



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»

Отчёт

по лабораторной работе №5

Название: Буферизованный и небуферизованный ввод-вывод

Дисциплина: Операционные системы

Студент

ИУ7-64Б

(Группа)

(Подпись, дата)

Л.Е.Тартыков

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Н.Ю.Рязанова

(И.О. Фамилия)

Москва, 2022

1 Задание

В лабораторной работе анализируется результат выполнения трех программ. Программы демонстрируют открытие одного и того же файла несколько раз. Реализация, когда файл открывается в одной программе несколько раз выбрана для простоты. Однако, как правило, такая ситуация возможна в системе, когда один и тот же файл несколько раз открывают разные процессы или потоки одного процесса. При выполнении асинхронных процессов такая ситуация является вероятной и ее надо учитывать, чтобы избежать потери данных, получения неверного результата при выводе данных в файл или чтения данных не в той последовательности, в какой предполагалось, и в результате при обработке этих данных получения неверного результата. Каждую из приведенных программ надо выполнить в многопоточном варианте: в программах создается дополнительный поток, а работа с открываемым файлом выполняется в потоках.

1.1 Первая программа

Код первой программы приведен на листинге 1.1

Листинг 1.1 – Код программы 1

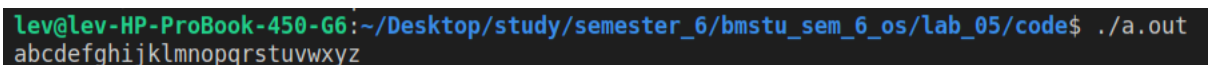
```
1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <pthread.h>
4 #include <unistd.h>
5
6 #define BUFFER_SIZE 20
7 #define FILE_NAME "alphabet.txt"
8
9 void *run(void *arg);
10
11 int main()
12 {
13     pthread_t tid1, tid2;
14     char buffer_1[BUFFER_SIZE], buffer_2[BUFFER_SIZE];
15     int fd = open(FILE_NAME, O_RDONLY);
16
17     FILE *fd1 = fdopen(fd, "r");
18     setvbuf(fd1, buffer_1, _IOFBF, BUFFER_SIZE);
```

```

19
20 FILE *fd2 = fdopen(fd, "r");
21 setvbuf(fd2, buffer_2, _IOFBF, BUFFER_SIZE);
22
23 pthread_create(&tid1, NULL, run, fd1);
24 pthread_create(&tid2, NULL, run, fd2);
25
26 pthread_join(tid1, NULL);
27 pthread_join(tid2, NULL);
28
29 close(fd);
30
31 return 0;
32 }
33
34 void *run(void *arg)
35 {
36     FILE *fs = arg;
37     int flag = 1;
38     char c;
39
40     while (flag == 1) {
41         flag = fscanf(fs, "%c", &c);
42         if (flag == 1)
43             fprintf(stdout, "%c", c);
44     }
45     return NULL;
46 }

```

Результат выполнения программы приведен на рисунке 1.1.



```

lev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$ ./a.out
abcdefghijklmnopqrstuvwxyz

```

Рисунок 1.1 – Результат выполнения программы 1.

На рисунке 1.2 приведена схема связи структур между собой.

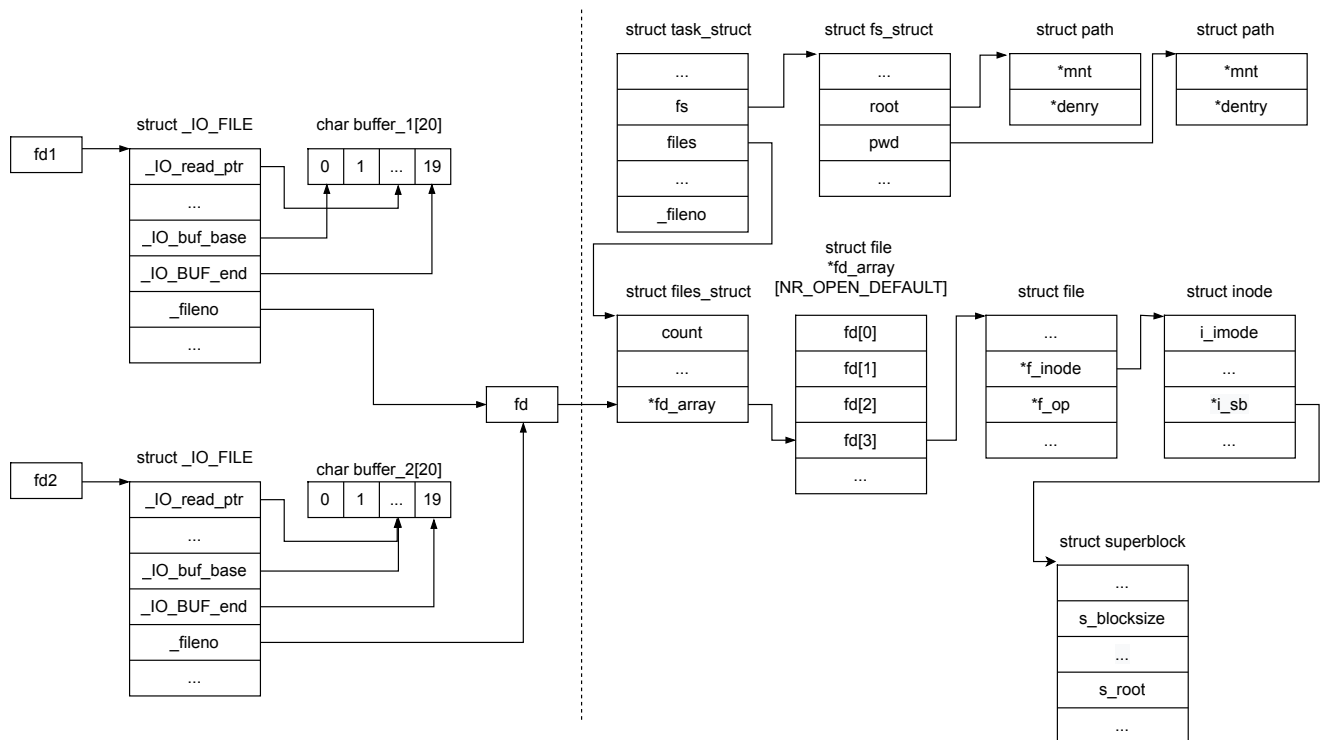


Рисунок 1.2 – Схема связи структур программы 1.

1.2 Вторая программа

Код второй программы приведен на листинге 1.2

Листинг 1.2 – Код программы 2

```

1 #include <unistd.h>
2 #include <fcntl.h>
3 #include <pthread.h>
4 #include <stdio.h>
5
6 #define FILE_NAME "alphabet.txt"
7 pthread_mutex_t lock;
8
9 void *run(void *arg);
10
11 int main()
12 {
13     char c;
14     pthread_t tid1, tid2;
15
16     if (pthread_mutex_init(&lock, NULL) != 0) {
17         perror("failed mutex init");
18         return -1;
19     }

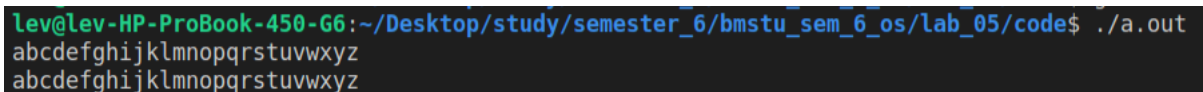
```

```

20
21     int fd1 = open(FILE_NAME, O_RDONLY);
22     int fd2 = open(FILE_NAME, O_RDONLY);
23
24     pthread_create(&tid1, NULL, run, &fd1);
25     pthread_create(&tid2, NULL, run, &fd2);
26
27     pthread_join(tid1, NULL);
28     pthread_join(tid2, NULL);
29     pthread_mutex_destroy(&lock);
30
31     close(fd1);
32     close(fd2);
33
34     return 0;
35 }
36
37 void *run(void *arg)
38 {
39     int fd = *(int*)arg;
40     int flag = 1;
41     char c;
42
43     pthread_mutex_lock(&lock);
44     while (flag == 1){
45         flag = read(fd, &c, 1);
46         if (flag == 1){
47             write(1, &c, 1);
48         }
49     }
50     pthread_mutex_unlock(&lock);
51
52     return 0;
53 }

```

Результат выполнения программы приведен на рисунке 1.3.



```

lev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$ ./a.out
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz

```

Рисунок 1.3 – Результат выполнения программы 2.

На рисунке 1.4 приведена схема связи структур между собой.

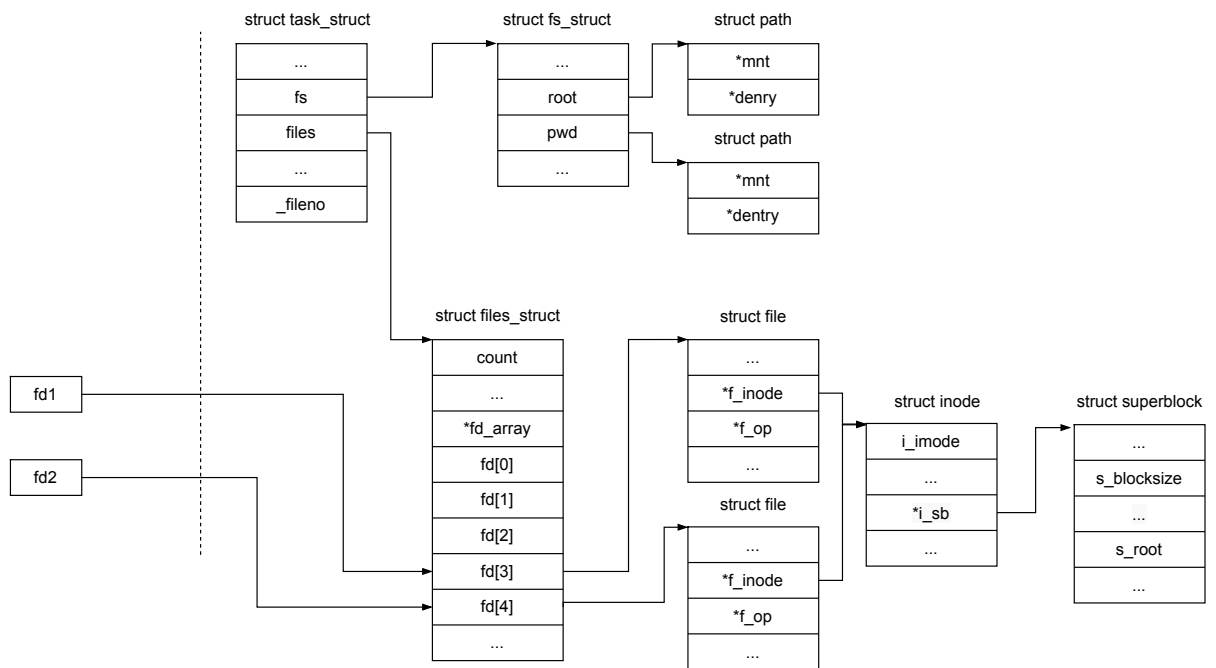


Рисунок 1.4 – Схема связи структур программы 2.

1.3 Третья программа

Код третьей программы приведен на листинге 1.3

Листинг 1.3 – Код программы 3

```

1 #include <fcntl.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <pthread.h>
5 #include <sys/stat.h>
6
7 #define FILE_NAME "alphabet_2.txt"
8
9 void *run_1(void *arg);
10
11 int main()
12 {
13     struct stat stat_buf_open, stat_buf_close;
14     pthread_t tid1;
15     FILE *fd1 = fopen(FILE_NAME, "w");
16     FILE *fd2 = fopen(FILE_NAME, "w");
17
18     lstat(FILE_NAME, &stat_buf_open);
19     printf("open_file: inode=%lu, size=%lu\n", stat_buf_open.st_ino,
20           stat_buf_open.st_size);

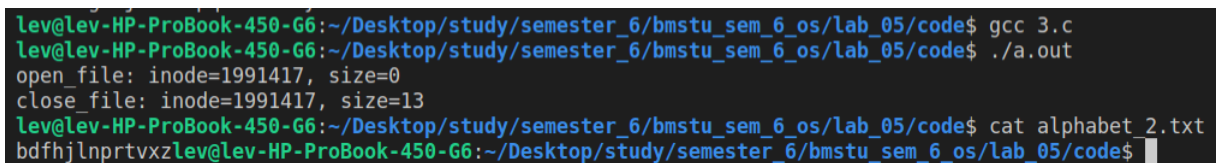
```

```

21 pthread_create(&tid1, NULL, run_1, fd1);
22
23 for (char c = 'b'; c <= 'z'; c += 2){
24     fprintf(fd2, "%c", c);
25 }
26
27 pthread_join(tid1, NULL);
28
29 fclose(fd1);
30 fclose(fd2);
31
32 lstat(FILE_NAME, &stat_buf_close);
33 printf("close_file: inode=%lu, size=%lu\n", stat_buf_close.st_ino,
34        stat_buf_close.st_size);
35
36 return 0;
37 }
38
39 void *run_1(void *arg)
40 {
41     FILE *fd1 = arg;
42     for (char c = 'a'; c <= 'z'; c += 2){
43         fprintf(fd1, "%c", c);
44     }
45
46     return 0;
47 }

```

Результат выполнения программы приведен на рисунке 1.5.



```

lev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$ gcc 3.c
lev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$ ./a.out
open_file: inode=1991417, size=0
close_file: inode=1991417, size=13
lev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$ cat alphabet_2.txt
bdfhjlnprtvxzlev@lev-HP-ProBook-450-G6:~/Desktop/study/semester_6/bmstu_sem_6_os/lab_05/code$

```

Рисунок 1.5 – Результат выполнения программы 3.

На рисунке 1.6 приведена схема связи структур между собой.

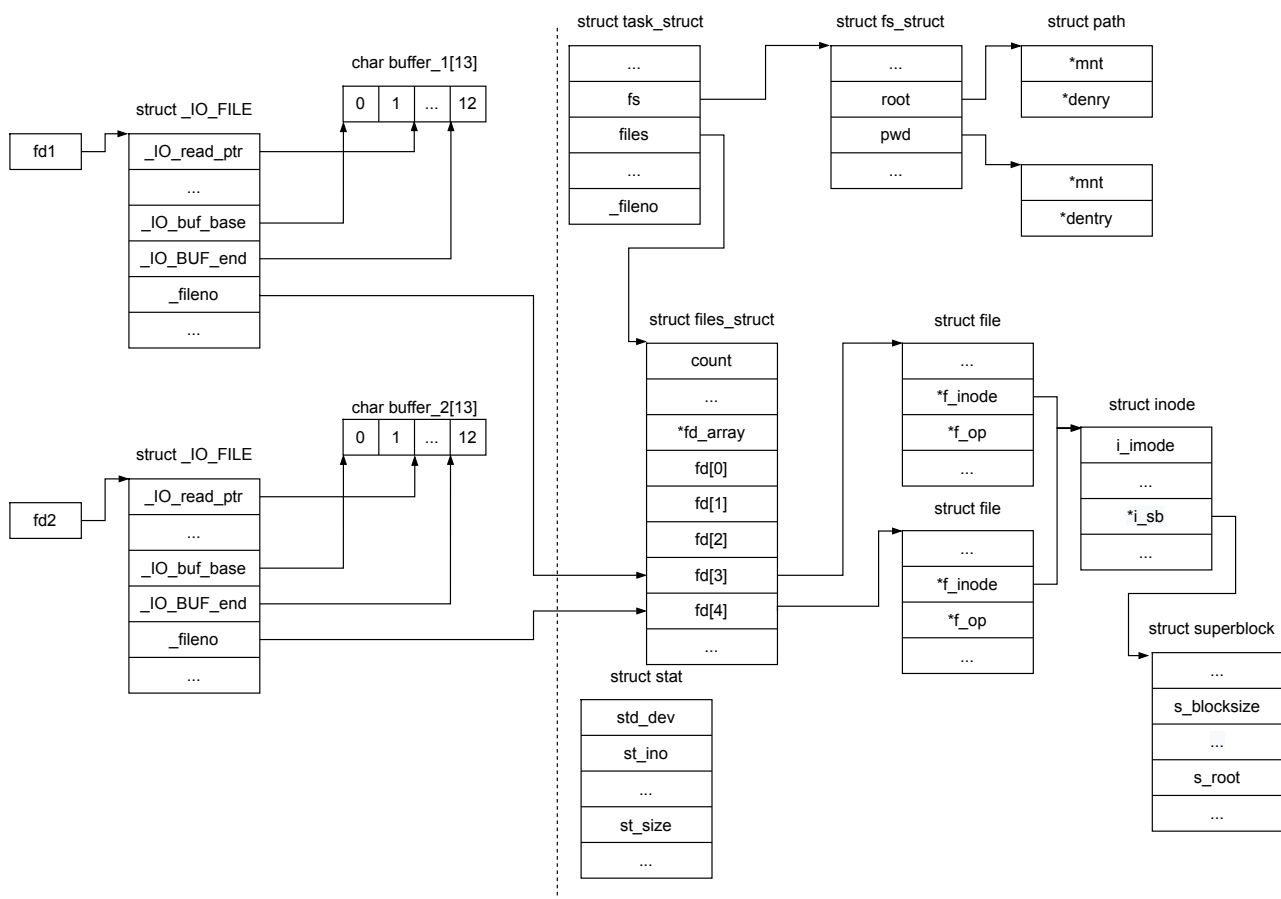


Рисунок 1.6 – Схема связи структур программы 3.

1.4 Анализ

При открытии файла с помощью системного вызова `open()` в режиме добавления (`O_APPEND`) (или `fopen(path, "a")`) перед каждым вызовом `write()` смещение в файле устанавливается в конец этого файла, как, например, при выполнении вызова `lseek(2)`.

Системный вызов `open()` назначает свободный файловый дескриптор для открытого файла в системной таблице открытых файлов.

При выполнении записи в файл сначала информация записывается в буфер, затем выгружается в файл. При чтении из файла информация также записывает сначала в буфер. Содержание буфера выгружается в файл в трех случаях: когда буфер переполнен, при вызове `fclose()` и `fflush()`.

Одновременный доступ разных процессов к одному и тому же файлу может быть источником проблем – `race condition`.