



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«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»

Лабораторная работа № 5

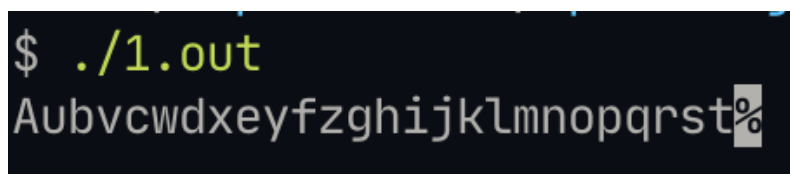
Дисциплина	Операционные системы.
Тема	Буферизованный и небуферизованный ввод-вывод
Студент	Степанов А. О.
Группа	ИУ7-63Б
Оценка (баллы)	
Преподаватель	Рязанова Н.Ю.

Москва, 2020 г.

ЗАДАНИЕ 1

Листинг 1: Текст программы задания 1

```
1 #include <stdio.h>
2 #include <fcntl.h>
3
4 int main()
5 {
6     int fd = open("alphabet.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
18    while(flag1 == 1 || flag2 == 1)
19    {
20        char c;
21
22        flag1 = fscanf(fs1, "%c", &c);
23        if (flag1 == 1)
24            fprintf(stdout, "%c", c);
25
26        flag2 = fscanf(fs2, "%c", &c);
27        if (flag2 == 1)
28            fprintf(stdout, "%c", c);
29    }
30
31    return 0;
32 }
```



```
$ ./1.out
Aubvcwdxeyfzghijklmnopqrst%
```

Рис. 1: Результат работы программы 1

В результате использования системного вызова `open()` создается дескриптор файла, который открывается только на чтение, указатель устанавливается на начало файла. В результате появляется запись в системной таблице открытых файлов. Возвращенный системным вызовом файловый дескриптор является наименьшим, который еще не открыт процессом.

Функция `fdopen()` связывает поток с существующим дескриптором файла. Функция `setvbuf()` изменяет тип буферизации на блочную. В данной программе связываются два потока с одним дескриптором и устанавливается блочная буферизация размером в 20 байт.

Далее в цикле происходит чтение из потоков и вывод в `stdout`. Системная функция `fscanf()` возвращает -1, если число прочитанных символов равно нулю, и 1 в ином случае.

Поскольку размер буфера был установлен в 20 байт, то в буфер `buff1` помещается строка `Abcdefghijklmnopqrst`, а в буфер `buff2` – `uvwxyz`. В результате поочередного вывода в цикле получается такой вывод: `Aubvcwdkeyfzghijklmnopqrst`.

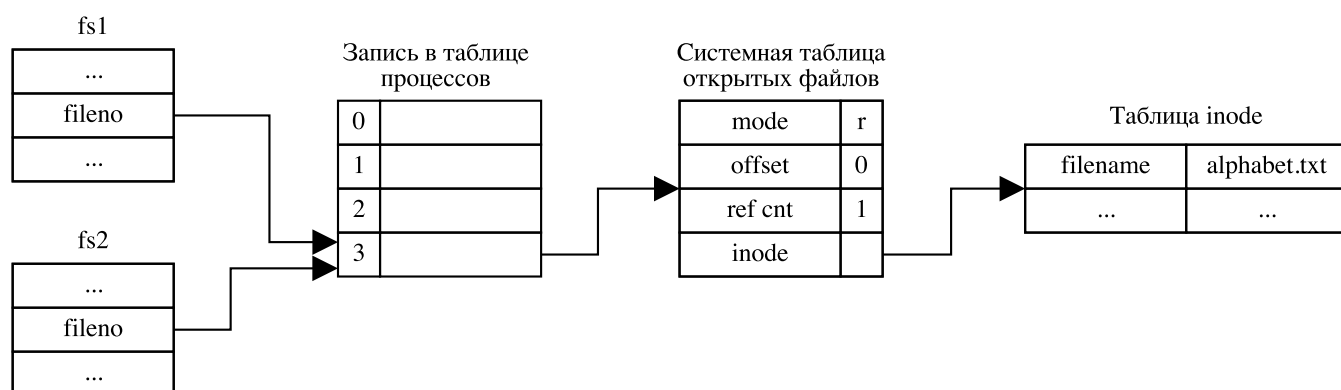


Рис. 2: Схема связи дескрипторов в программе 1

ЗАДАНИЕ 2

Листинг 2: Текст программы задания 2

```
1 #include <fcntl.h>
2 #include <unistd.h>
3
4 int main()
5 {
```

```

6   char c;
7   int cond1 = 1, cond2 = 1;
8   int fd1 = open("alphabet.txt", O_RDONLY);
9   int fd2 = open("alphabet.txt", O_RDONLY);
10
11  while(cond1 || cond2)
12  {
13      if ((cond1 = read(fd1, &c, 1)) == 1)
14          write(1, &c, 1);
15
16      if ((cond2 = read(fd2, &c, 1)) == 1)
17          write(1, &c, 1);
18  }
19
20  return 0;
21 }

```

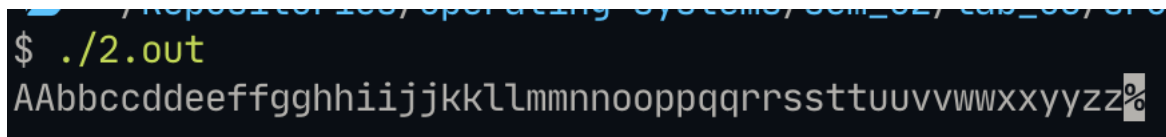


Рис. 3: Результат работы программы 2

В данной программе создается два файловых дескриптора и, соответственно, две разные записи в таблице открытых файлов. Поскольку это два разных файловых дескриптора, то положения указателей в них будут независимы. В результате получается, что на каждой итерации цикла выводится два одинаковых символа. Получаем соответствующий результат с дублирующимися символами.

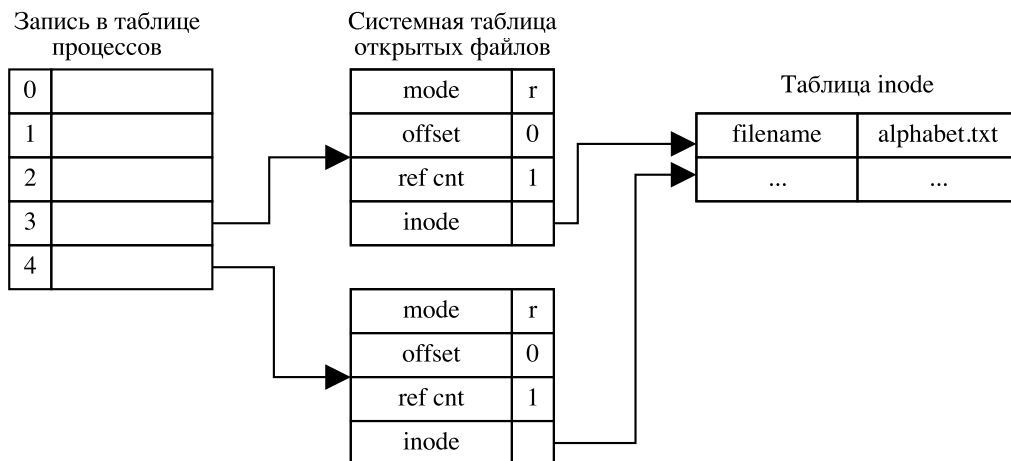


Рис. 4: Схема связи дескрипторов в программе 2

ЗАДАНИЕ 3

Листинг 3: Текст программы задания 3

```
1 #include <stdio.h>
2
3 int main()
4 {
5     const char letters[] = "Abcdefghijklmnopqrstuvwxyz";
6     FILE *fd[2];
7     fd[0] = fopen("out.txt", "wr");
8     fd[1] = fopen("out.txt", "wr");
9
10    for (int i = 0; i < sizeof(letters) - 1; ++i)
11    {
12        fprintf(fd[i % 2], "%c", letters[i]);
13    }
14
15    fclose(fd[0]);
16    fclose(fd[1]);
17
18    return 0;
19 }
```

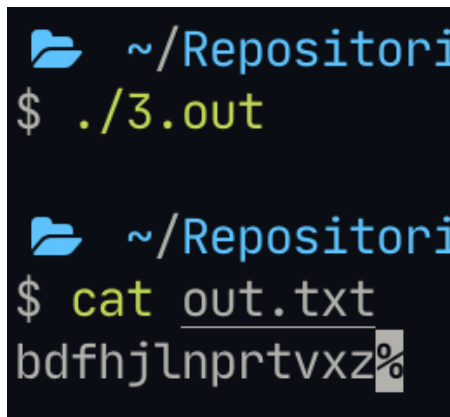


Рис. 5: Результат работы программы 3

С помощью функции `fopen()` открываются два потока на запись. Два потока имеют разные файловые дескрипторы, а, следовательно, независимые указатели в файле. После открытия файлов, в цикле поочередно выводятся символы в разные потоки. Четные (с индексами 0, 2, 3 и т.д.) записываются в первый буфер, это буквы а, с, е, г и т.д. Нечетные (с индексами 1, 3, 5 и т.д.) записываются во второй буфер, это буквы b, d, f и т.д.

Функция `fprintf()` организует буферизованный вывод, поэтому предполагается, что данные записаны в файл, но реально данные еще там отсутствуют.

Затем происходит вызов функции `fclose()` для каждого буфера. Эта функция отделяет поток от файла. Если поток использовался для вывода, то все данные, содержащиеся в буфере принудительно запишутся в файл с использованием функции `fflush()`. Поскольку оба потока открыты на запись, то после выполнения второго `fclose()`, данные, записанные в файл из первого потока, будут перезаписаны данными из второго. Поэтому в файле мы увидим буквы b, d, f и т.д.

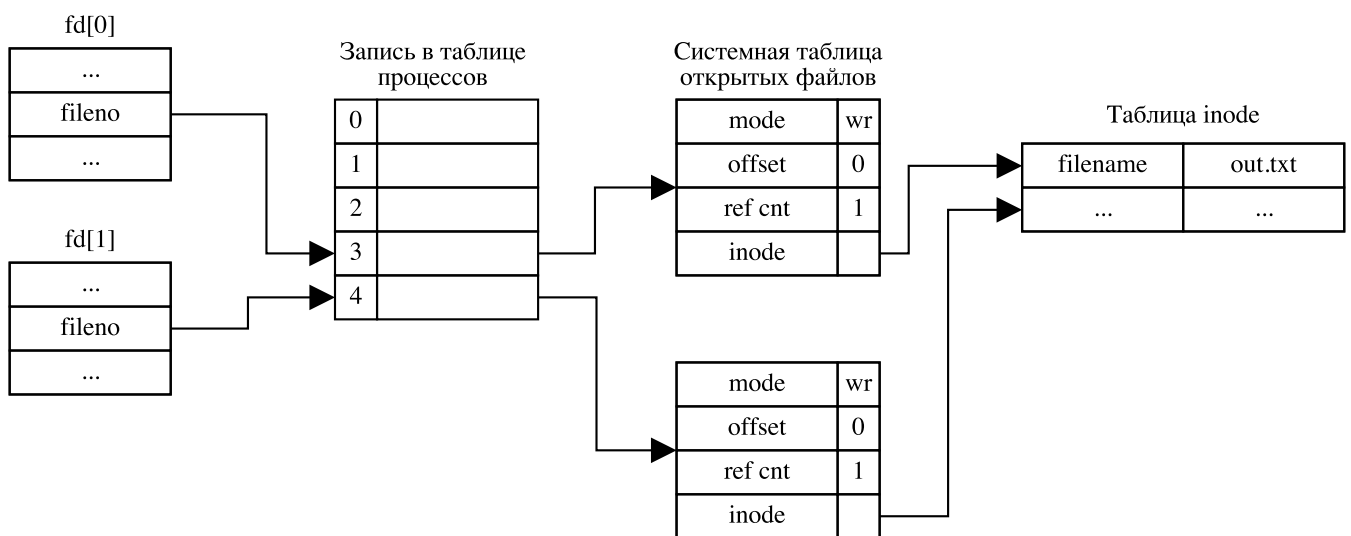


Рис. 6: Схема связи дескрипторов в программе 3

СТРУКТУРА FILE

Листинг 4: Структура `file`

```

1 struct file {
2     union {
3         struct llist_node    fu_llist;
4         struct rcu_head      fu_rcuhead;
5     } f_u;
6     struct path              f_path;
7     struct inode              *f_inode;    /* cached value */
8     const struct file_operations *f_op;
9
10    /*
```

```

11      * Protects f_ep_links, f_flags.
12      * Must not be taken from IRQ context.
13      */
14      spinlock_t      f_lock;
15      enum rw_hint     f_write_hint;
16      atomic_long_t    f_count;
17      unsigned int     f_flags;
18      fmode_t          f_mode;
19      struct mutex      f_pos_lock;
20      loff_t           f_pos;
21      struct fown_struct f_owner;
22      const struct cred *f_cred;
23      struct file_ra_state f_ra;
24
25      u64              f_version;
26  \#ifdef CONFIG_SECURITY
27      void             *f_security;
28  \#endif
29      /* needed for tty driver, and maybe others */
30      void             *private_data;
31
32  \#ifdef CONFIG_EPOLL
33      /* Used by fs/eventpoll.c to link all the hooks to this file */
34      struct list_head  f_ep_links;
35      struct list_head  f_tfile_llink;
36  \#endif /* #ifdef CONFIG_EPOLL */
37      struct address_space *f_mapping;
38      errseq_t          f_wb_err;
39  }

```