



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

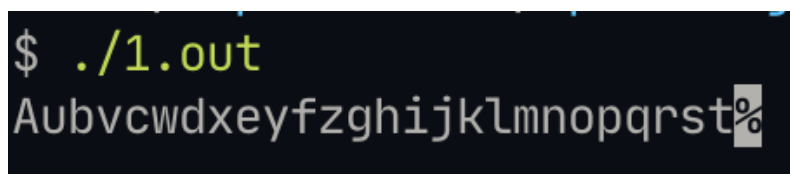
Дисциплина	Операционные системы.
Тема	Буферизованный и небуферизованный ввод-вывод
Студент	Степанов А. О.
Группа	ИУ7-63Б
Оценка (баллы)	
Преподаватель	Рязанова Н.Ю.

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`.

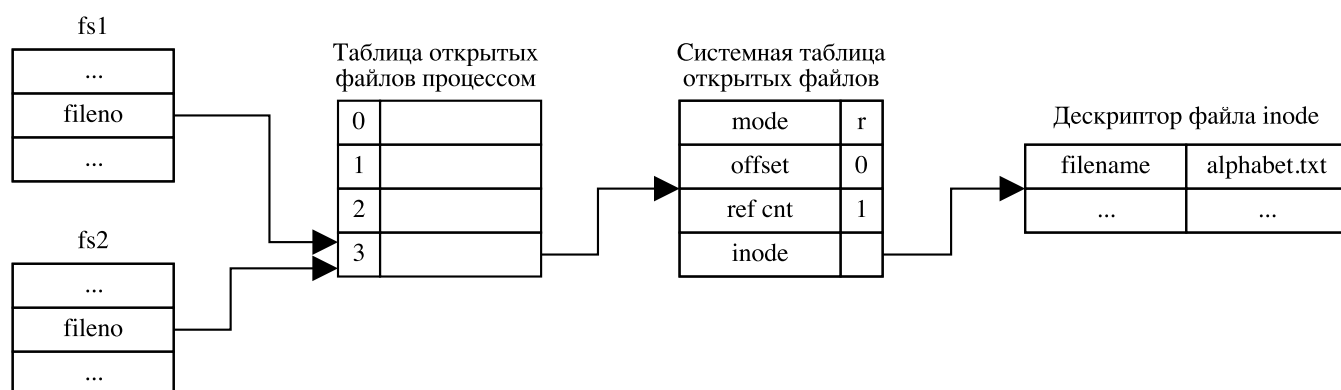


Рис. 2: Схема связи дескрипторов в программе 1

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 }

```

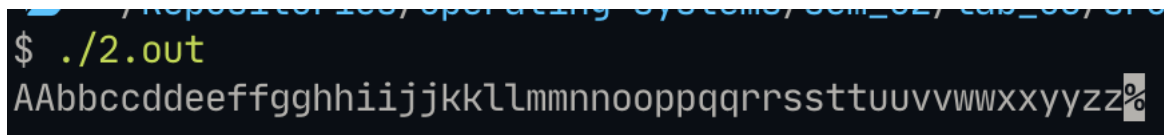


Рис. 3: Результат работы программы 2

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.

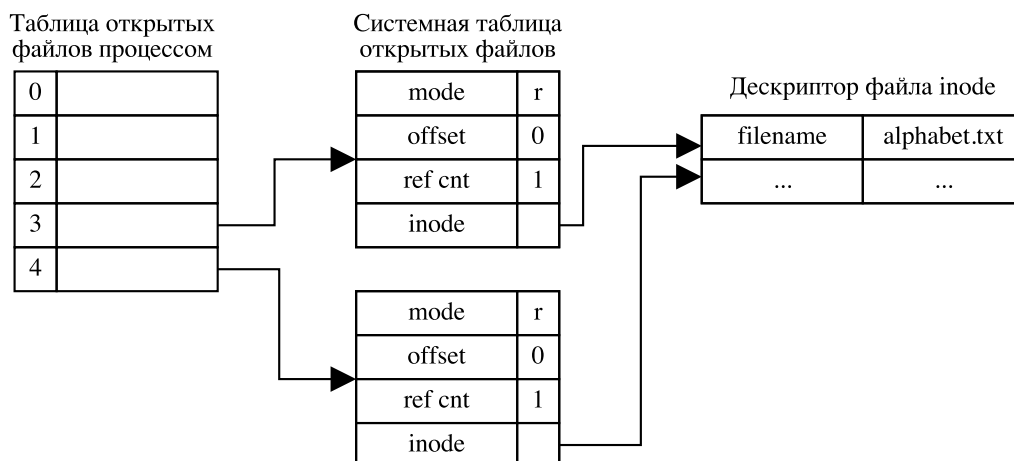


Рис. 4: Схема связи дескрипторов в программе 2

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", "w");
8     fd[1] = fopen("out.txt", "w");
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 }
```

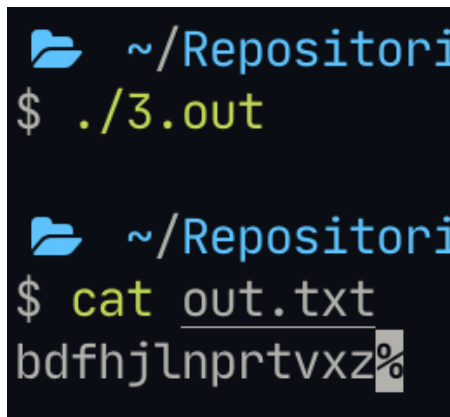


Рис. 5: Результат работы программы 3

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.

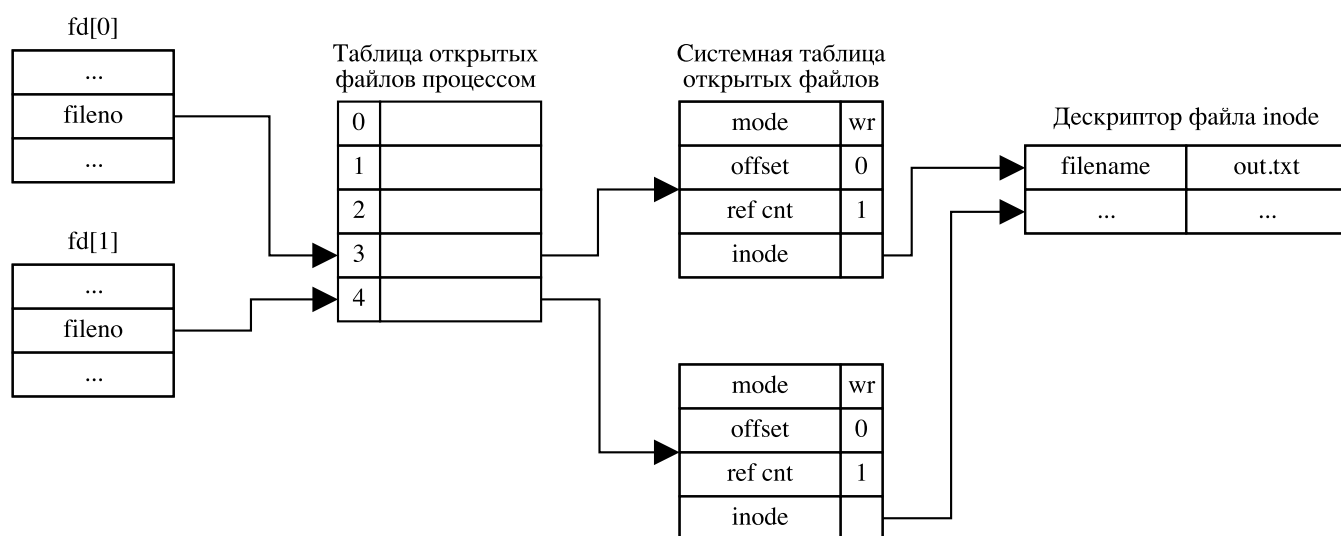


Рис. 6: Схема связи дескрипторов в программе 3

СТРУКТУРА FILE

Листинг 4: Структура `_IO_FILE`

```

1 struct _IO_FILE
2 {
3     int _flags;           /* High-order word is _IO_MAGIC; rest is flags. */
4     /* The following pointers correspond to the C++ streambuf protocol. */
5     char *_IO_read_ptr;   /* Current read pointer */
6     char *_IO_read_end;   /* End of get area. */
7     char *_IO_read_base;  /* Start of putback+get area. */
8     char *_IO_write_base; /* Start of put area. */
9     char *_IO_write_ptr;  /* Current put pointer. */
10    char *_IO_write_end;   /* End of put area. */

```

```

11  char *_IO_buf_base;          /* Start of reserve area. */
12  char *_IO_buf_end;          /* End of reserve area. */
13  /* The following fields are used to support backing up and undo. */
14  char *_IO_save_base; /* Pointer to start of non-current get area. */
15  char *_IO_backup_base;
16  /* 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 };

```

Листинг 5: typedef в файле FILE.h

```

1  typedef struct _IO_FILE FILE;

```

ВЫВОДЫ

Исходя из рассуждений по данной лабораторной можно сделать следующие выводы:

- Функции `fopen()`, `fclose()`, `fscanf()`, `fprintf()` являются обертками, которые выполняют системные вызовы `open()`, `close()`, `read()`, `write()` соответственно;
- Лучше использовать функцию `fopen()`, чем `open()`, так как `fopen()` выполняет ввод-вывод с буферизацией, что может оказаться значительно быстрее, так же структура `FILE` позволяет использовать функции `stdio.h`;
- Стоит помнить о буферизации и вовремя использовать `fclose()` для записи буферизованных данных в файл;
- Необходимо следить за режимом, в котором открывается поток.