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, искусственный интелект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе № 10 по курсу «Операционные системы»

Тема Буферизованный и не буферизованный ввод–вывод

Студент Волков Г. В.

Группа ИУ7-61Б

Оценка (баллы)

Преподаватель Рязанова Н. Ю.

Москва — 2023 г.

Структура FILE

```
1    typedef struct _IO_FILE FILE;
2    struct _IO_FILE
3    {
4        int _flags;     /* High-order word is _IO_MAGIC; rest is flags.
           */
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       /* The following fields are used to support backing up and
           undo. */
15       char *_IO_save_base; /* Pointer to start of non-current get
           area. */
16       char *_IO_backup_base;  /* Pointer to first valid character of
           backup area */
17       char *_IO_save_end; /* Pointer to end of non-current get area.
           */
18       struct _IO_marker *_markers;
19       struct _IO_FILE *_chain;
20       int _fileno;
21       int _flags2;
22       __off_t _old_offset; /* This used to be _offset but it's too
           small.  */
23       /* 1+column number of pbase(); 0 is unknown. */
24       unsigned short _cur_column;
25       signed char _vtable_offset;
26       char _shortbuf[1];
27       _IO_lock_t *_lock;
28       #ifdef _IO_USE_OLD_IO_FILE
29   };
```

Программа №1

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

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

    // create two a 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 alternatingly 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:
aubvcwdxeyfzghijklmnopqrst

Программа с дополнительным потоком:

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

void *thread_routine(void *fd) {
    int flag = 1;
    char c;

    FILE *fs = fdopen(*((int *)fd), "r");
    char buf[20];
    setvbuf(fs, buf, _IOFBF, 20);

    while (flag == 1) {
        flag = fscanf(fs, "%c", &c);
        if (flag == 1) {
            fprintf(stdout, "%c", c);
        }
    }
}

int main(void) {
    int fd = open("alphabet.txt", O_RDONLY);

    FILE *fs = fdopen(fd, "r");
    char buf[20];
    setvbuf(fs, buf, _IOFBF, 20);

    pthread_t thr_worker;

    pthread_create(&thr_worker, NULL, thread_routine, &fd);

    int flag = 1;
    char c;
    while (flag == 1) {
        flag = fscanf(fs, "%c", &c);
        if (flag == 1) {
            fprintf(stdout, "%c", c);
        }
    }
    pthread_join(thr_worker, NULL);
```

```
41      return 0;
42  }
```

Результат работы программы:
abcdefghijklmnopqrstuvwxyz

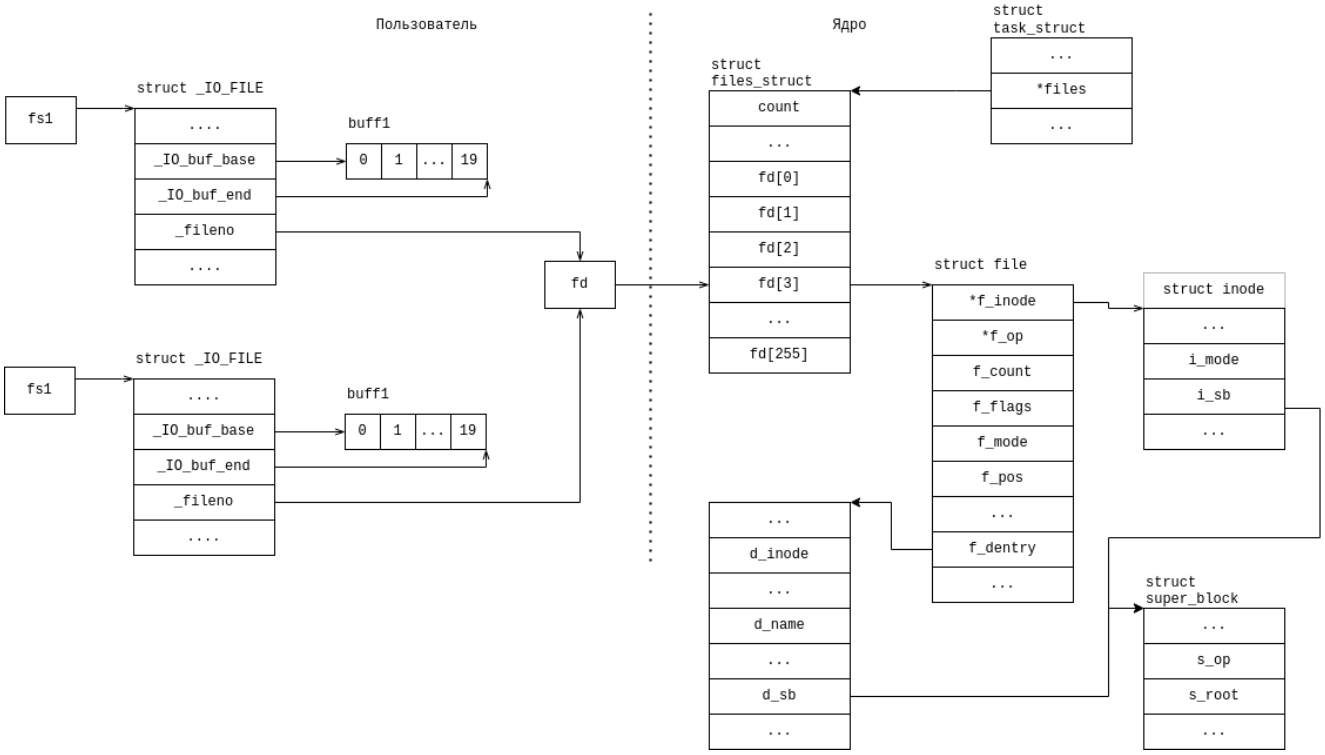
С помощью системного вызова open() создается дескриптор открытого только для чтения файла. Системный вызов open() возвращает индекс в массиве fd структуры files_struct. fdopen() создает экземпляры структуры типа FILE(fs1 и fs2), которые ссылаются на дескриптор, созданный вызовом open. Далее создаются буферы buff1 и buff2 размером 20 байт. Для дескрипторов fs1 и fs2 помощью setbuv задаются соответствующие буферы и тип буферизации _IOFBF(полная буферизация).

Далее fscanf() выполняется в цикле 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 считаются первые 20 символов (abcdefghijklmnopqrst), в переменную с записывается, 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 переменную с записывается символ 'u'.

Внутри цикла будут поочередно выводится символы из buff1 и buff2 до тех пор, пока символы в одном из буферов не закончатся. Тогда на экран будут последовательно выведены оставшиеся символы из другого буфера.

Связь структур:



Программа №2

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

int main() {
    int fd1 = open("alphabet.txt", O_RDONLY);
    int fd2 = open("alphabet.txt", O_RDONLY);

    char c;

    while (read(fd1, &c, 1) == 1 && read(fd2, &c, 1) == 1) {
        write(1, &c, 1);
        write(1, &c, 1);
    }

    return 0;
}
```

Результат работы программы:

aabbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz

Программа с дополнительным потоком:

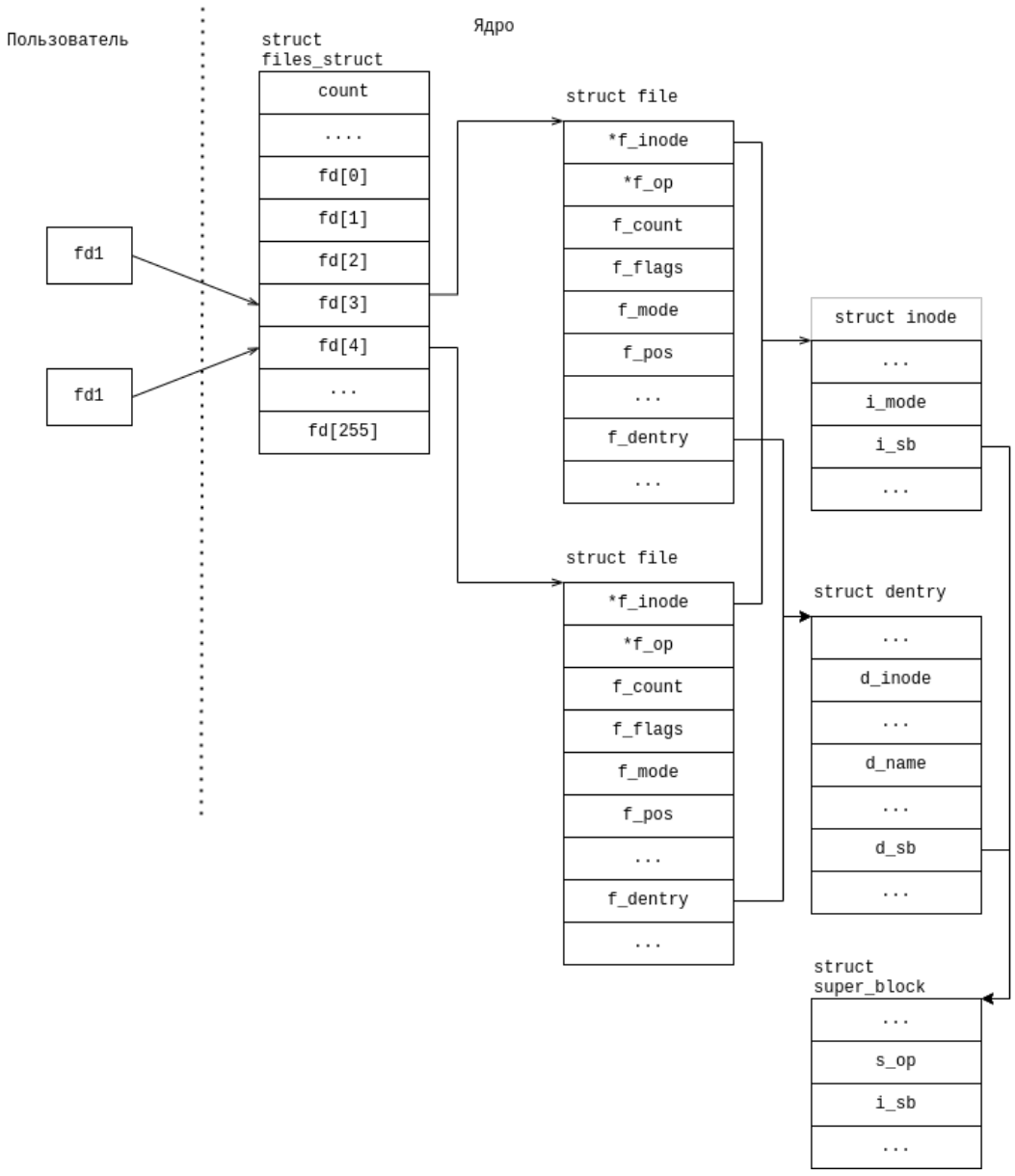
```c
#include <fcntl.h>
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

void *thread_routine(void *arg) {
    int fd = *((int *)arg);

    int flag = 1;
    char c;

    while (flag == 1) {
        flag = read(fd, &c, 1);
        if (flag == 1) write(1, &c, 1);
    }
}

int main() {
    int fd1 = open("alphabet.txt", O_RDONLY);
    int fd2 = open("alphabet.txt", O_RDONLY);

    pthread_t thr_worker;

    pthread_create(&thr_worker, NULL, thread_routine, &fd1);

    int flag = 1;
    char c;

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

    pthread_join(thr_worker, NULL);

    return 0;
}
```

Результат работы программы:

abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz

В программе один и тот же файл открыт 2 раза для чтения. При вызове системного вызова open() создается дескриптор файла в системной таблице файлов, открытых процессом и запись в системной таблице открытых файлов. Так как в данном случае файл открывается 2 раза, то в таблице открытых процессом файлов будет 2 дескриптора и каждый такой дескриптор имеет собственный f_pos. По этой причине чтение становится независимым – при вызове read() для обоих дескрипторов по очереди, оба указателя проходят по всем позициям файла, и каждый символ считывается и выводится по два раза. Несмотря на то, что существует 2 дескриптора открытого файла, открывается один и тот же файл, т.е. inode один и тот же.

Связь структур:



Программа №3

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.

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  int main() {
6      FILE *f1 = fopen("out.txt", "w");
7      FILE *f2 = fopen("out.txt", "w");
8
9      for (char letr = 'a'; letr < '{'; letr++) {
10         letr % 2 ? fprintf(f1, "%c", letr) : fprintf(f2, "%c", letr);
11     }
12
13     fclose(f2);
14     fclose(f1);
15
16     return 0;
17 }
```

Результат работы программы:
acegikmoqsuwy

Файл out.txt открывается функцией fopen() на запись дважды. Создается два дескриптора открытых файлов, две независимые позиции, но с одним и тем же inode. Функция fprintf() (функция записи в файл) самостоятельно создаёт буфер, в который заносимая в файл информация первоначально и помещается. Из буфера информация переписывается в результате трех действий:

1. Информация из буфера записывается в файл когда буфер полон. В этом случае содержимое буфера автоматически переписывается в файл.

2. Если вызвана fflush - принудительная запись содержимого в файл.

3. Если вызвана fclose.

В данном случае запись в файл происходит в результате вызова функции fclose. При вызове fclose() для fs1 буфер для fs1 записывается в файл. При вызове fclose() для fs2, все содержимое файла очищается, а в файл записывается содержимое буфера для fs2. В итоге произошла потеря данных, в файле окажется только содержимое буфера для fs2.

Решение. Необходимо использовать open() с флагом O_APPEND. Если этот флаг установлен, то каждой операции добавления гарантируется неделимость.

Программа с дополнительным потоком

```
1  #include <fcntl.h>
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <sys/stat.h>
5  #include <unistd.h>
6
7  struct stat statbuf;
8
9  void *thread_routine() {
10     FILE *f2 = fopen("out.txt", "a");
11     stat("out.txt", &statbuf);
12     printf("open for fs2: inode  = %ld, buffsize = %ld blocksize=
         %ld\n",
13     (long int)statbuf.st_ino, (long int)statbuf.st_size,
14     (long int)statbuf.st_blksize);
15
16     for (char letr = 'a'; letr < '{'; letr += 2) fprintf(f2, "%c",
         letr);
17     fclose(f2);
18     stat("out.txt", &statbuf);
19     printf("close for fs2: inode  = %ld, buffsize = %ld blocksize=
         %ld\n",
20       (long int)statbuf.st_ino, (long int)statbuf.st_size,
21       (long int)statbuf.st_blksize);
22 }
23
24 int main() {
25     FILE *f1 = fopen("out.txt", "a");
26     stat("out.txt", &statbuf);
27     printf("open for fs1: inode  = %ld, buffsize = %ld blocksize=
         %ld\n",
28     (long int)statbuf.st_ino, (long int)statbuf.st_size,
```

```
29      (long int)statbuf.st_blksize);
30
31      pthread_t thr_worker;
32      pthread_create(&thr_worker, NULL, thread_routine, f1);
33      pthread_join(thr_worker, NULL);
34
35      for (char letr = 'a'; letr < '{'; letr += 2) fprintf(f1, "%c",
            letr);
36
37      fclose(f1);
38      stat("out.txt", &statbuf);
39      printf("close for fs2: inode  = %ld, buffsize = %ld blocksize=
          %ld\n",
40      (long int)statbuf.st_ino, (long int)statbuf.st_size,
41      (long int)statbuf.st_blksize);
42
43      return 0;
44  }
```

Результат работы программы:

acegikmoqsuwyacegikmoqsuwy

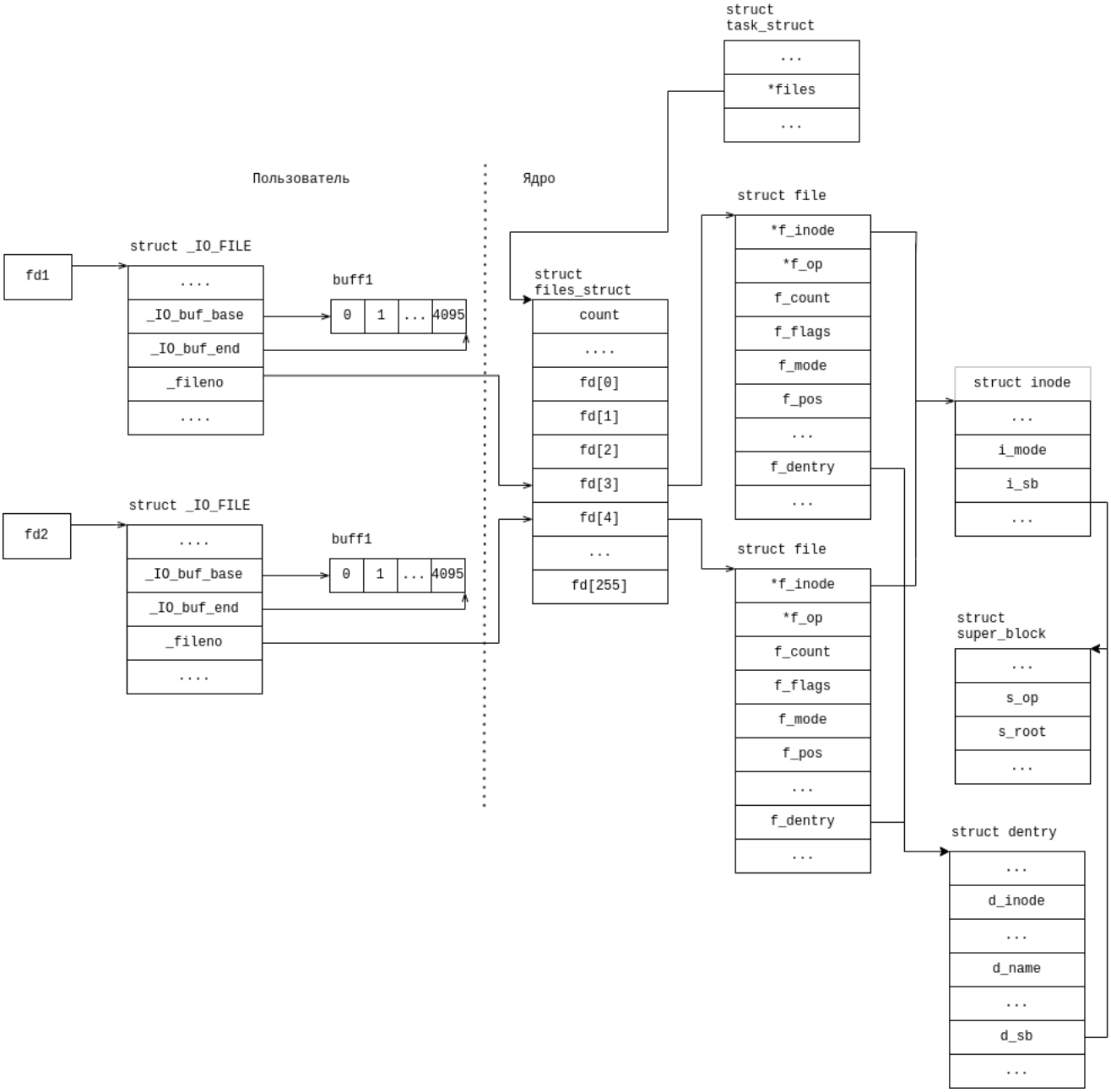
open for fs1: inode = 6964209, buffsize = 13 blocksize= 4096

open for fs2: inode = 6964209, buffsize = 13 blocksize= 4096

close for fs2: inode = 6964209, buffsize = 26 blocksize= 4096

close for fs2: inode = 6964209, buffsize = 39 blocksize= 4096

Связь структур:



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

int main() {
    int fd1 = open("out.txt", O_RDWR);
    int fd2 = open("out.txt", O_RDWR);

    for (char letr = 'a'; letr < '{'; letr++) {
        letr % 2 ? write(fd1, &letr, 1) : write(fd2, &letr, 1);
    }

    close(fd2);
    close(fd1);

    return 0;
}
```

Результат работы программы:

bdfhjlnprtvxz

Создаются 2 файловых дескриптора в системной таблице открытых файлов struct_file для одного файла. Каждый имеет своё поле f_pos и при записи в файл происходит потеря данных.