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

по курсу «Операционные системы»

на тему: «Буферизованный и не буферизованный ввод–вывод»

Студент ИУ7-63Б (Группа) _____ (Подпись, дата) Авдейкина В. П. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Рязанова Н. Ю. (И. О. Фамилия)

2024 г.

1 Структуры

Версия ядра: `6.5.0-32-generic`.

Листинг 1.1 – Структура struct _IO_FILE

```
struct _IO_FILE
{
  int _flags;       /* High-order word is _IO_MAGIC; rest is flags. */

  /* The following pointers correspond to the C++ streambuf protocol. */
  char *_IO_read_ptr;   /* Current read pointer */
  char *_IO_read_end;   /* End of get area. */
  char *_IO_read_base;  /* Start of putback+get area. */
  char *_IO_write_base; /* Start of put area. */
  char *_IO_write_ptr;  /* Current put pointer. */
  char *_IO_write_end;  /* End of put area. */
  char *_IO_buf_base;   /* Start of reserve area. */
  char *_IO_buf_end;    /* End of reserve area. */

  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
  int _flags2;
  __off_t _old_offset; /* This used to be _offset but it's too small.  */

  /* 1+column number of pbase(); 0 is unknown. */
  unsigned short _cur_column;
  signed char _vtable_offset;
```

```
  char _shortbuf[1];

  _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

Листинг 1.2 – Структура struct stat

```
struct stat {
  unsigned long st_dev;      /* Device.  */
  unsigned long st_ino;      /* File serial number.  */
  unsigned int  st_mode;     /* File mode.  */
  unsigned int  st_nlink;    /* Link count.  */
  unsigned int  st_uid;      /* User ID of the file's owner.  */
  unsigned int  st_gid;      /* Group ID of the file's group. */
  unsigned long st_rdev;     /* Device number, if device.  */
  unsigned long __pad1;
  long       st_size;    /* Size of file, in bytes.  */
  int        st_blksize; /* Optimal block size for I/O.  */
  int        __pad2;
  long       st_blocks;  /* Number 512-byte blocks allocated. */
  long       st_atime;   /* Time of last access.  */
  unsigned long st_atime_nsec;
  long       st_mtime;   /* Time of last modification.  */
  unsigned long st_mtime_nsec;
  long       st_ctime;   /* Time of last status change.  */
  unsigned long st_ctime_nsec;
  unsigned int  __unused4;
  unsigned int  __unused5;
};
```

2 Программы

2.1 Первая программа

2.1.1 Без использования потоков

Листинг 2.1 – Первая программа

```
#include <stdio.h>
#include <fcntl.h>
int main()
{
  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);
  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;
}
```

```
sheglar:src$ ./1.out
Aubvcwdxeyfzg
```

Рисунок 2.1 — Результат выполнения программы

2.1.2 С использованием двух дополнительных потоков

Листинг 2.2 – Первая программа

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

struct thread_arg {
    FILE *fs;
    int i;
};

void *thread_start(void *arg)
{
    struct thread_arg *targ = arg;
    char c;

    while (fscanf(targ->fs, "%c", &c) == 1)
        fprintf(stdout, "thread %d: %c\n", targ->i, c);

    return NULL;
}

int main()
{
    int fd = open("alphabet.txt", O_RDONLY);
    FILE *fs[2] = {fdopen(fd, "r"),
                   fdopen(fd, "r")};
    char buff[2][20];

    setvbuf(fs[0], buff[0], _IOFBF, 20);
    setvbuf(fs[1], buff[1], _IOFBF, 20);

    pthread_t thr[2];
    struct thread_arg targ[2];

    for (int i = 0; i < 2; i++)
    {
        targ[i].fs = fs[i];
        targ[i].i = i;

        if (pthread_create(&thr[i], NULL, thread_start,
           &targ[i]))
```

```
        {
            perror("pthread_create");
            return 1;
        }
    }

    for (int i = 0; i < 2; i++)
        if (pthread_join(thr[i], NULL))
        {
            perror("pthread_join");
            return 1;
        }

    return 0;
}
```

```
sheglar:src$ ./1_thread.out
thread 0: A
thread 0: b
thread 0: c
thread 0: d
thread 0: e
thread 0: f
thread 0: g
thread 0: h
thread 0: i
thread 1: u
thread 1: v
thread 1: w
thread 1: x
thread 1: y
thread 1: z
thread 1:

thread 0: j
thread 0: k
thread 0: l
thread 0: m
thread 0: n
thread 0: o
thread 0: p
thread 0: q
thread 0: r
thread 0: s
thread 0: t
```

Рисунок 2.2 — Результат выполнения программы

2.2 Вторая программа, первый вариант

2.2.1 Без использования потоков

Листинг 2.3 – Вторая программа, первый вариант

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

int main()
{
  char c;
  int fd1 = open("alphabet.txt", O_RDONLY);
  int fd2 = open("alphabet.txt", O_RDONLY);

  int flag1 = 1, flag2 = 1;

  while ((flag1 == 1) && (flag2 == 1))
  {
    if (1 == (flag1 = read(fd1, &c, 1)))
    {
      write(1, &c, 1);

      if (1 == (flag2 = read(fd2, &c, 1)))
          write(1, &c, 1);
    }
  }

  return 0;
}
```

```
sheglar:src$ ./2.out
AAbbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz

sheglar:src$
```

Рисунок 2.3 — Результат выполнения программы

2.2.2 С использованием двух дополнительных потоков

Листинг 2.4 – Вторая программа, первый вариант (два дополнительных потока)

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

void *thread_start(void *arg)
{
  int *fd = arg;
  char c;

  while (read(*fd, &c, 1))
    write(1, &c, 1);

  return NULL;
}

int main()
{
  int fd[2] = {open("alphabet.txt", O_RDONLY),
               open("alphabet.txt", O_RDONLY)};
  pthread_t thr[2];

  for (int i = 0; i < 2; i++)
    if (pthread_create(&thr[i], NULL, thread_start, &fd[i]))
    {
      perror("pthread_create");
      return 1;
    }

  for (int i = 0; i < 2; i++)
    if (pthread_join(thr[i], NULL))
    {
      perror("pthread_join");
      return 1;
    }

  close(fd[0]);
  close(fd[1]);

```

```
  return 0;
}
```

```
sheglar:src$ ./2_thread.out
AbcdefghijklmnAobpcqdresftguhviwjxykzl
mnopqrstuvwxyz
sheglar:src$
```

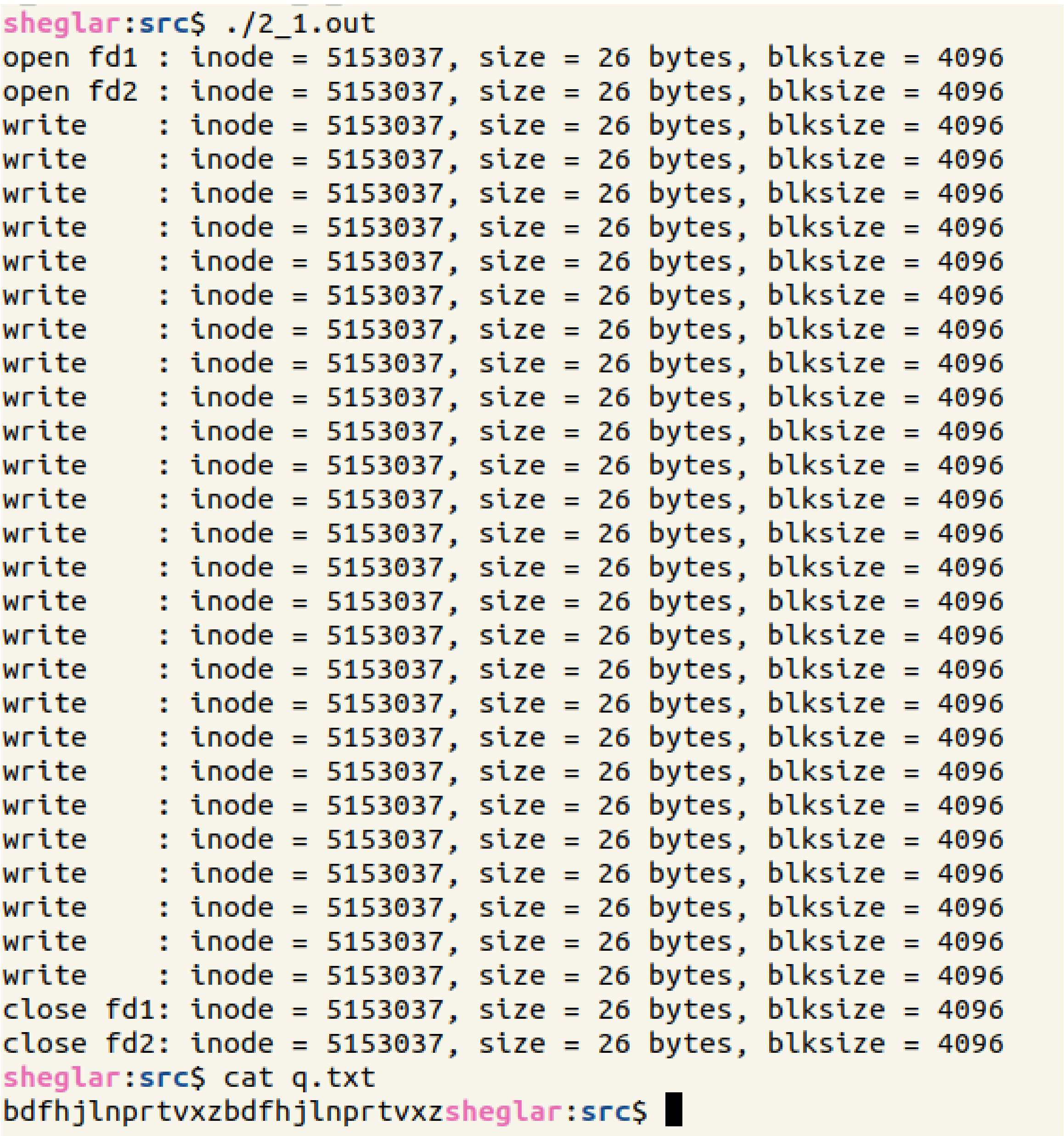
Рисунок 2.4 — Результат выполнения программы

2.3 Вторая программа, второй вариант

2.3.1 Без использования потоков

Листинг 2.5 – Вторая программа, второй вариант

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>

struct stat statbuf;

#define PRINT_STAT(action) \
  do { \
    stat("q.txt", &statbuf); \
    fprintf(stdout, action ": inode = %ld, size = %ld bytes,
       blksize = %ld\n", \
      statbuf.st_ino, statbuf.st_size, \
      statbuf.st_blksize); \
  } while (0);

int main()
{
  int fd1 = open("q.txt", O_RDWR);
  PRINT_STAT("open fd1 ");
  int fd2 = open("q.txt", O_RDWR);
  PRINT_STAT("open fd2 ");
  for (char c = 'a'; c <= 'z'; c++)
  {
    if (c % 2)
      write(fd1, &c, 1);
    else
      write(fd2, &c, 1);
    PRINT_STAT("write    ");
  }
  close(fd1);
  PRINT_STAT("close fd1");
  close(fd2);
  PRINT_STAT("close fd2");
  return 0;
}
```



```
sheglar:src$ ./2_1.out
open fd1 : inode = 5153037, size = 26 bytes, blksize = 4096
open fd2 : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
write    : inode = 5153037, size = 26 bytes, blksize = 4096
close fd1: inode = 5153037, size = 26 bytes, blksize = 4096
close fd2: inode = 5153037, size = 26 bytes, blksize = 4096
sheglar:src$ cat q.txt
bdfhjlnprtvxzbdfhjlnprtvxzsheglar:src$ 
```

Рисунок 2.5 — Результат выполнения программы

2.3.2 С использованием двух дополнительных потоков

Листинг 2.6 – Вторая программа, второй вариант (два дополнительных потока)

```c
#include <fcntl.h>
#include <unistd.h>
#include <pthread.h>
#include <stdio.h>
#include <sys/stat.h>

struct stat statbuf;
```

```

#define PRINT_STAT(action, i) \
  do { \
    stat("q.txt", &statbuf); \
    fprintf(stdout, action " %d: inode number = %ld, size = %ld bytes, blksize = %ld\n", \
      i, statbuf.st_ino, statbuf.st_size, \
      statbuf.st_blksize); \
  } while (0);

pthread_mutex_t mutex;

struct thread_arg {
  int fd;
  int i;
};

void *thread_start(void *arg)
{
  struct thread_arg *targ = arg;

  for (char c = 'a'; c <= 'z'; c++)
    if (c % 2 == targ->i)
    {
      pthread_mutex_lock(&mutex);
      write(targ->fd, &c, 1);
      PRINT_STAT("write", targ->i);
      pthread_mutex_unlock(&mutex);
    }

  return NULL;
}

int main()
{
  int fd[2] = {open("q.txt", O_RDWR),
               open("q.txt", O_RDWR | O_APPEND)};
  pthread_t thr[2];
  struct thread_arg targ[2];

  if (pthread_mutex_init(&mutex, NULL))
```

```
  {
    perror("pthread_mutex_init");
    return 1;
  }

  for (int i = 0; i < 2; i++)
{
    targ[i].fd = fd[i];
    targ[i].i = i;

    if (pthread_create(&thr[i], NULL, thread_start, &targ[i]))
    {
      perror("pthread_create");
      return 1;
    }
  }

  for (int i = 0; i < 2; i++)
    if (pthread_join(thr[i], NULL))
    {
      perror("pthread_join");
      return 1;
    }

  if (pthread_mutex_destroy(&mutex))
  {
    perror("pthread_mutex_destroy");
    return 1;
  }

  close(fd[0]);
  close(fd[1]);

  return 0;
}
```

```
sheglar:src$ ./2_1_thread.out
write 0: inode number = 5153037, size = 1 bytes, blksize = 4096
write 0: inode number = 5153037, size = 2 bytes, blksize = 4096
write 0: inode number = 5153037, size = 3 bytes, blksize = 4096
write 0: inode number = 5153037, size = 4 bytes, blksize = 4096
write 0: inode number = 5153037, size = 5 bytes, blksize = 4096
write 0: inode number = 5153037, size = 6 bytes, blksize = 4096
write 0: inode number = 5153037, size = 7 bytes, blksize = 4096
write 0: inode number = 5153037, size = 8 bytes, blksize = 4096
write 0: inode number = 5153037, size = 9 bytes, blksize = 4096
write 0: inode number = 5153037, size = 10 bytes, blksize = 4096
write 0: inode number = 5153037, size = 11 bytes, blksize = 4096
write 0: inode number = 5153037, size = 12 bytes, blksize = 4096
write 0: inode number = 5153037, size = 13 bytes, blksize = 4096
write 1: inode number = 5153037, size = 14 bytes, blksize = 4096
write 1: inode number = 5153037, size = 15 bytes, blksize = 4096
write 1: inode number = 5153037, size = 16 bytes, blksize = 4096
write 1: inode number = 5153037, size = 17 bytes, blksize = 4096
write 1: inode number = 5153037, size = 18 bytes, blksize = 4096
write 1: inode number = 5153037, size = 19 bytes, blksize = 4096
write 1: inode number = 5153037, size = 20 bytes, blksize = 4096
write 1: inode number = 5153037, size = 21 bytes, blksize = 4096
write 1: inode number = 5153037, size = 22 bytes, blksize = 4096
write 1: inode number = 5153037, size = 23 bytes, blksize = 4096
write 1: inode number = 5153037, size = 24 bytes, blksize = 4096
write 1: inode number = 5153037, size = 25 bytes, blksize = 4096
write 1: inode number = 5153037, size = 26 bytes, blksize = 4096
sheglar:src$ cat q.txt
bdfhjlnprtvxzacegikmoqsuwysheglar:src$
```

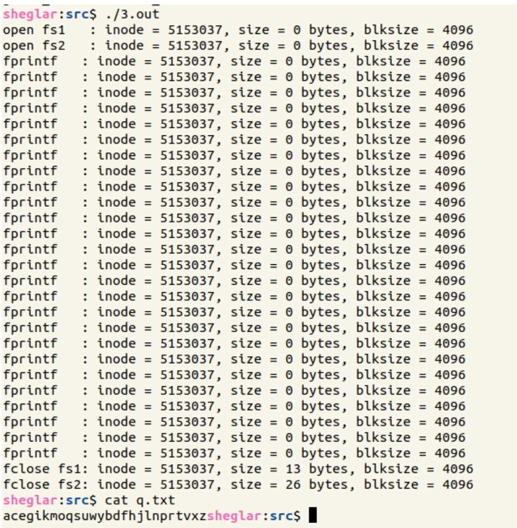
Рисунок 2.6 — Результат выполнения программы

2.4 Первая программа

2.4.1 Без использования потоков

Листинг 2.7 – Третья программа

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>

#define PRINT_STAT(action) \
  do { \
    stat("q.txt", &statbuf); \
    fprintf(stdout, action ": inode = %ld, size = %ld bytes,
       blksize = %ld\n", \
      statbuf.st_ino, statbuf.st_size, \
      statbuf.st_blksize); \
  } while (0);

struct stat statbuf;

int main()
{
  FILE *fs1 = fopen("q.txt", "w");
  FILE *fs2 = fopen("q.txt", "a");

  for (char c = 'a'; c <= 'z'; c++)
  {
    if (c % 2)
      fprintf(fs1, "%c", c);
    else
      fprintf(fs2, "%c", c);
    PRINT_STAT("write");
  }

  fclose(fs1);
  PRINT_STAT("fclose fs1");
  fclose(fs2);

  return 0;
}
```



```
sheglar:src$ ./3.out
open fs1   : inode = 5153037, size = 0 bytes, blksize = 4096
open fs2   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fprintf   : inode = 5153037, size = 0 bytes, blksize = 4096
fclose fs1: inode = 5153037, size = 13 bytes, blksize = 4096
fclose fs2: inode = 5153037, size = 26 bytes, blksize = 4096
sheglar:src$ cat q.txt
acegikmoqsuwybdfhjlnprtvxzsheglar:src$
```

Рисунок 2.7 — Результат выполнения программы

2.4.2 С использованием двух дополнительных потоков

Листинг 2.8 – Третья программа (два дополнительных потока)

```c
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <pthread.h>

struct thread_arg {
    FILE *fs;
    int i;
};

#define PRINT_STAT(action, i) \
    do { \
        stat("q.txt", &statbuf); \
        fprintf(stdout, action " %d: inode = %ld, size = %ld bytes, blksize = %ld\n", \
            i, statbuf.st_ino, statbuf.st_size, \
            statbuf.st_blksize); \
    } while (0);

struct stat statbuf;

void *thread_start(void *arg)
{
```

```
    struct thread_arg *targ = arg;

    for (char c = 'a'; c <= 'z'; c++)
        if (c % 2 == targ->i)
        {
            fprintf(targ->fs, "%c", c);
            PRINT_STAT("write", targ->i);
        }
}

int main()
{
    FILE *fs[2] = {fopen("q.txt", "w"),
                   fopen("q.txt", "a")};
    pthread_t thr[2];
    struct thread_arg targ[2];

    for (int i = 0; i < 2; i++)
    {
        targ[i].fs = fs[i];
        targ[i].i = i;

        if (pthread_create(&thr[i], NULL, thread_start,
           &targ[i]))
        {
            perror("pthread_create");
            return 1;
        }
    }

    for (int i = 0; i < 2; i++)
        if (pthread_join(thr[i], NULL))
        {
            perror("pthread_join");
            return 1;
        }

    fclose(fs[0]);
    PRINT_STAT("fclose fs", 0);
    fclose(fs[1]);
    PRINT_STAT("fclose fs", 1);
```

```

    return 0;
}
```

```
sheglar:src$ ./3_thread.out
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 0: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
write 1: inode = 5153037, size = 0 bytes, blksize = 4096
fclose fs 0: inode = 5153037, size = 13 bytes, blksize = 4096
fclose fs 1: inode = 5153037, size = 26 bytes, blksize = 4096
sheglar:src$ cat q.txt
bdfhjlnprtvxzacegikmoqsuwysheglar:src$
```

Рисунок 2.8 — Результат выполнения программы