



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»

ОТЧЕТ

по лабораторной работе №7

по курсу «Операционные системы»

на тему: «Буферизированный и небуферизированный ввод-вывод»

Студент ИУ7-62Б
(Группа)

(Подпись, дата)

Готов И. А.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Рязанова Н. Ю.
(И. О. Фамилия)

2023 г.

1 Структура FILE

Листинг 1.1 – Структура FILE

```
1 typedef struct _IO_FILE FILE;
2 struct _IO_FILE
3 {
4     int _flags;      /* High-order word is _IO_MAGIC; rest is flags. */
5
6     /* The following pointers correspond to the C++ streambuf protocol. */
7     char *_IO_read_ptr; /* Current read pointer */
8     char *_IO_read_end; /* End of get area. */
9     char *_IO_read_base; /* Start of putback+get area. */
10    char *_IO_write_base; /* Start of put area. */
11    char *_IO_write_ptr; /* Current put pointer. */
12    char *_IO_write_end; /* End of put area. */
13    char *_IO_buf_base; /* Start of reserve area. */
14    char *_IO_buf_end; /* End of reserve area. */
15
16    /* The following fields are used to support backing up and undo. */
17    char *_IO_save_base; /* Pointer to start of non-current get area. */
18    char *_IO_backup_base; /* Pointer to first valid character of backup area */
19    char *_IO_save_end; /* Pointer to end of non-current get area. */
20
21    struct _IO_marker *_markers;
22
23    struct _IO_FILE *_chain;
24
25    int _fileno;
26    int _flags2;
27    __off_t _old_offset; /* This used to be _offset but it's too small. */
28
29    /* 1+column number of pbase(); 0 is unknown. */
30    unsigned short _cur_column;
31    signed char _vtable_offset;
32    char _shortbuf[1];
33
34    _IO_lock_t *_lock;
35    #ifdef _IO_USE_OLD_IO_FILE
36 };
```

2 Программа №1

Листинг 2.1 – Программа №1

```
1 #include <fcntl.h>
2 #include <stdio.h>
3
4 int main() {
5     // have kernel open connection to file alphabet.txt
6     int fd = open("alphabet.txt", O_RDONLY);
7
8     // create two a C I/O buffered streams using the above connection
9     FILE *fs1 = fdopen(fd, "r");
10    char buff1[20];
11    setvbuf(fs1, buff1, _IOFBF, 20);
12
13    FILE *fs2 = fdopen(fd, "r");
14    char buff2[20];
15    setvbuf(fs2, buff2, _IOFBF, 20);
16
17    // read a char & write it alternately from fs1 and fs2
18    int flag1 = 1, flag2 = 2;
19    while (flag1 == 1 || flag2 == 1) {
20        char c;
21
22        flag1 = fscanf(fs1, "%c", &c);
23        if (flag1 == 1) fprintf(stdout, "%c", c);
24
25        flag2 = fscanf(fs2, "%c", &c);
26        if (flag2 == 1) fprintf(stdout, "%c", c);
27    }
28
29    return 0;
30 }
```

Результат работы: aubvcwdxeyfzghijklmnopqrst

Листинг 2.2 – Программа №1 (многопоточная)

```
1 #include <fcntl.h>
2 #include <pthread.h>
3 #include <stdio.h>
4 #define BUF_SIZE 20
5 #define FILENAME "alphabet.txt"
6
7 void *t1_run(void *args) {
8     int *fd = (int *)args;
```

```

9 FILE *fs1 = fdopen(*fd, "r");
10 char buff1[BUF_SIZE];
11 setvbuf(fs1, buff1, _IOFBF, BUF_SIZE);
12 int flag = 1;
13 char c;
14 while ((flag = fscanf(fs1, "%c", &c)) == 1)
15     fprintf(stdout, "%c", c);
16 return NULL;
17 }
18
19 void *t2_run(void *args) {
20     int *fd = (int *)args;
21     FILE *fs2 = fdopen(*fd, "r");
22     char buff2[BUF_SIZE];
23     setvbuf(fs2, buff2, _IOFBF, BUF_SIZE);
24     int flag = 1;
25     char c;
26     while ((flag = fscanf(fs2, "%c", &c)) == 1)
27         fprintf(stdout, "%c", c);
28     return NULL;
29 }
30
31 int main() {
32     pthread_t t1, t2;
33     int fd = open(FILENAME, O_RDONLY);
34
35     pthread_create(&t1, NULL, t1_run, &fd);
36     pthread_create(&t2, NULL, t2_run, &fd);
37
38     pthread_join(t1, NULL);
39     pthread_join(t2, NULL);
40     return 0;
41 }

```

Результаты работы (несколько запусков программы):

- abcdefghijklmnopqrstuvwxyz
- abuvwxyzcdefghijklmnopqrst
- aubvcwdxeyfzghijklmnopqrst

В программе файл открывается *один* раз системным вызовом `open`. Системный вызов `open` возвращает дескриптор файла типа `int`. Возвращенный

номер `fd` – индекс дескриптора открытого файла в таблице дескрипторов файлов процесса.

Функция `fdopen()` создаёт указатель на структуру `FILE`. Поле `_fileno` в `struct _IO_FILE` содержит номер дескриптора, который вернула функция `open()`.

Функция `setvbuf()` явно задает размер буфера в 20 байт и меняет тип буферизации (для `fs1` и `fs2`) на полный.

При первом вызове функции `fscanf()` в цикле (для `fs1`) `buff1` будет заполнен полностью – первыми 20 символами (буквами алфавита). `f_pos` в структуре `struct_file` открытого файла увеличится на 20.

При втором вызове `fscanf()` в цикле (для `fs2`) буффер `buff2` будет заполнен оставшимися 6 символами (начиная с `f_pos`).

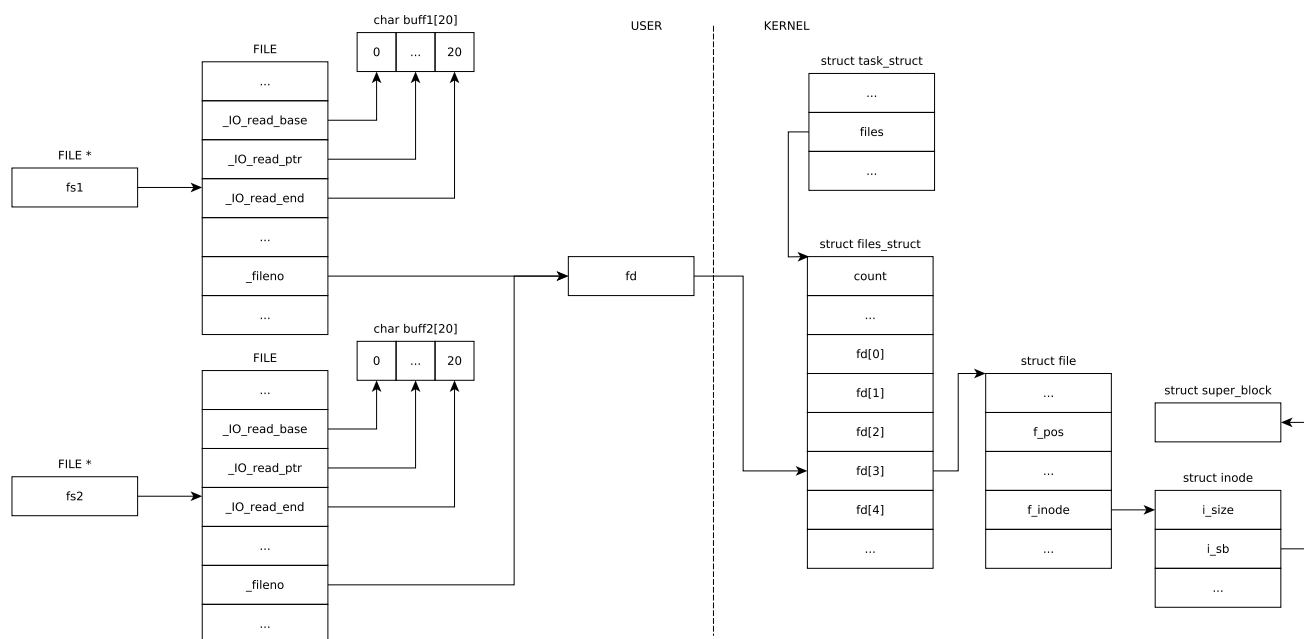


Рисунок 2.1 – Используемые структуры

2.1 Программа №2

Листинг 2.3 – Программа №2

```

1 #include <fcntl.h>
2 #include <unistd.h>

```

```

3 int main() {
4     int fd1 = open("alphabet.txt", O_RDONLY);
5     int fd2 = open("alphabet.txt", O_RDONLY);
6     char c;
7     while (read(fd1, &c, 1) == 1 && read(fd2, &c, 1) == 1) {
8         write(1, &c, 1);
9         write(1, &c, 1);
10    }
11    return 0;
12 }

```

Результат работы: aabbccddeeffgghhiijjkkllmmnnnooppqrrssttuuvvwwxxyyzz

Листинг 2.4 – Программа №2 (многопоточная)

```

1  #include <fcntl.h>
2  #include <pthread.h>
3  #include <stdio.h>
4  #define BUF_SIZE 20
5  #define FILENAME "alphabet.txt"
6
7  void *t1_run(void *args) {
8      int *fd = (int *)args;
9      FILE *fs1 = fdopen(*fd, "r");
10     char buff1[BUF_SIZE];
11     setvbuf(fs1, buff1, _IOFBF, BUF_SIZE);
12     int flag = 1;
13     char c;
14     while ((flag = fscanf(fs1, "%c", &c)) == 1)
15         fprintf(stdout, "%c", c);
16     return NULL;
17 }
18
19 void *t2_run(void *args) {
20     int *fd = (int *)args;
21     FILE *fs2 = fdopen(*fd, "r");
22     char buff2[BUF_SIZE];
23     setvbuf(fs2, buff2, _IOFBF, BUF_SIZE);
24     int flag = 1;
25     char c;
26     while ((flag = fscanf(fs2, "%c", &c)) == 1)
27         fprintf(stdout, "%c", c);
28     return NULL;
29 }
30
31 int main() {

```

```

32     pthread_t t1, t2;
33     int fd = open(FILENAME, O_RDONLY);
34
35     pthread_create(&t1, NULL, t1_run, &fd);
36     pthread_create(&t2, NULL, t2_run, &fd);
37
38     pthread_join(t1, NULL);
39     pthread_join(t2, NULL);
40     return 0;
41 }

```

Результаты работы (несколько запусков программы):

- aabcbefcfdheifjgkhlmjnkolpmqnrosptqurvswtxuyvzwxxyz
- aabbcdcedfegghhijjklmmnnnooppqrrssttuuvvwvwxxyzz
- aabbccddeeffgghihjijjklmmnnopqprqsrtstutvuwvwxwyxzyz

Функция `open()` создает файловый дескриптор, два раза для одного и того же файла, поэтому в программе существует две различные `struct file`, но ссылающиеся на один и тот же `struct inode`.

Из-за того что структуры разные (у каждой структуры свое поле `f_pos`), посимвольная печать дважды выведет содержимое файла в формате «aabbcc...» (в случае однопоточной реализации).

В случае многопоточной реализации, потоки выполняются с разной скоростью и символы перемешаются.

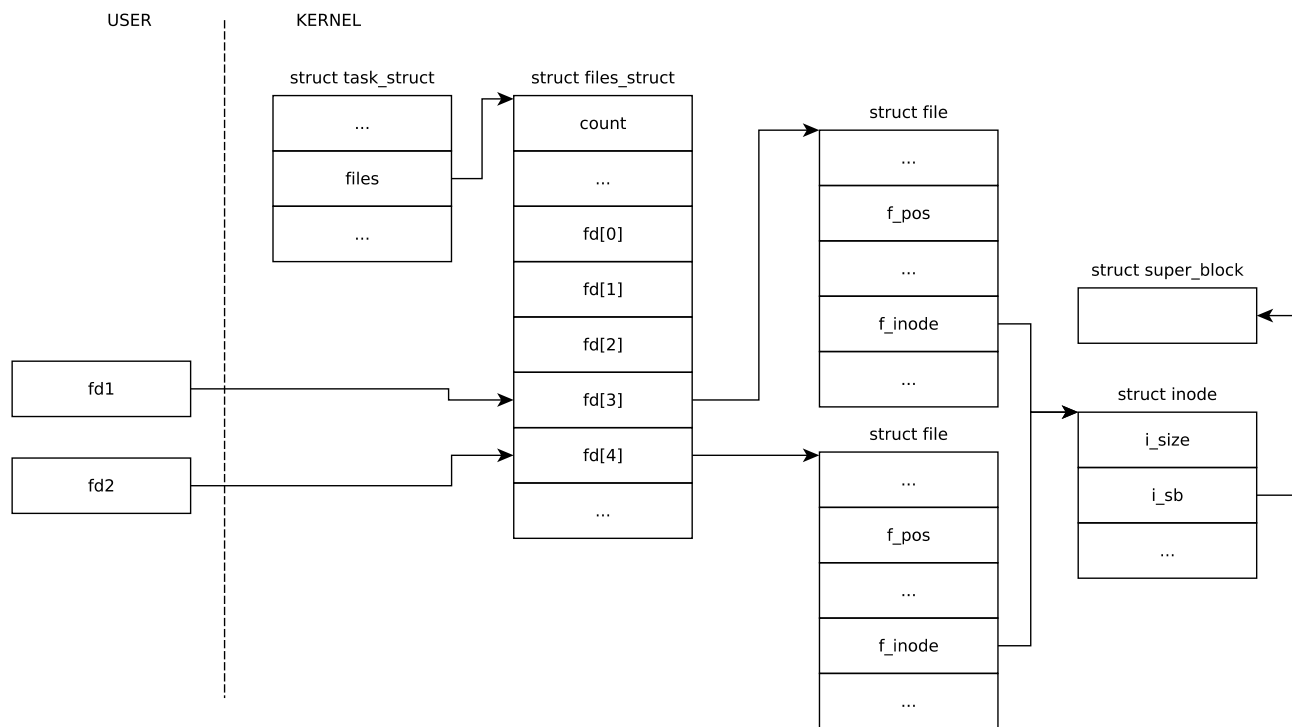


Рисунок 2.2 – Используемые структуры

2.2 Программа №3

Листинг 2.5 – Программа №3

```

1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 #define FILENAME "out.txt"
6
7 int main() {
8     FILE *f1 = fopen(FILENAME, "w");
9     FILE *f2 = fopen(FILENAME, "w");
10
11     for (char c = 'a'; c <= 'z'; c++) {
12         if (c % 2) {
13             fprintf(f1, "%c", c);
14         } else {
15             fprintf(f2, "%c", c);
16         }
17     }
18 }
```



```

18
19     fclose(f1);
20     fclose(f2);
21
22     return 0;
23 }

```

Результат работы: bdfhjlnprtvxz

Листинг 2.6 – Программа №3 (многопоточная)

```

1 #include <fcntl.h>
2 #include <pthread.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 #define FILENAME "out.txt"
7
8 void *t1_run(void *args) {
9     FILE *f = fopen(FILENAME, "w");
10
11     for (char c = 'a'; c <= 'z'; c += 2) {
12         fprintf(f, "%c", c);
13     }
14
15     fclose(f);
16
17     return NULL;
18 }
19
20 void *t2_run(void *args) {
21     FILE *f = fopen(FILENAME, "w");
22
23     for (char c = 'b'; c <= 'z'; c += 2) {
24         fprintf(f, "%c", c);
25     }
26
27     fclose(f);
28
29     return NULL;
30 }
31
32 int main() {
33     pthread_t t1, t2;
34     pthread_create(&t1, NULL, t1_run, NULL);
35     pthread_create(&t2, NULL, t2_run, NULL);

```

```

36
37     pthread_join(t1, NULL);
38     pthread_join(t2, NULL);
39
40     return 0;
41 }

```

Результаты работы (несколько запусков программы):

- acegikmoqsuwy
- bdfhjlnprtvxz

Файл открывается на запись два раза функцией `fopen()`.

Из-за того `f_pos` независимы для каждого дескриптора файла, запись в файл будет производиться с нулевой позиции.

Функция `fprintf()` предоставляет буферизованный вывод.

Изначально информация пишется в буфер, а из буфера в файл если произошло одно из событий:

1. буфер полон
2. вызвана функция `fclose()`
3. вызвана функция `fflush()`

В случае нашей программы, информация в файл запишется в результате вызова функции `fclose()`. Запись в файл происходит в результате вызова функции `fclose`. При вызове `fclose()` для `fs1` буфер для `fs1` записывается в файл. При вызове `fclose()` для `fs2`, все содержимое файла очищается, а в файл записывается содержимое буфера для `fs2`. В итоге произошла потеря данных, в файле окажется только содержимое буфера для `fs2`. Чтобы этого избежать, необходимо использовать `open()` с флагом `O_APPEND`. Если этот флаг установлен, то каждой операции добавления гарантируется неделимость.

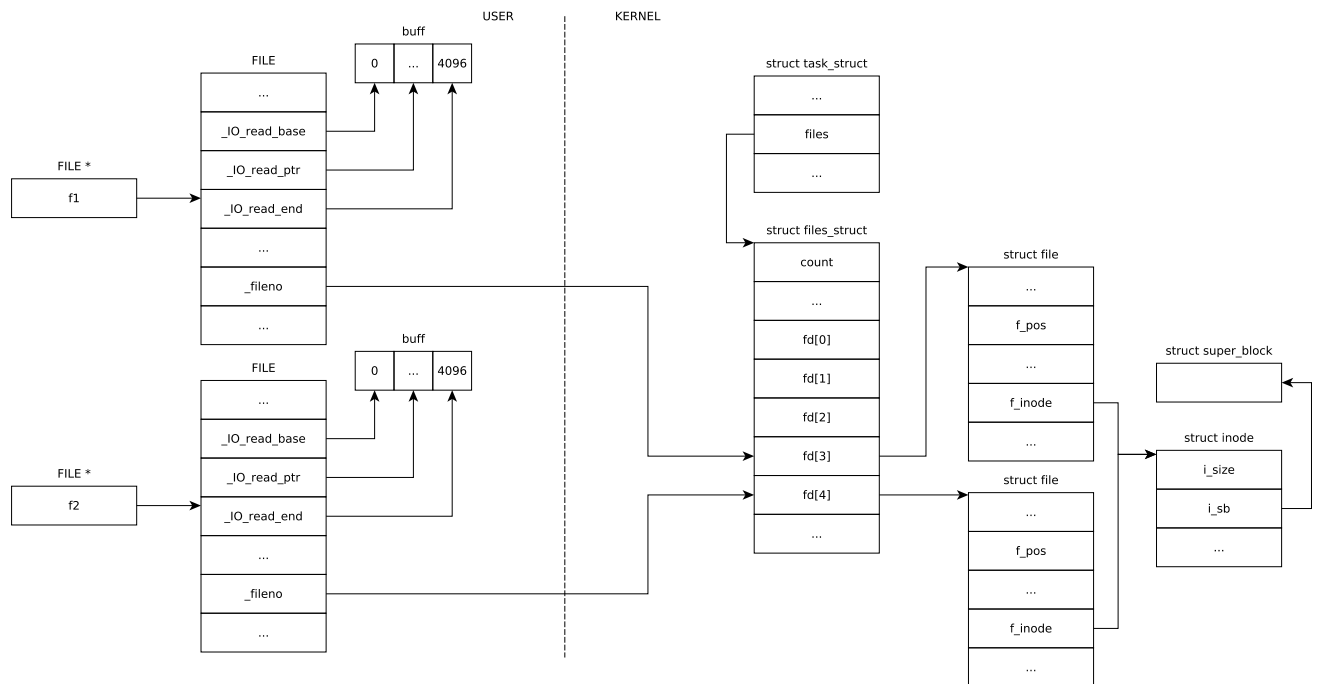


Рисунок 2.3 – Используемые структуры

2.3 Программа №3 (с open, read, write)

Листинг 2.7 – Программа №3 (с open, read, write)

```

1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 #define FILENAME "out.txt"
6
7 int main() {
8     int f1 = open(FILENAME, O_WRONLY | O_CREAT);
9     int f2 = open(FILENAME, O_WRONLY | O_CREAT);
10
11     for (char c = 'a'; c <= 'z'; c++) {
12         if (c % 2) {
13             write(f1, &c, 1);
14         } else {
15             write(f2, &c, 1);
16         }
17     }
18
19     close(f1);

```

```

20     close(f2);
21     return 0;
22 }

```

Результат работы: bdfhjlnprtvxz

Листинг 2.8 – Программа №3 (с open, read, write) (многопоточная)

```

1  #include <fcntl.h>
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <unistd.h>
5
6  #define FILENAME "out.txt"
7
8  void *t1_run(void *args) {
9      int f = open(FILENAME, O_WRONLY | O_CREAT);
10
11     for (char c = 'a'; c <= 'z'; c += 2) {
12         write(f, &c, 1);
13     }
14
15     close(f);
16
17     return NULL;
18 }
19
20 void *t2_run(void *args) {
21     int f = open(FILENAME, O_WRONLY | O_CREAT);
22
23     for (char c = 'b'; c <= 'z'; c += 2) {
24         write(f, &c, 1);
25     }
26
27     close(f);
28
29     return NULL;
30 }
31
32 int main() {
33     pthread_t t1, t2;
34     pthread_create(&t1, NULL, t1_run, NULL);
35     pthread_create(&t2, NULL, t2_run, NULL);
36     pthread_join(t1, NULL);
37     pthread_join(t2, NULL);
38     return 0;

```

Результаты работы (несколько запусков программы):

- acegikmoqsuwy
- bdfhjlnprtvxz