

Московский государственный технический университет
имени Н. Э. Баумана



Факультет: Информатика и системы управления

Кафедра: Программное обеспечение ЭВМ и информационные технологии

Дисциплина: Операционные системы

Лабораторная работа №4
Анализ особенностей работы функций ввода-вывода в
UNIX/Linux

Выполнила: Щербатюк Дарья
Группа: ИУ7-64

Москва, 2018 г.

1 testCIO.c

Текст программы

```
1 //testCIO.c
2 #include <stdio.h>
3 #include <fcntl.h>
4
5 /*
6  On my machine, a buffer size of 20 bytes
7  translated into a 12-character buffer.
8  Apparently 8 bytes were used up by the
9  stdio library for bookkeeping.
10 */
11
12 int main()
13 {
14     // have kernel open connection to file alphabet.txt
15     int fd = open("alphabet.txt", O_RDONLY);
16
17     // create two a C I/O buffered streams using the above connection
18     // associates a stream with the existing file descriptor
19     FILE *fs1 = fdopen(fd, "r");
20     char buff1[20];
21     setvbuf(fs1, buff1, _IOFBF, 20); // set fully buffering
22
23     FILE *fs2 = fdopen(fd, "r");
24     char buff2[20];
25     setvbuf(fs2, buff2, _IOFBF, 20);
26
27     // read a char and write it alternately from fs1 and fs2
28     int flag1 = 1, flag2 = 2;
29     while(flag1 == 1 || flag2 == 1)
30     {
31         char c;
32         flag1 = fscanf(fs1, "%c", &c);
33
34         if (flag1 == 1) { fprintf(stdout, "%c", c); }
35         flag2 = fscanf(fs2, "%c", &c);
36         printf("%d\n", flag2);
37         if (flag2 == 1) { fprintf(stdout, "%c", c); }
38     }
39
40     return 0;
41 }
```

Вывод

```
1 Aubvcwdxeyfzghijklmnopqrst
```

Анализ

С помощью системного вызова *open()* создается дескриптор файла, файл открывается только на чтение, указатель устанавливается на начало файла. Если системный вызов завершается успешно, возвращенный файловый дескриптор является наименьшим, который еще не открыт процессом. В результате этого вызова появляется новый открытый файл, не разделяемый никакими процессами, и запись в системной таблице открытых файлов.

Далее функция *fdopen()* связывает два потока с существующим дескриптором файла. Функция *setvbuf()* изменяет тип буферизации на блочную (полную) размером в 20 байт.

В цикле осуществляется чтение из потоков и вывод в *stdoutc* помощью системных функций *fscanf*, *fprintf*. Флаги *flag1*, *flag2* изменяют свое значение с 1 на -1 тогда, когда число прочитанных символов станет равно нулю. Стоит помнить о том, что открытые файлы, для которых используется ввод/вывод потоков, буферизуются. Т.к. размер буфера установлен в 20 байт, по факту в *buff1* помещается строка *Abcdefghijklmnopqrst*, а в *buff2* - *uvwxyz*. В результате поочередного вывода из каждого буфера потока получим строку выше.

2 testKernelIO.c

Текст программы

```
1 //testKernelIO.c
2 #include <fcntl.h>
3
4 int main()
5 {
6     // have kernel open two connection to file alphabet.txt
7     int fd1 = open("alphabet.txt",O_RDONLY);
8     int fd2 = open("alphabet.txt",O_RDONLY);
9
10
11     // read a char & write it alternately from connections fs1 & fd2
12     while(1)
13     {
14         char c;
15         if (read(fd1,&c,1) != 1) break;
16         write(1,&c,1);
17         if (read(fd2,&c,1) != 1) break;
18         write(1,&c,1);
19     }
20
21     return 0;
22 }
```

Вывод

```
1 AAbbccddeeffgghhiijjkkllmmnnnooppqrrssttuuvvwxxxyzz
```

Анализ

В отличие от первой программы, создаются два файловых дескриптора и две разные записи в системной таблице открытых файлов. Файловые дескрипторы независимы друг от друга, поэтому положения указателей в файле, связанных с данным файловым дескриптором, будут независимы. В результате прохода цикла, где с помощью системных вызовов *read*, *write* в *stdout* выводится по одному символу из файла, увидим строку с дублирующимися буквами.

3 testFOpen.c

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

Вывод

```
1 bdfhjlnprtvxz
```

4 Вывод