



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 и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №5

По предмету: «Операционные системы»

Тема: Буферизованный и не буферизованный ВВОД-ВЫВОД

Преподаватель: Рязанова Н.Ю.

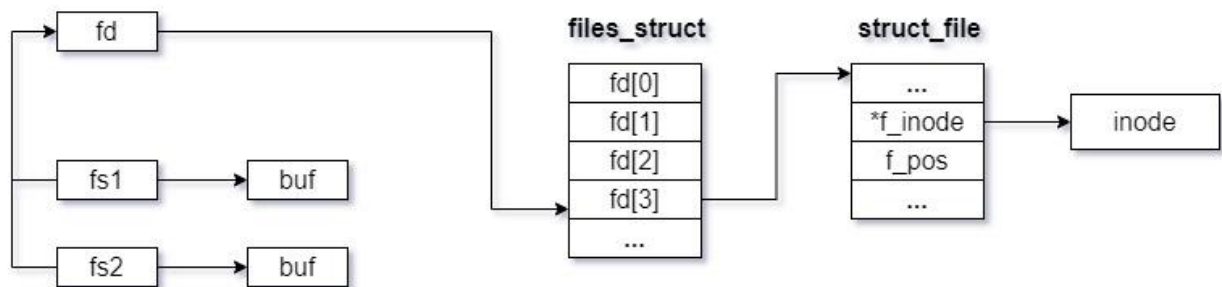
Студент: Гасанзаде М.А.,

Группа: ИУ7-66Б

Москва, 2020 г.

Программа №1

Связь структур



Анализ

С помощью системного вызова **open()** создается дескриптор файла. Файл открывается только на чтение, так как передается флаг **O_RDONLY**.

При успешном завершении системного вызова появляется новый дескриптор открытого файла **alphabet.txt** и запись в системной таблице открытых файлов.

Функция **fdopen()** создает два объекта типа **FILE**, которые связываются с открытым файлом, на который ссылается файловый дескриптор **fd**.

```
struct FILE {
    ssize_t _cnt;           // число байт в буфере _cnt
    unsigned char *_ptr;    // указатель на следующий
                           // символ, который подлежит чтению/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
    ...
};
```

При создании буфера размер для данного потока выбирается системой. Чтобы его изменить используется системный вызов **setvbuf()**. В данной программе системный вызов **setvbuf()** изменяет тип буферизации для каждого объекта **FILE** на полную буферизацию, а также явно задает размер буфера 20 байт.

При первом вызове **fscanf()** буфер структуры **FILE** заполняется до тех пор, пока он не будет заполнен полностью, либо пока не будет достигнут конец файла. Так как буфер имеет размер 20 байт, а файл содержит 26 байт данных (26 букв латинского алфавита), то после первого вызова **fscanf()** в буфере первой структуры **FILE** будут находиться первые 20 байт файла (то есть буквы с **A** по **t**). Так как оба объекта **FILE** связаны с одним и тем же файловым дескриптором, то позиция в файле будет определяться для обоих файловых потоков вводом полем **f_pos** структуры **struct file**, на которую ссылается указанный дескриптор файла. Поэтому после второго вызова **fscanf()** в буфере второй структуры **FILE** окажутся последние 6 байт файла (то есть символы **s** и **o** по **z**).

Затем в стандартный поток вывода **stdout** будет поочередно осуществляться вывод по одному символу из каждого буфера. Когда второй буфер опустеет, из первого буфера продолжат выводиться оставшиеся символы.

1. Листинг

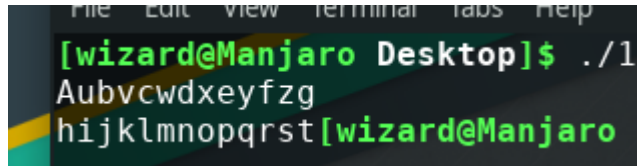
```
#include <stdio.h>
#include <fcntl.h>

int main()
{
    // have kernel open connection to our our file
    int fd = open("alphabet.txt", O_RDONLY);
    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 alternately from connections fs1 & fd2
    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;
}
```

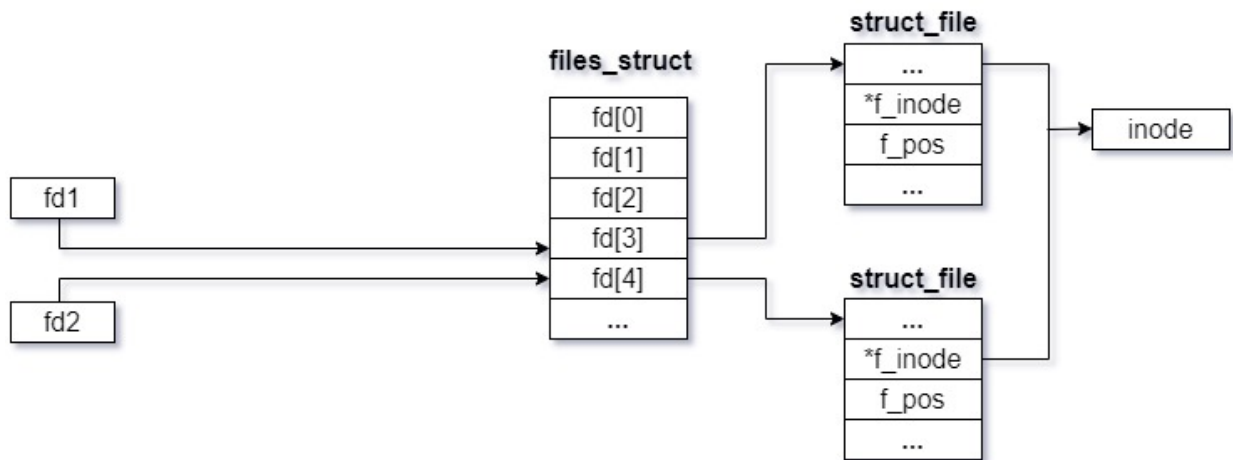
Результат работы программы:



```
File Edit View Terminal Tabs Help
[wizard@Manjaro Desktop]$ ./1
Aubvcwdxeyfzg
hijklmnopqrst[wizard@Manjaro Desktop]$
```

Программа №2

Связь структур



Анализ

В данной программе файл открывается дважды для чтения с помощью системного вызова **open()**. Создаются две различных структуры **struct file**, которые описывают открытый файл, которые связаны с одним и тем же физическим файлом. Текущие позиции в файле для каждой структуры будут изменяться независимо друг от друга. Поэтому чтение с использованием одной структуры не затрагивает текущую позицию в другой структуре, и каждый символ из физического файла будет продублирован.

2. Листинг

```
#include <fcntl.h>
int main() {
    char c;
    // have kernel open for two connection to our file
    int fd1 = open("alphabet.txt", O_RDONLY);
    int fd2 = open("alphabet.txt", O_RDONLY);
    int break_flag = 1;
    // read a char & write it alternatingly from connections fs1 & fd2
    while(break_flag) {
        if (read(fd1, &c, 1) != 1)
            break_flag = 0;
        write(1, &c, 1);

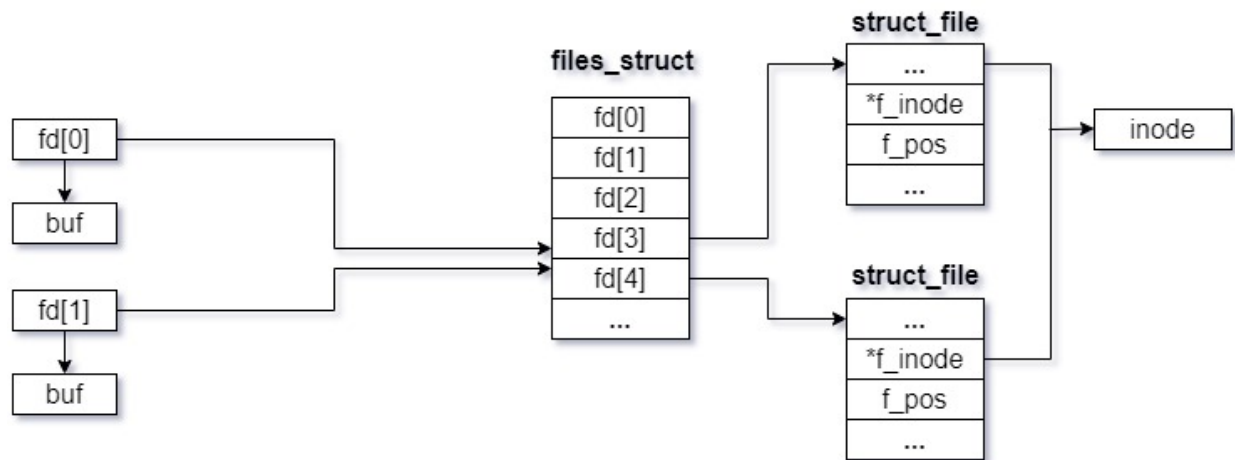
        if (read(fd2, &c, 1) != 1)
            break_flag = 0;
        write(1, &c, 1);
    }
    return 0;
}
```

Результат работы программы:

```
[wizard@Manjaro Desktop]$ ./2
AAbbccddeeffgghhiijjkkllmmnnooppqrrssttuuvvwwxxyyzz
```

Программа №3

Связь структур



Анализ

В данной программе с помощью функции **fopen()** два раза открывается на запись файл `alphabet.txt`, то есть создаются две разных структуры **struct file**, в которых поля **f_pos** при вызове функций ввода/вывода меняются независимо (аналогично предыдущему примеру).

В результате поочередной записи букв латинского алфавита в первый буфер будут записаны нечетные символы, а во второй буфер — четные.

Запись из буфера в файл происходит автоматически в следующих случаях:

- при заполнении буфера:
- по завершении процесса:
- при вызове функций **fclose()** или **fflush()**.

Сначала закрывается первый поток, поэтому изначально в файл `in.txt` осуществляется запись буфера первого потока (`асегіқтмоқсууу`), при этом данные записываются с начала файл, затем происходит закрытие второго потока и запись его буфера в файл `in.txt`. При этом так как оба объекта `FILE` связаны с разными структурами **struct file**, то значения их полей **f_pos** изменяются независимо, и запись второго буфера будет также произведена с начала файла, то есть данные, записанные в файл из буфера первого потока, будут перезаписаны.

3. Листинг

```
#include <stdio.h>

int main()
{
    FILE *f1;
    FILE *f2;

    f1 = fopen("alphabet.txt", "w");
    f2 = fopen("alphabet.txt", "w");
```

```
int flag;
for (char c = 'a'; c <= 'z'; c++)
{
    if (!flag)
        fprintf(f1, "%c", c);

    if (!flag)
        fprintf(f2, "%c", c);
    flag = !flag;
}
fclose(f1);
fclose(f2);

return 0;
}
```

Результат работы программы, файл `alphabet.txt`:

bdfhjlnprtvxz

ВЫВОД

При буферизованном вводе/выводе необходимо учитывать факт записи/чтения данных из буфера, т.к. неправильные действия с данными, записываемыми(или считываемыми) в(из) файл(а), могут привести к неправильной последовательности данных (программа 1) или даже к их потере(программа 3).

Также необходимо учитывать, что при одновременном открытии одного и того же файла создается дескриптор открытого файла. Каждый дескриптор **struct file** имеет поле **f_pos**, указатель на позицию чтения или записи в логическом файле.