Министерство образования и науки 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

ДИСЦИПЛИНА Операционные системы

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по теме: «Анализ особенностей работы функций ввода/вывода в UNIX/Linux»

Студент Покасова А.И.

Группа ИУ7-61

Москва, 2019 г.

Программа 1

Листинг программы:

```
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.               flag1 = fscanf(fs1, "%c", &c);
22.               if (flag1 == 1)
23.               {
24.                    fprintf(stdout, "%c", c);
25.               }
26.
27.               flag2 = fscanf(fs2, "%c", &c);
28.               if (flag2 == 1)
29.               {
30.                    fprintf(stdout, "%c", c);
31.               }
32.          }
33.          fprintf(stdout, "\n");
34.
35.          return 0;
36.     }
```

Вывод:

Анализ программы

В данной программе используется стандартная библиотека «stdio.h», предоставляющая функции буферизованного ввода/вывода. С помощью системного вызова open() создается новый дескриптор открытого файла «alphabet.txt», запись в системной таблице открытых файлов.

```
int open(const char *pathname, int flags);
```

Соответствующая запись заносится в таблицу открытых файлов процесса, описываемую полем struct files_struct *files в структуре struct task_struct, а также в системную таблицу открытых файлов. Связь между этими таблицами отображена на рисунке 1. Файл открывается для чтения, о чем свидетельствует передаваемый в open() флаг O_RDONLY.

С помощью вызова функции fdopen() создаются два объекта типа FILE, которые связываются с открытым файлом, на который ссылается файловый дескриптор fd, передаваемый в функцию fdopen() в качестве аргумента.

```
FILE *fdopen (int *filed, const char *mode);
```

Каждый поток стандартной библиотеки представлен указателем на структуру FILE, в которой хранится указатель на буфер _base, указатель на следующий символ, подлежащий чтению или записи _ptr, число байт в буфере _cnt, указатель на файловый дескриптор _file, с которым ассоциирован данный поток, а также флаги состояния потока _flag:

```
struct FILE

{

      ssize_t        _cnt;
      unsigned char *_ptr
      unsigned char *_base;
      unsigned char _flag;
      unsigned char _file;
      …
};
```

При создании буфера размер для данного потока выбирается системой, однако его можно изменить с помощью вызова 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.

```
int setvbuf(FILE *stream, char *buf, int mode, size_t size);
```

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, то после первого вызова fscanf() в буфере первой структуры FILE будут находиться первые 20 байт файла (первые 20 символов – “abcdefghijklmnopqrst”).

Так как оба объекта FILE связаны с одним и тем же файловым дескриптором, то позиция в файле будет определяться для обоих файловых потоков ввода 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 (uvwxyz).

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.

Программа 2

Листинг программы:

```
1. #include <unistd.h>
2. #include <fcntl.h>
3.
4. int main()
5. {
6.      int fd1 = open("alphabet.txt", O_RDONLY);
7.      int fd2 = open("alphabet.txt", O_RDONLY);
8.
9.      while(1)
10.          {
11.              char c;
12.              if (read(fd1, &c, 1) != 1)
13.              {
14.                  break;
15.              }
16.              write(1, &c, 1);
17.
18.              if (read(fd2, &c, 1) != 1)
19.              {
20.                  break;
21.              }
22.              write(1, &c, 1);
23.          }
24.
25.          return 0;
26.     }
```

Вывод:

```
aabbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz
```

Анализ программы

В данной программе файл alphabet.txt дважды открывается для чтения с помощью системного вызова open(), при этом создаются две различных структуры struct file, описывающих открытый файл (две разные записи в системной таблице открытых файлов), которые связаны с одним и тем же физическим файлом, что видно из рисунка 2. В данном случае текущие позиции в файле для каждой структуры (поле f_pos) будут изменяться независимо друг от друга. Поэтому чтение с использованием одной структуры не затрагивает текущую позицию в другой структуре, и каждый символ из физического фала будет считан дважды с использованием поочередно первой и второй структуры.

Программа 3

Листинг программы:

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.       FILE* fs1 = fopen("test.txt", "w");
6.       FILE* fs2 = fopen("test.txt", "w");
7.
8.       int flag = 0;
9.
10.          for(char c = 'a'; c <= 'z'; c++)
11.          {
12.               if (!flag)
13.               {
14.                   fprintf(fs1, "%c", c);
15.               }
16.               else
17.               {
18.                   fprintf(fs2, "%c", c);
19.               }
20.               flag = !flag;
21.          }
22.
23.          fclose(fs1);
24.          fclose(fs2);
25.
26.          return 0;
27.     }
```

Вывод:

Анализ программы

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

```
FILE * fopen(const char * filename, const char * mode);
```

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 (fclose(fs1)), поэтому изначально в файл test.txt осуществляется запись буфера первого потока (acegikmoqsuwy), при этом данные записываются с начала файла, затем происходит закрытие второго потока (fclose(fs2)) и, соответственно, запись его буфера в файл test.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.