)`, в `buff2` считываются оставшиеся в файле символы.

Зная содержимое буферов, `f_pos` указывает на конец файла, очевидно, что можно получить результат, как показано на рисунке.



## 2. Вторая программа:

Код:

```

#include <fcntl.h>
#include <unistd.h>
int main()
{
    char c;
    // have kernel open two connection to file alphabet.txt
    int fd1 = open("alphabet.txt", O_RDONLY);
    int fd2 = open("alphabet.txt", O_RDONLY);
    // read a char & write it alternately from connections fs1 & fd2
    int flag = 1;
    while(flag)
    {
        if (read(fd1, &c, 1) == 1) {
            write(1, &c, 1);
            if (read(fd2, &c, 1) == 1) {
                write(1, &c, 1);
            } else {
                flag = 0;
            }
        } else {
            flag = 0;
        }
    }
    return 0;
}

```

Результат работы программы:

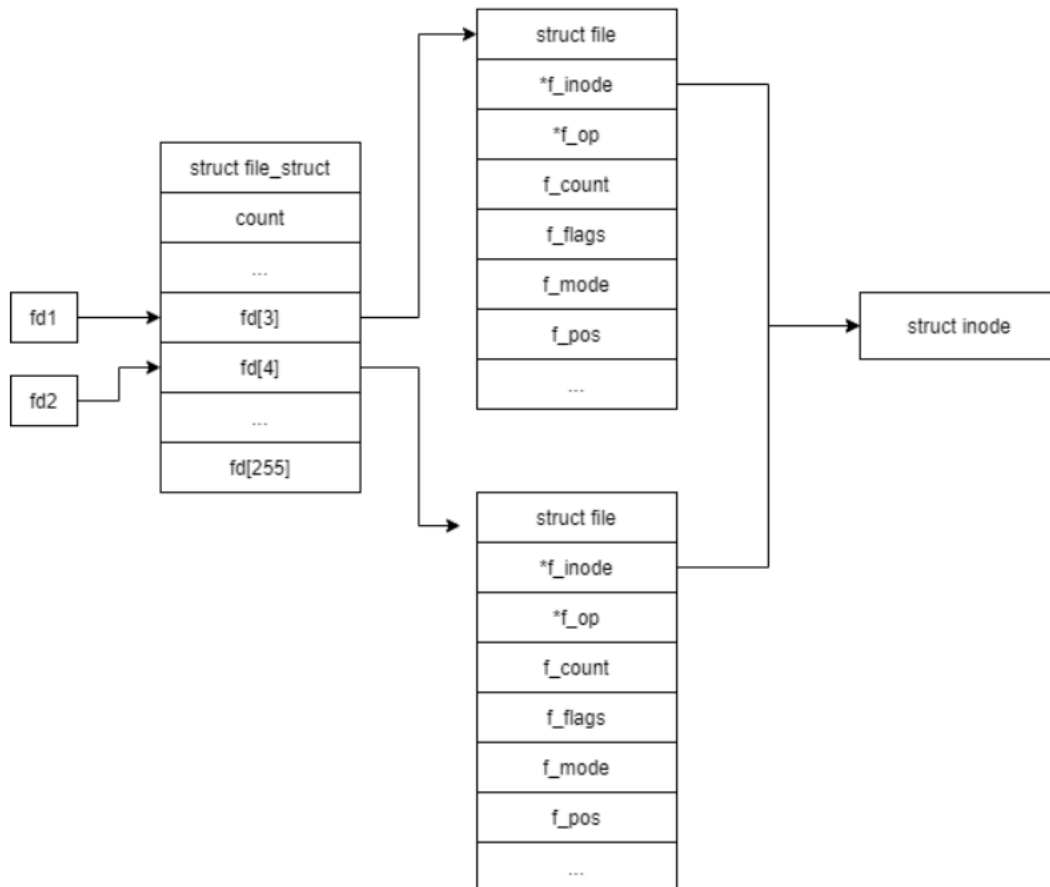
```

thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$ ./2.exe
AAbbccddeeffgghhiijjkkllmmnnoppqqrrssttuuvvwwxxyyzz
thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$

```

### Объяснить результаты их работы:

С помощью системного вызова `open()` создается два дескриптора открытого на чтение файла, в системной таблице открытых файлов создаются две новых записи. Так как разные `f_pos` дескриптора, то чтение будет проходить независимо. На экран дважды напечатаются все символы из `alphabet.txt`.



### 3. Третья программа:

Код:

```

#include <stdio.h>
#include <sys/stat.h>
#include <pthread.h>

#define FMT_STR " FS%d: inode = %ld, size = %ld\n"

void *thr_fn(int data)
{
    int fid = (int)data;
    struct stat statbuf;

    FILE *fs = fopen("out.txt", "w");
    stat("out.txt", &statbuf);

```

```

printf("FOPEN " FMT_STR, fid, statbuf.st_ino, statbuf.st_size);

for (char c = 'a'; c <= 'z'; c++)
{
    if ((c % 2) == (fid == 1))
        fprintf(fs, "%c", c);
}

fclose(fs);
stat("out.txt", &statbuf);
printf("FCLOSE" FMT_STR, fid, statbuf.st_ino, statbuf.st_size);
}

int main()
{
    pthread_t tid[2];
    int fid[2] = {0, 1};

    for (int i = 0; i < 2; i++)
    {
        if (pthread_create(&tid[i], NULL, thr_fn, fid[i]))
        {
            printf("Error: can't create thread\n");
            return -1;
        }
    }

    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);

    return 0;
}

```

### Результат работы программы:

```

thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$ ./3.exe
FOPEN FS1: inode = 74, size = 0
FOPEN FS0: inode = 74, size = 0
FCLOSE FS1: inode = 74, size = 0
FCLOSE FS0: inode = 74, size = 13
thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$ cat out.txt
bdfhjlnprtvxz
thanhthien@thanhthien-VirtualBox:/media/sharedDir/os_tt/lab5$

```

### Объяснить результаты их работы:

Файл out.txt открывается функцией fopen() на запись дважды, в два различных потока. Для этого объявляются два файловых дескриптора. По умолчанию используется полная буферизация, при которой запись в файл из буфера произойдет либо при полностью заполнении буфера, либо при вызове fclose(), либо при вызове fflush().

В цикле записываются в файл буквы латинского алфавита поочередно передавая функции fprintf() то первый дескриптор, то – второй.

Предположим, сначала вызвать fclose для fs1. При вызове fclose() для fs1 buff1 записывается в файл. При вызове fclose() для fs2, все содержимое файла очищается

(перезаписывать с `f_pos=0`), а в файл записывается содержимое `buff2`. Произошла утеря данных, в файле окажется только содержимое `buff2`.

