



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»

Отчёт

по лабораторной работе № 5

Название: Буферизованный и не буферизованный ввод-вывод

Дисциплина: Операционные системы

Студент

ИУ7-65Б

(Группа)

Д.В. Сусликов

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Н.Ю. Рязанова

(Подпись, дата)

(И.О. Фамилия)

Москва, 2021

Программа 1

Листинг:

```
1  #include <stdio.h>
2  #include <fcntl.h>
3
4  int main()
5  {
6      int fd = open("alph.txt", O_RDONLY);
7
8      FILE *fs1 = fdopen(fd, "r");
9      char buff1[20];
10     setvbuf(fs1, buff1, _IOFBF, 20);
11
12     FILE *fs2 = fdopen(fd, "r");
13     char buff2[20];
14     setvbuf(fs2, buff2, _IOFBF, 20);
15
16     int flag1 = 1, flag2 = 2;
17     while(flag1 == 1 || flag2 == 1)
18     {
19         char c;
20         flag1 = fscanf(fs1, "%c", &c);
21         if (flag1 == 1)
22             fprintf(stdout, "%c", c);
23
24         flag2 = fscanf(fs2, "%c", &c);
25         if (flag2 == 1)
26             fprintf(stdout, "%c", c);
27     }
28
29     return 0;
30 }
```

```
danil@danil-virtual-machine:~/Рабочий стол/os/lr5$ ./1.out  
AUBVCWDXEYFZGHIJKLMNOPQRSTdanil@danil-virtual-machine:~/Рабочий стол/os/lr5$
```

Рисунок 1 – Результат программы 1

Системный вызов `open()` создает дескриптор открытого на чтение файла. `Open()` возвращает индекс в массиве `fd` структуры `files_struct`. `fdopen()` создает структуры типа `FILE` (`fs1` и `fs2`), которые ссылаются на дескриптор, созданный системным вызовом `open`.

Создаём буферы `buff1` и `buff2` размером 20 байт. С помощью `setbuf` задаём соответствующие буферы для дескрипторов `fs1` и `fs2` и устанавливаем тип буферизации `_IOFBF` (полная буферизация).

Далее в цикле `fscanf()` поочерёдно для `fs1` и `fs2`. Так как установлена полная буферизация, то при первом вызове `fscanf()` буфер будет заполнен полностью, либо вплоть до конца файла, а `f_pos` установится на следующий за последним записанным в буфер символ.

При первом вызове `fscanf(fs1,"%c &c)` в буфер `buff1` считываются первые 20 символов (`abcdefghijklmnopqrst`), запись производится через переменную `s`, а затем выводится с помощью `fprintf`, символ `'a'`.

При первом вызове `fscanf(fs2,"%c &c)`, в буфер `buff2` считываются оставшиеся в файле символы – `uvwxyz` (в `s` записывается символ `'u'`).

Внутри цикла будут поочередно выводиться символы из двух буферов до тех пор, пока символы в одном из буферов не закончатся. Тогда на экран будут последовательно выведены оставшиеся символы из другого буфера.

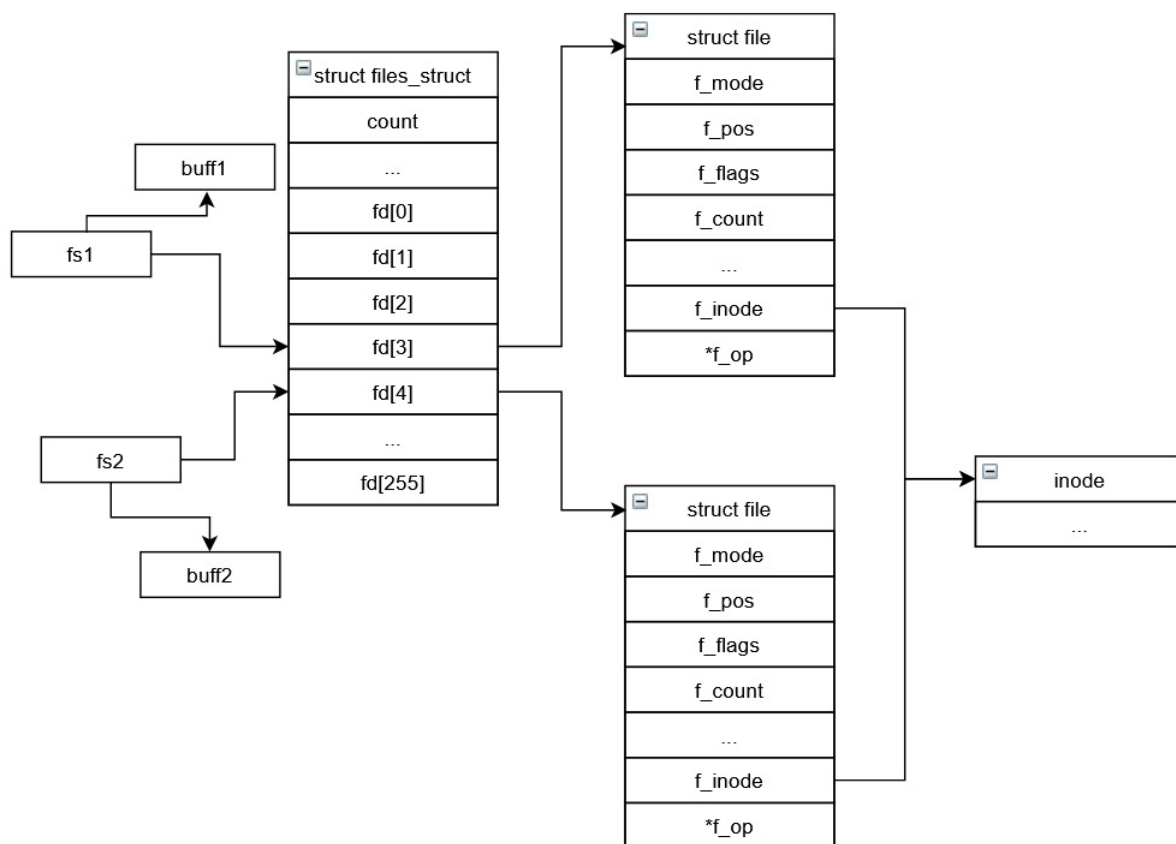
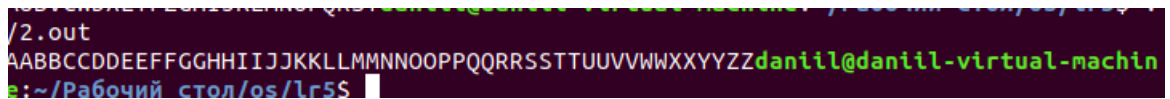


Рисунок 2 – Схема связей структур

Листинг:

```
1 #include <fcntl.h>
2
3 int main()
4 {
5     char c;
6     int fd1 = open("alph.txt", O_RDONLY);
7     int fd2 = open("alph.txt", O_RDONLY);
8
9     while(read(fd1, &c, 1) == 1 && read(fd2, &c, 1) == 1)
10    {
11        write(1, &c, 1);
12        write(1, &c, 1);
13    }
14    return 0;
15 }
```



```
/2.out
AABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRRSSTTUUVVWWXXYYZZdanil@danil-virtual-machin
~ /Рабочий стол/os/1r55
```

Рисунок 3 – Результат программы 2

Open() создаст два дескриптора одного файла, доступного только для чтения. Будут созданы две структуры struct file в общесистемной таблице открытых файлов. Таким образом, поля f_pos в каждой из структур независимы.

В цикле выполняются read(), считывающий символ и write(), записывающий символ в стандартный поток. Указатель f_pos изменяется независимо от другого дескриптора, поэтому каждый символ будет выведен дважды.

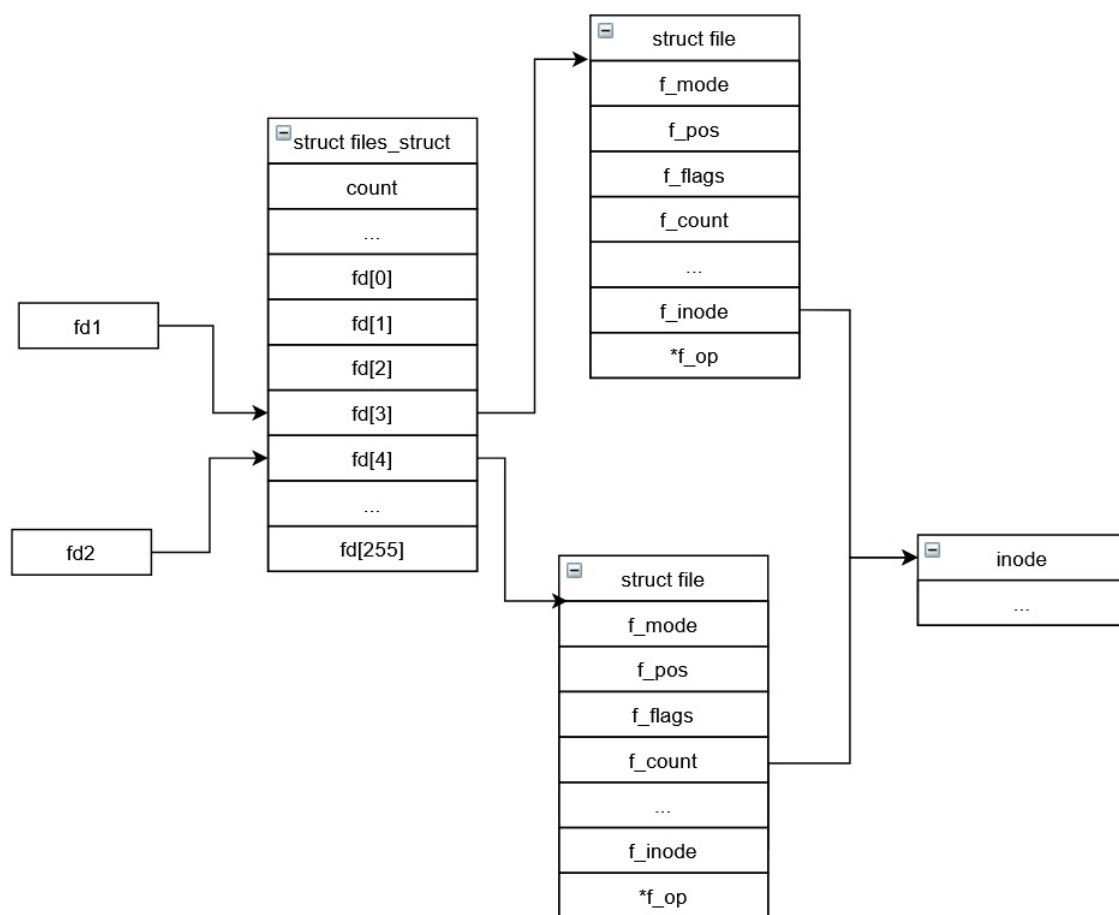
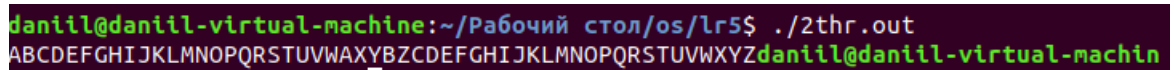


Рисунок 4 – Схема связей структур

Листинг:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <fcntl.h>
5 void *reading(int *fd)
6 {
7     char c;
8     while (read(*fd, &c, 1) == 1)
9         write(1, &c, 1);
10 }
11 int main()
12 {
13     int fd1 = open("alph.txt", O_RDONLY);
14     int fd2 = open("alph.txt", O_RDONLY);
15     pthread_t thread1;
16
17     int stat1 = pthread_create(&thread1, NULL, reading, &fd1);
18     if (stat1 != 0)
19     {
20         printf("Error. Can't create thread 1\n");
21         return -1;
22     }
23
24     reading(&fd2);
25
26     pthread_join(thread1, NULL);
27
28     return 0;
29 }
```



```
daniil@daniil-virtual-machine:~/Рабочий стол/os/lr5$ ./2thr.out
ABCDEFGHIJKLMNOPQRSTUVWXYZdaniil@daniil-virtual-machin
```

Рисунок 5 – Результат программы 2 с потоками

Аналогично программе без потока, вызов `open()` создаст два независимых дескриптора открытых файлов, содержащих собственные значения `f_pos`. В результате каждый поток выполнит чтение и запись независимо, что приведет к повтору каждого символа в результирующей строке.

Листинг:

```
1  #include <stdio.h>
2  #include <sys/stat.h>
3
4  int main()
5  {
6      struct stat stats;
7
8      FILE *fp1 = fopen("alphChange.txt", "w");
9      stat("alphChange.txt", &stats);
10     printf("1:  inode — %d  size — %d\n", (int) stats.st_ino, (
        int) stats.st_size);
11
12     FILE *fp2 = fopen("alphChange.txt", "w");
13     stat("alphChange.txt", &stats);
14     printf("2:  inode — %d  size — %d\n", (int) stats.st_ino, (
        int) stats.st_size);
15
16     for (char c = 'a'; c <= 'z'; c++)
17     {
18         if (c % 2 == 0)
19             fprintf(fp1, "%c", c);
20         else
21             fprintf(fp2, "%c", c);
22     }
23     fclose(fp1);
24     stat("alphChange.txt", &stats);
25     printf("1:  inode — %d  size — %d\n", (int) stats.st_ino, (
        int) stats.st_size);
26
27     fclose(fp2);
28     stat("alphChange.txt", &stats);
29     printf("2:  inode — %d  size — %d\n", (int) stats.st_ino, (
        int) stats.st_size);
```

```

danil@danil-virtual-machine:~/Рабочий стол/os/lr5$ ./3.out
1: inode - 262547 size - 0
2: inode - 262547 size - 0
1: inode - 262547 size - 13
2: inode - 262547 size - 13
danil@danil-virtual-machine:~/Рабочий стол/os/lr5$

```

Рисунок 6 – Результат программы 3

Функция `fopen()` возвращает указатель на открытый файл. Режим 'w' указывает на открытие файла только для записи; система помещает указатель в начало файла. Если файл не существует - пробует его создать. Создаются дескрипторы открытых файлов, следовательно, два независимых `f_pos`.

Функция `fprintf()` – функция буферизованного ввода/вывода. В цикле происходит запись каждого четного символа в один буфер, и нечетного в другой. Запись в файл из буфера происходит при вызове функции `fclose()`. Поскольку сначала вызывается `fclose(fp1)`, то буфер, ассоциированный с `fp1` будет записан в файл.

При следующем вызове `fclose(fp2)` содержимое файла будет перезаписано, в результате чего в нем окажутся лишь символы из буфера, связанного с `fp2`. Таким образом, данные из первого буфера будут утеряны.

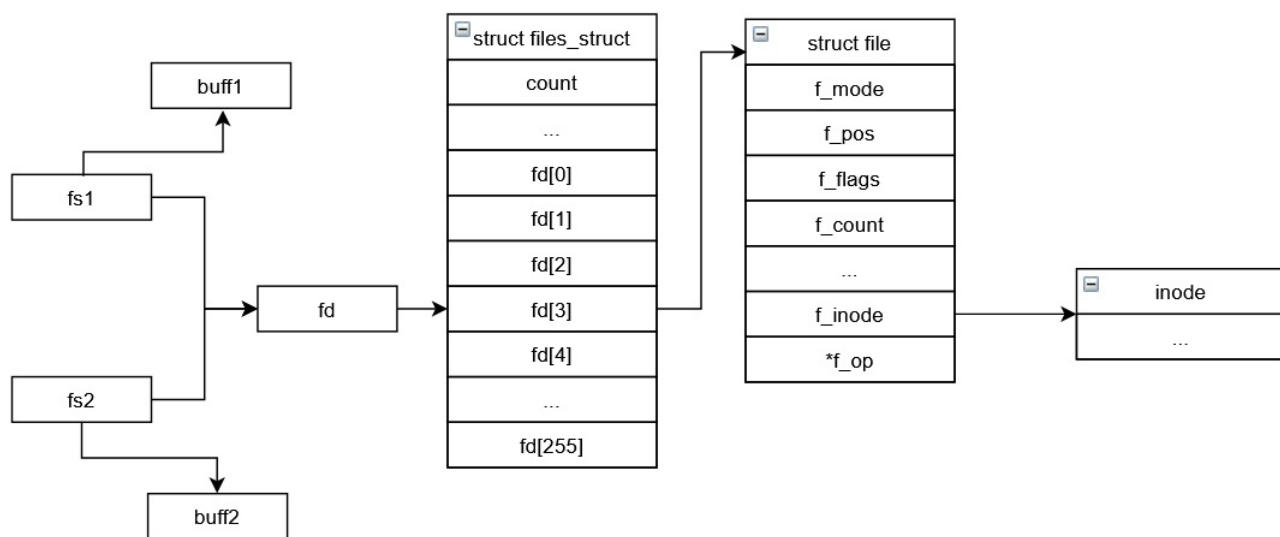
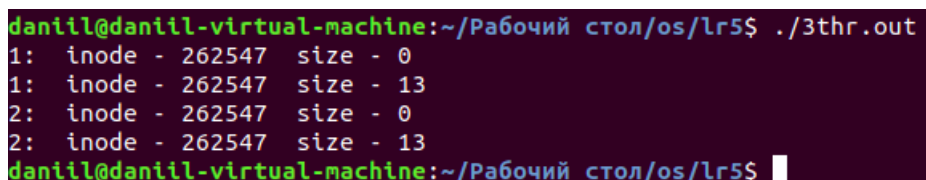


Рисунок 7 – Схема связей структур

Листинг:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/stat.h>
4 #include <pthread.h>
5
6 void *writing(int *fd)
7 {
8     struct stat sb;
9
10    FILE *fp = fopen("alphChange.txt", "w");
11    stat("alphChange.txt", &sb);
12    printf("%d:  inode — %d  size — %d\n", *fd, (int) sb.st_ino, (
13        int) sb.st_size);
14    for (char c = 'a'; c <= 'z'; c++)
15    {
16        if (c % 2 == 0 && *fd % 2 == 1)
17            fprintf(fp, "%c", c);
18
19        if (c % 2 == 1 && *fd % 2 == 0)
20            fprintf(fp, "%c", c);
21    }
22    fclose(fp);
23    stat("alphChange.txt", &sb);
24    printf("%d:  inode — %d  size — %d\n", *fd, (int) sb.st_ino, (
25        int) sb.st_size);
26
27    }
28
29 int main()
30 {
31     pthread_t thread1, thread2;
32     int fd1 = 1, fd2 = 2;
33
34     int stat1 = pthread_create(&thread1, NULL, writing, &fd1);
```

```
33     if (stat1 != 0)
34     {
35         printf("Error. Can't create thread 1\n");
36         return -1;
37     }
38     writing(&fd2);
39
40     pthread_join(thread1, NULL);
41
42     return 0;
43 }
```



```
danil@danil-virtual-machine:~/Рабочий стол/os/lr5$ ./3thr.out
1: inode - 262547 size - 0
1: inode - 262547 size - 13
2: inode - 262547 size - 0
2: inode - 262547 size - 13
danil@danil-virtual-machine:~/Рабочий стол/os/lr5$
```

Рисунок 8 – Результат программы 3 с потоками

Аналогично программе без потоков, будут созданы два независимых дескриптора. Каждый из потоков будет выполнять запись символов в соответствующий буфер. И в результате вызова `fclose()` дважды, в файле сохранить содержимое того буфера, чей поток завершился последним.

```
struct stat {  
    dev_t st_dev; /* устройство */  
    ino_t st_ino; /* inode */  
    mode_t st_mode; /* режим доступа */  
    nlink_t st_nlink; /* количество жестких ссылок */  
    uid_t st_uid; /* идентификатор пользователя-владельца */  
    gid_t st_gid; /* идентификатор группы-владельца */  
    dev_t st_rdev; /* тип устройства */  
    /* (если это устройство) */  
    off_t st_size; /* общий размер в байтах */  
    blksize_t st_blksize; /* размер блока ввода-вывода */  
    /* в файловой системе */  
    blkcnt_t st_blocks; /* количество выделенных блоков */  
    time_t st_atime; /* время последнего доступа */  
    time_t st_mtime; /* время последней модификации */  
    time_t st_ctime; /* время последнего изменения */  
};
```

```

        typedef struct __sFILE {
...
    unsigned char *_p;
    short _flags;
    short _file;
    struct __sbuf _bf;
...
    void *_cookie;
...
    struct __sbuf _ub;
    struct __sFILEX *_extra;
    int _ur;
...
    unsigned char _ubuf[3];
    unsigned char _nbuf[1];
...
    struct __sbuf _lb;
    int _blksize;
    fpos_t _offset;
} FILE;

```