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
по дисциплине «Операционные системы»

Тема: Буферизованный и небуферизованный ввод-вывод
Студент: Карпова Е. О.
Группа: ИУ7-62Б
Оценка (баллы): _____
Преподаватель: Рязанова Н. Ю.

Москва — 2023 г.

1. Структуры

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  };
```

Листинг 1..2: Структура stat

```
1   struct stat
2   {
3     /* These are the members that POSIX.1 requires.  */
4
5     __mode_t st_mode;    /* File mode.  */
6 #ifndef __USE_FILE_OFFSET64
7     __ino_t st_ino;    /* File serial number.  */
8 #else
9     __ino64_t st_ino;    /* File serial number.  */
10 #endif
11    __dev_t st_dev;    /* Device containing the file.  */
12    __nlink_t st_nlink;    /* Link count.  */
13
14    __uid_t st_uid;    /* User ID of the file's owner.  */
15    __gid_t st_gid;    /* Group ID of the file's group.  */
16 #ifndef __USE_FILE_OFFSET64
17    __off_t st_size;     /* Size of file, in bytes.  */
18 #else
19    __off64_t st_size;     /* Size of file, in bytes.  */
20 #endif
21
22    __time_t st_atime;     /* Time of last access.  */
23    __time_t st_mtime;     /* Time of last modification.  */
24    __time_t st_ctime;     /* Time of last status change.  */
25
26    /* This should be defined if there is a `st_blksize' member.  */
27 #undef  _STATBUF_ST_BLKSIZE
28  };
```

2. Программа 1

2.1. Однопоточная реализация

Листинг 2..1: Программа №1

```
1  #include <stdio.h>
2  #include <fcntl.h>
3
4  #define BUF_SIZE 20
5  #define FILENAME "alphabet.txt"
6
7  int main()
8  {
9    int fd = open(FILENAME, O_RDONLY);
10
11   FILE* fs1 = fdopen(fd, "r");
12   char buff1[BUF_SIZE];
13   setvbuf(fs1, buff1, _IOFBF, BUF_SIZE);
14
15   FILE* fs2 = fdopen(fd, "r");
16   char buff2[BUF_SIZE];
17   setvbuf(fs2, buff2, _IOFBF, BUF_SIZE);
18
19   int flag1 = 1, flag2 = 2;
20   while(flag1 == 1 || flag2 == 1)
21   {
22     char c;
23
24     flag1 = fscanf(fs1, "%c", &c);
25     if (flag1 == 1)
26     {
27       fprintf(stdout, "%c", c);
28     }
29
30     flag2 = fscanf(fs2, "%c", &c);
```

```
31      if (flag2 == 1)
32      {
33        fprintf(stdout, "%c", c);
34      }
35    }
36
37    return 0;
38  }
```

Вывод программы

```
1   aubvcwdxeyfzghijklmnopqrst
```

2.2. Многопоточная реализация

```
1   #include <fcntl.h>
2   #include <pthread.h>
3   #include <stdio.h>
4
5   #define BUF_SIZE 20
6   #define FILENAME "alphabet.txt"
7
8   void* thread1(void *args)
9   {
10    int* fd = (int*)args;
11    FILE* fs1 = fdopen(*fd, "r");
12    char buff1[BUF_SIZE];
13    setvbuf (fs1, buff1, _IOFBF, BUF_SIZE);
14
15    int flag = 1;
16    char c;
17    while ((flag = fscanf(fs1, "%c", &c)) == 1)
18    {
19      fprintf(stdout, "%c", c);
20    }
21
22    return NULL;
23  }
```

```
24
25 void* thread2(void* args)
26 {
27   int* fd = (int*)args;
28   FILE* fs2 = fdopen(*fd, "r");
29   char buff2[BUF_SIZE];
30   setvbuf ( fs2, buff2, _IOFBF, BUF_SIZE);
31
32   int flag = 1;
33   char c;
34   while ((flag = fscanf(fs2, "%c", &c)) == 1)
35   {
36     fprintf(stdout, "%c", c);
37   }
38
39   return NULL;
40 }
41
42 int main()
43 {
44   pthread_t t1, t2 ;
45
46   int fd = open(FILENAME, O_RDONLY);
47
48   pthread_create(&t1, NULL, thread1, &fd);
49   pthread_create(&t2, NULL, thread2, &fd);
50
51   pthread_join(t1, NULL);
52   pthread_join(t2, NULL);
53
54   return 0 ;
55 }
```

Вывод программы на разных запусках

```
1  abcdefghijklmnopqrstuvwxyz
2  abcdefuvwxyzghijklmnopqrst
3  abcdeuvwxyzfghijklmnopqrst
```

2.3. Пояснение

В программе файл открывается 1 раз системным вызовом `open()`, который возвращает дескриптор файла типа int в usermode. Возвращенный номер fd — индекс дескриптора открытого файла в таблице дискрипторов открытых файлов процесса.

Затем 2 раза вызывается функция `fdopen()` библиотеки stdio, которая создаёт указатель на структуру `FILE`, определенную с помощью define на базе `struct _IO_FILE`.

Функции `fdopen()` нужно передать возвращённый из `open()` дескриптор fd. Поле _fileno в `struct _IO_FILE` содержит номер этого дескриптора.

Функция `setvbuf()` устанавливает размер буфера 20 байт.

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

При втором вызове `fscanf()` в цикле (для fs2) буфер buff2 будет заполнен оставшимися 6 символами (начиная с f_pos, изменённого после 1-ого вызова `fscanf()`).

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

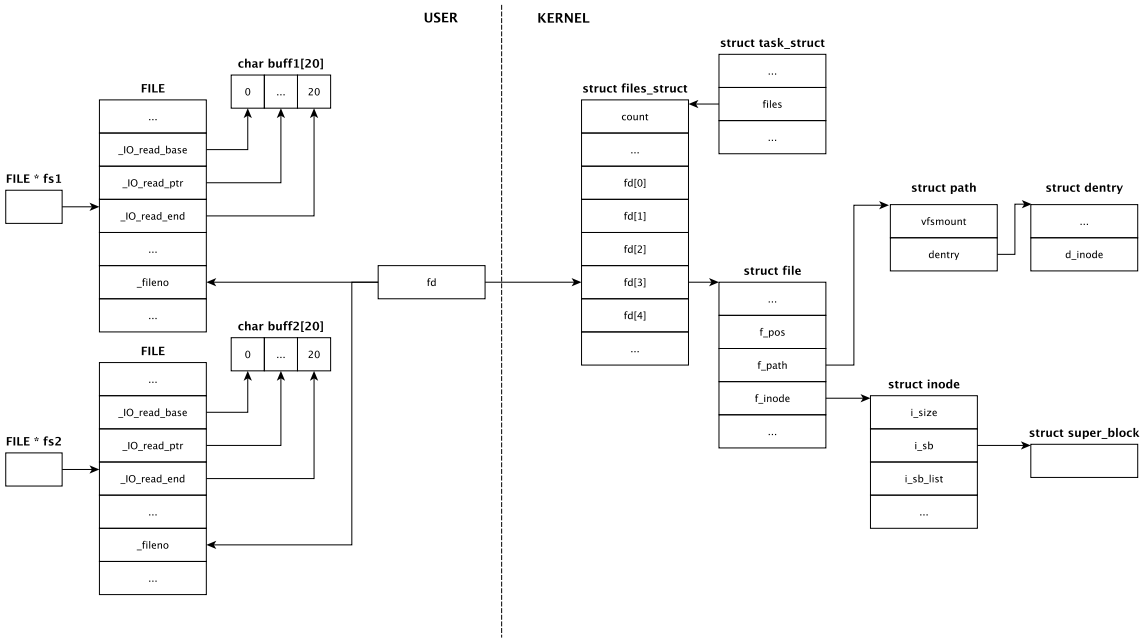

Рисунок 2.1 — Используемые структуры

3. Программа 2

3.1. Однопоточная реализация

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

Вывод программы

```
1   aabbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz
```

3.2. Многопоточная реализация

```
1   #include <fcntl.h>
2  #include <pthread.h>
3  #include <unistd.h>
4  #include <stdio.h>
5
6  #define BUF_SIZE 20
7  #define FILENAME "alphabet.txt"
```

```
8
9  void* thread1()
10 {
11   int fd = open(FILENAME, O_RDONLY);
12
13   char c;
14   while ((read(fd, &c, 1)) == 1) write(1, &c, 1);
15
16   return NULL;
17 }
18
19 void* thread2()
20 {
21   int fd = open(FILENAME, O_RDONLY);
22
23   char c;
24   while ((read(fd, &c, 1)) == 1) write(1, &c, 1);
25
26   return NULL;
27 }
28
29 int main()
30 {
31   pthread_t t1, t2;
32
33   pthread_create(&t1, NULL, thread1, NULL);
34   pthread_create(&t2, NULL, thread2, NULL);
35
36   pthread_join(t1, NULL);
37   pthread_join(t2, NULL);
38
39   return 0;
40 }
```

Вывод программы на разных запусках

```
1  aabbcdcedfegfhgihjikjlkmlnmonpoqprqsrtsutvuwvxwyxzyz
2  aabcbdcedfegfhgihjikjlklmmnnooppqqrrssttuuvvwwxxyyzz
3  aabbccddeeffgghhijklmnoipjqkrlsmtnuovpwqxrysztuvwxyz
```

3.3. Пояснение

Функция `open()` два раза создает дескриптор для одного и того же файла в системной таблице открытых файлов, поэтому в программе существует два различных дескриптора открытого файла (`struct file`), ссылающихся на один и тот же `struct inode`.

Так как у каждой структуры `struct file` свое поле `f_pos`, выводимые символы будут дублироваться.

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

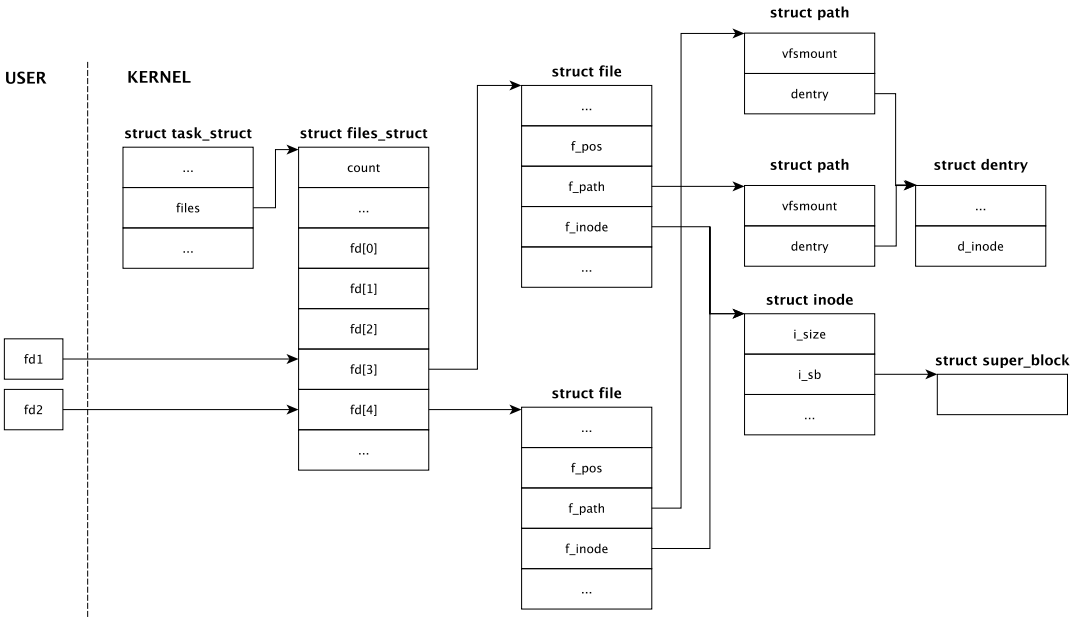

Рисунок 3.1 — Используемые структуры

4. Программа 3

4.1. fopen()

4.2. Однопоточная реализация

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <unistd.h>
6
7  #define FILENAME "tmp1.txt"
8
9  int main()
10 {
11   struct stat buf;
12
13   FILE* f1 = fopen(FILENAME, "w");
14
15   int rc = fstat(f1->_file, &buf);
16   if (!rc)
17   {
18     fprintf(stdout, "after fopen f1: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
19   }
20
21   FILE* f2 = fopen(FILENAME, "w");
22
23   rc = fstat(f2->_file, &buf);
24   if (!rc)
25   {
26     fprintf(stdout, "after fopen f2: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
27   }
```

```
28
29  for (char c = 'a'; c <= 'z'; c++)
30  {
31    if (c % 2)
32    {
33      fprintf(f1, "%c", c);
34    }
35    else
36    {
37      fprintf(f2, "%c", c);
38    }
39  }
40
41  rc = fstat(f1->_file, &buf);
42  if (!rc)
43  {
44    fprintf(stdout, "before fclose f1: inode - %ju, total size - %lld\n",
        (uintmax_t)buf.st_ino, buf.st_size);
45  }
46
47  rc = fstat(f2->_file, &buf);
48  if (!rc)
49  {
50    fprintf(stdout, "before fclose f2: inode - %ju, total size - %lld\n",
        (uintmax_t)buf.st_ino, buf.st_size);
51  }
52
53  fclose(f1);
54
55  rc = stat(FILENAME, &buf);
56  fprintf(stdout, "after fclose f1: inode - %ju, total size - %lld\n", (
      uintmax_t)buf.st_ino, buf.st_size);
57
58  fclose(f2);
59
60  rc = stat(FILENAME, &buf);
61  fprintf(stdout, "after fclose f2: inode - %ju, total size - %lld\n", (
      uintmax_t)buf.st_ino, buf.st_size);
```

```
62
63    return 0;
64  }
```

Вывод программы

```
1   bdfhjlnprtvxz
```

Вывод stat

```
1  after open f1: inode - 17875612, total size - 0
2  after open f2: inode - 17875612, total size - 0
3  before fclose f1: inode - 17875612, total size - 0
4  before fclose f2: inode - 17875612, total size - 0
5  after fclose f1: inode - 17875612, total size - 13
6  after fclose f2: inode - 17875612, total size - 13
```

4.3. Многопоточная реализация

```
1   #include <fcntl.h>
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <unistd.h>
5
6  #define FILENAME "tmp2.txt"
7
8  void* thread1()
9  {
10    FILE* f = fopen(FILENAME, "w");
11
12    for (char c = 'a'; c <= 'z'; c += 2)
13    {
14      fprintf(f, "%c", c);
15    }
16
17    fclose(f);
18
19    return NULL;
20  }
```

```
21
22 void* thread2()
23 {
24   FILE* f = fopen(FILENAME, "w");
25
26   for (char c = 'b'; c <= 'z'; c += 2)
27   {
28     fprintf(f, "%c", c);
29   }
30
31   fclose(f);
32
33   return NULL;
34 }
35
36 int main()
37 {
38   pthread_t t1, t2;
39
40   pthread_create(&t1, NULL, thread1, NULL);
41   pthread_create(&t2, NULL, thread2, NULL);
42
43   pthread_join(t1, NULL);
44   pthread_join(t2, NULL);
45
46   return 0;
47 }
```

Вывод программы на разных запусках

```
1 acegikmoqsuwy
2 bdfhjlnprtvxz
```

4.4. Пояснение

Файл открывается на запись два раза функцией `fopen()`, которая внутри своего кода вызывает `open()`. Каждое открытие файла сопровождается созданием дескриптора открытого файла `struct file`, содержащего поле f_pos.

Поля `f_pos` двух разных дескрипторов одного открытого файла независимы, поэтому запись в файл будет производится с нулевой позиции.

Функция `fprintf()` предоставляет буферизованный вывод, поэтому сначала информация пишется в буфер, а из буфера в файл при одном из условий:

1. буфер заполнен;
2. вызвана функция `fclose()`;
3. вызвана функция `fflush()` (принудительная запись).

При вызове `fclose()` для fs1 буфер для fs1 записывается в файл. При вызове `fclose()` для fs2, все содержимое файла затирается, а в файл записывается содержимое буфера для fs2. В итоге произошла потеря данных, в файле окажется только содержимое буфера для fs2.

Чтобы этого избегать, необходимо использовать флаг O_APPEND. Если этот флаг установлен, то первая запись в файл не теряется.

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

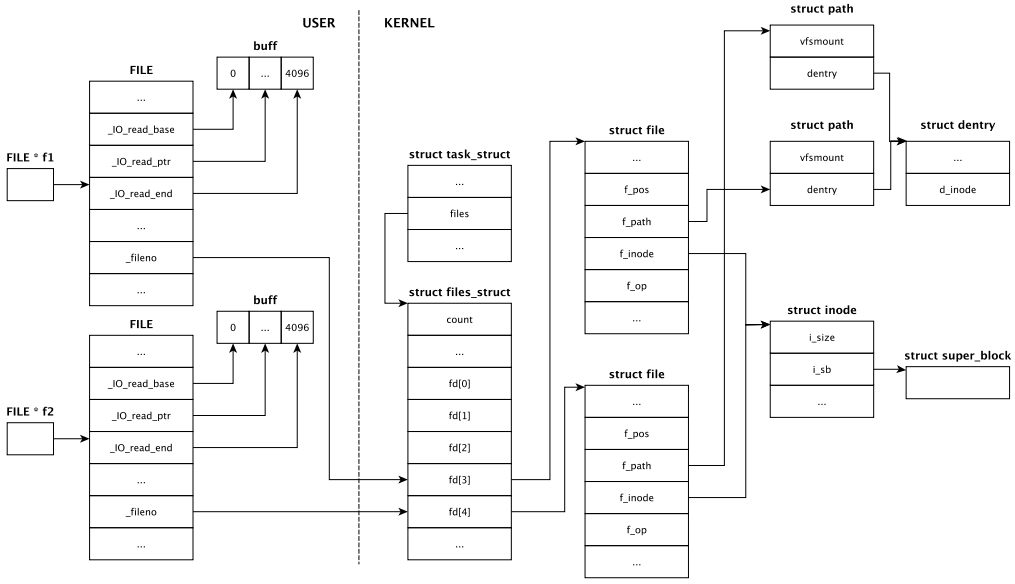

Рисунок 4.1 — Используемые структуры

4.5. open()

4.6. Однопоточная реализация

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <unistd.h>
6
7  #define FILENAME "tmp3.txt"
8
9  int main()
10 {
11   struct stat buf;
12
13   int f1 = open(FILENAME, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);
14
15   int rc = fstat(f1, &buf);
16   if (!rc)
17   {
18     fprintf(stdout, "after open f1: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
19   }
20
21   int f2 = open(FILENAME, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);
22
23   rc = fstat(f2, &buf);
24   if (!rc)
25   {
26     fprintf(stdout, "after open f2: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
27   }
28
29   for (char c = 'a'; c <= 'z'; c++)
30   {
31     if (c % 2)
32     {
```

```
33          write(f1, &c, 1);
34        }
35        else
36        {
37          write(f2, &c, 1);
38        }
39    }
40
41    rc = fstat(f1, &buf);
42    if (!rc)
43    {
44      fprintf(stdout, "before close f1: inode - %ju, total size - %lld\n", (
            uintmax_t)buf.st_ino, buf.st_size);
45    }
46    rc = fstat(f2, &buf);
47    if (!rc)
48    {
49      fprintf(stdout, "before close f2: inode - %ju, total size - %lld\n", (
            uintmax_t)buf.st_ino, buf.st_size);
50    }
51
52    close(f1);
53
54    rc = stat(FILENAME, &buf);
55    fprintf(stdout, "after close f1: inode - %ju, total size - %lld\n", (
          uintmax_t)buf.st_ino, buf.st_size);
56
57    close(f2);
58
59    rc = stat(FILENAME, &buf);
60    fprintf(stdout, "after fclose f2: inode - %ju, total size - %lld\n", (
          uintmax_t)buf.st_ino, buf.st_size);
61
62    return 0;
63  }
```

Вывод программы

```
1   bdfhjlnprtvxz
```

Вывод stat

```
after open f1: inode - 17885541, total size - 0
after open f2: inode - 17885541, total size - 0
before close f1: inode - 17885541, total size - 13
before close f2: inode - 17885541, total size - 13
after close f1: inode - 17885541, total size - 13
after fclose f2: inode - 17885541, total size - 13
```

4.7. Многопоточная реализация

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

#define FILENAME "tmp4.txt"

void* thread1()
{
  int f = open(FILENAME, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);

  for (char c = 'a'; c <= 'z'; c += 2)
  {
    write(f, &c, 1);
  }

  close(f);

  return NULL;
}

void* thread2()
{
  int f = open(FILENAME, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);

  for (char c = 'b'; c <= 'z'; c += 2)
  {
```

```
28      write(f, &c, 1);
29    }
30
31    close(f);
32
33    return NULL;
34  }
35
36  int main()
37  {
38    pthread_t t1, t2;
39    pthread_create(&t1, NULL, thread1, NULL);
40    pthread_create(&t2, NULL, thread2, NULL);
41
42    pthread_join(t1, NULL);
43    pthread_join(t2, NULL);
44
45    return 0;
46  }
```

Вывод программы на разных запусках

```
1   bdfhjlnprtvxz
2   acegikmoqsuwy
```

4.8. open() + O_APPEND

4.9. Однопоточная реализация

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <unistd.h>
6
7  #define FILENAME "tmp5.txt"
8
9  int main()
```

```
10 {
11   struct stat buf;
12
13   int f1 = open(FILENAME, O_WRONLY | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR
       );
14
15   int rc = fstat(f1, &buf);
16   if (!rc)
17   {
18     fprintf(stdout, "after open f1: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
19   }
20
21   int f2 = open(FILENAME, O_WRONLY | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR
       );
22
23   rc = fstat(f2, &buf);
24   if (!rc)
25   {
26     fprintf(stdout, "after open f2: inode - %ju, total size - %lld\n", (
         uintmax_t)buf.st_ino, buf.st_size);
27   }
28
29   for (char c = 'a'; c <= 'z'; c++)
30   {
31     if (c % 2)
32     {
33       write(f1, &c, 1);
34     }
35     else
36     {
37       write(f2, &c, 1);
38     }
39   }
40
41   rc = fstat(f1, &buf);
42   if (!rc)
43   {
```

```
44      fprintf(stdout, "before close f1: inode - %ju, total size - %lld\n", (
            uintmax_t)buf.st_ino, buf.st_size);
45    }
46
47    rc = fstat(f2, &buf);
48    if (!rc)
49    {
50      fprintf(stdout, "before close f2: inode - %ju, total size - %lld\n", (
            uintmax_t)buf.st_ino, buf.st_size);
51    }
52
53    close(f1);
54
55    rc = stat(FILENAME, &buf);
56    fprintf(stdout, "after close f1: inode - %ju, total size - %lld\n", (
          uintmax_t)buf.st_ino, buf.st_size);
57
58    close(f2);
59
60    rc = stat(FILENAME, &buf);
61    fprintf(stdout, "after fclose f2: inode - %ju, total size - %lld\n", (
          uintmax_t)buf.st_ino, buf.st_size);
62
63    return 0;
64  }
```

Вывод программы

```
1   abcdefghijklmnopqrstuvwxyz
```

4.10. Многопоточная реализация

```
1   #include <fcntl.h>
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <unistd.h>
5
6  #define FILENAME "tmp6.txt"
```

```
7
8  void* thread1()
9  {
10   int f = open(FILENAME, O_WRONLY | O_CREAT | O_APPEND,
11   S_IRUSR | S_IWUSR);
12
13   for (char c = 'a'; c <= 'z'; c += 2)
14   {
15     write(f, &c, 1);
16   }
17
18   close(f);
19
20   return NULL;
21 }
22
23 void* thread2()
24 {
25   int f = open(FILENAME, O_WRONLY | O_CREAT | O_APPEND,
26   S_IRUSR | S_IWUSR);
27
28   for (char c = 'b'; c <= 'z'; c += 2)
29   {
30     write(f, &c, 1);
31   }
32
33   close(f);
34
35   return NULL;
36 }
37
38 int main()
39 {
40   pthread_t t1, t2;
41
42   pthread_create(&t1, NULL, thread1, NULL);
43   pthread_create(&t2, NULL, thread2, NULL);
44
```

```
45    pthread_join(t1, NULL);
46    pthread_join(t2, NULL);
47
48    return 0;
49  }
```

Вывод программы на разных запусках

```
1  acbdefghijklmnopqrstuvwxyz
2  abcdfehgjilknmporqtsvuxwzy
```

Вывод stat

```
1  after open f1: inode - 17888251, total size - 0
2  after open f2: inode - 17888251, total size - 0
3  before close f1: inode - 17888251, total size - 26
4  before close f2: inode - 17888251, total size - 26
5  after close f1: inode - 17888251, total size - 26
6  after fclose f2: inode - 17888251, total size - 26
```

4.11. Пояснение

Если указан флаг O_APPEND, файл открывается в режиме добавления (записи в конец файла). Перед каждой операцией write файловый указатель будет устанавливаться в конце файла, как если бы использовался `lseek()`. Если этот флаг установлен, первая запись в файл не теряется.

5. Заключение

Таким образом, при буферизованном вводе-выводе все данные и при чтении, и при записи пишутся сначала в буфер, а только после этого в файл. При отсутствии буферизации данные сразу пишутся в/читаются из файла.

В программе с `fopen()`, `fprintf()` (с буферизацией) содержимое файла затирается при вызове `fclose()`. Поэтому размер файла до вызова `fclose()` по данным из `stat()` равен 0.

В программе с `open()`, `write()` (без буферизации) содержимое файла затирается при вызове `write()`. Поэтому размер файла до вызова `fclose()` по данным из `stat()` равен 13 — столько символов в итоге останется в файле.