

Программа 1

Листинг 1: testCIO

```
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    while(flag1 == 1 || flag2 == 1)
18    {
19        char c;
20        flag1 = fscanf(fs1, "%c", &c);
21        if (flag1 == 1)
22        {
23            fprintf(stdout, "%c", c);
24        }
25        flag2 = fscanf(fs2, "%c", &c);
26        if (flag2 == 1)
27        {
28            fprintf(stdout, "%c", c);
29        }
30    }
31    return 0;
32 }
```

Результат работы программы:

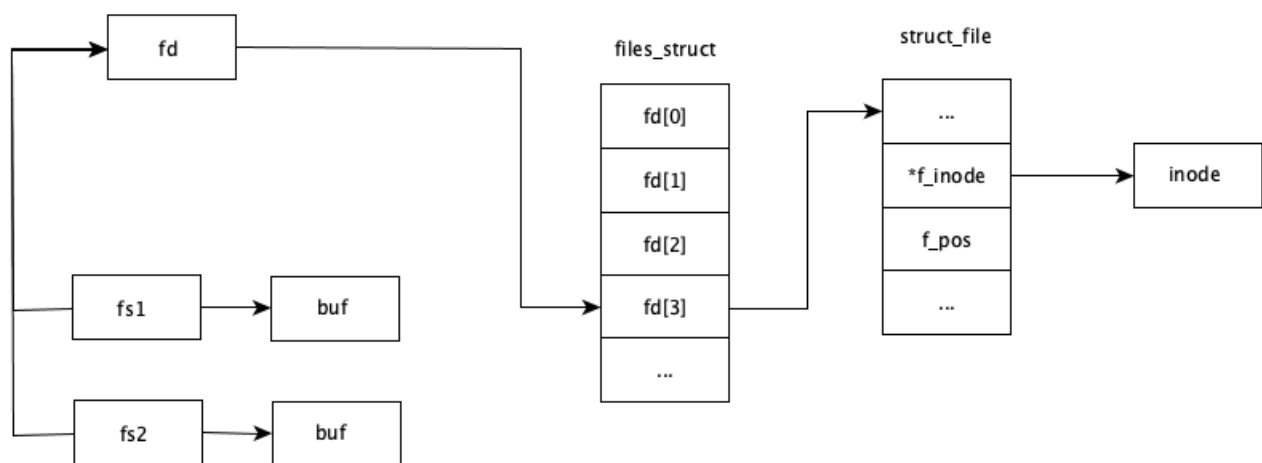
```
olga@olga-VirtualBox:~/Documents/lr5$ ./testCIO
aubvcwdxyfzg
hijklmnopqrstolga@olga-VirtualBox:~/Documents/lr5$
```

Анализ работы программы

В самом начале программы создается дескриптор файла “alphabet.txt” в таблице файловых дескрипторов процесса и запись в таблице открытых файлов. В записи этой таблицы хранится текущее смещение указателя в файле, которое используется во всех операциях чтения и записи в файл, а также режим открытия файла (O_RDONLY). При этом указатель устанавливается на начало файла. При выполнении операции чтения-записи система выполняет неявный сдвиг указателя.

Далее при помощи системного вызова `fdopen()` создаются два объекта типа `FILE`, которые ссылаются на созданный файловый дескриптор. При помощи функций `setvbuf()` задается размер буфера каждого объекта равный 20 символам. При первом вызове `fscanf()`, буфер первого файлового дескриптора заполняется первыми 20 символами - `abcdefghijklmnopqrst`, а при втором вызове буфер второго файлового дескриптора заполняется оставшимися 6 символами – `vwxyz` (т. к. обе структуры `FILE` ссылаются на одну и ту же запись в таблице открытых файлов, а при первом чтении указатель в файле уже сместился на 20 позиций). Далее в цикле происходит поочередная печать по одному символу, который уже берется не из файла, а из соответствующего буфера файлового дескриптора.

Связь структур:

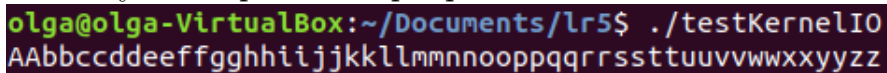


Программа 2

Листинг 2: testKernelIO

```
1 #include <fcntl.h>
2
3 int main()
4 {
5     char c;
6     int fd1 = open("alphabet.txt", O_RDONLY);
7     int fd2 = open("alphabet.txt", O_RDONLY);
8
9     char c1, c2;
10    while ((read(fd1, &c1, 1)) && (read(fd2, &c2, 1)))
11    {
12        write(1, &c1, 1);
13        write(1, &c2, 1);
14    }
15
16    close(fd1);
17    close(fd2);
18
19    return 0;
20 }
```

Результат работы программы:

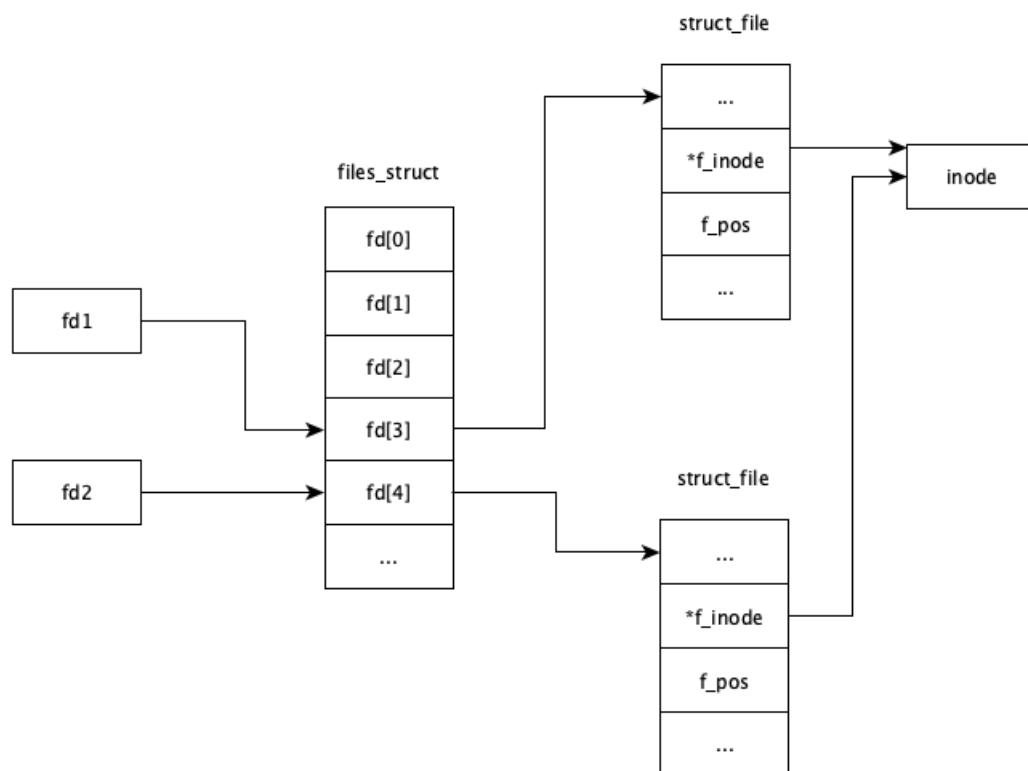


```
olga@olga-VirtualBox:~/Documents/lr5$ ./testKernelIO
AAbbccddeeffgghhiijjkkllmmnnooppqrrssttuuvvwwxxyyzz
```

Анализ работы программы

В данной программе создаются два дескриптора файла “alphabet.txt” и две различные записи в таблице открытых файлов системы. В этом случае, каждая запись имеет свое независимое смещение в файле. Поэтому при поочередной печати символа в цикле каждый указатель смещается независимо от другого указателя. В результате получается строка с дублированными символами.

Связь структур:



Программа 3

Листинг 3: testfopen

```
1 #include <stdio.h>
2
3 int main()
4 {
5     FILE *fd[2];
6
7     fd[0] = fopen("FOpen_output.txt", "w");
8     fd[1] = fopen("FOpen_output.txt", "w");
9
10    int curr = 0;
11    for(char c = 'a'; c <= 'z'; c++, curr = ((curr != 0) ? 0 : 1))
12    {
13        fprintf(fd[curr], "%c", c);
14    }
15
16    fclose(fd[0]);
17    fclose(fd[1]);
18
19    return 0;
20 }
21 }
```

Результат работы программы:

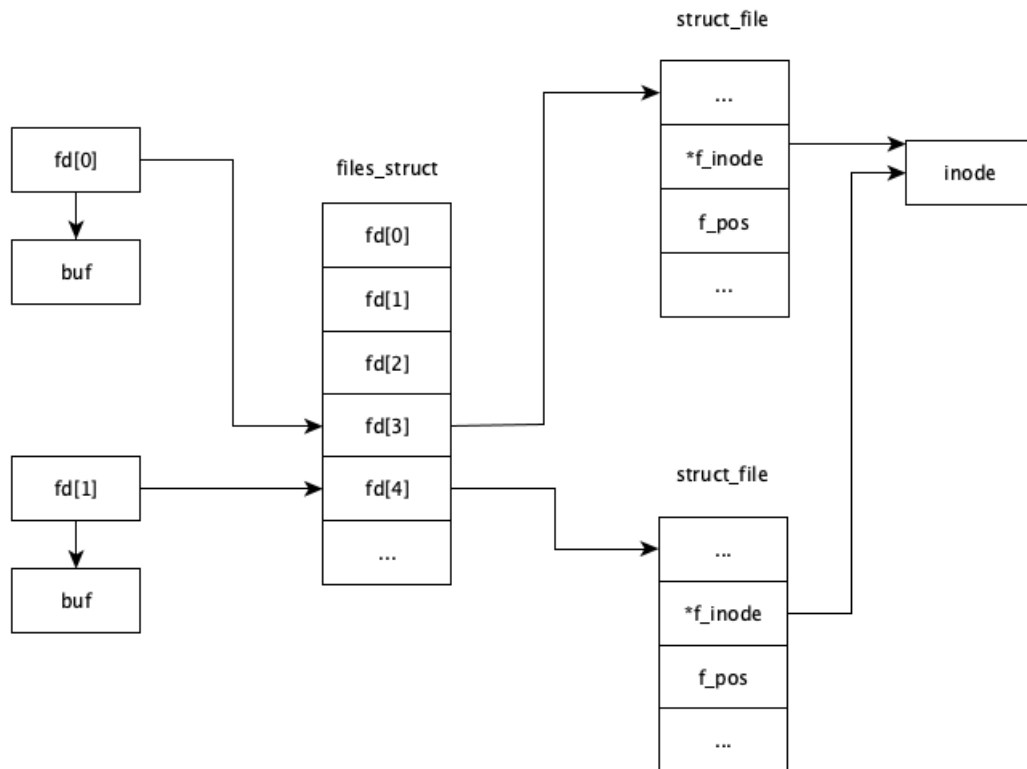
bdfhjlnprtvxz

Анализ работы программы

С помощью функции `fopen()` создаются два потока на запись с начала файла. Они имеют два разных файловых дескриптора и следовательно независимые позиции в файле. Затем в цикле с помощью `fprintf()` в потоки поочередно записываются буквы - в первый поток нечетные, а во второй четные. Функция `fprintf()` обеспечивает буферизацию. Запись непосредственно в сам файл происходит либо при полном заполнении буфера, либо при вызове функций `fclose()` или `fflush()`.

Когда в программе вызвалась функция `fclose(fd[0])`, в файл записались все буквы английского алфавита, которые были в буфере файлового дескриптора `fd[0]` (`asegikmoqsuwy`), тогда как после вызова функции `fclose(fd[1])`, содержимое файла удалилось (т.к. мы открыли файл для записи с режимом “w”), и записалась информация из буфера `fd[0]`.

Связь структур:



Структура FILE

```
struct FILE {
    ssize_t          _cnt; // число байт в буфере _cnt
    unsigned char    *_ptr; // указатель на следующий символ,
                          // который подлежит чтению или записи _ptr
    unsigned char    *_base; // указатель на буфер _base
    unsigned char    _flag; // флаги состояния потока _flag
    unsigned char    _file; // указатель на файловый дескриптор _file,
                          // с которым ассоциирован данный поток
    ...
};
```