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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

Студент	Никифорова Ирина Андреевна
Группа	РК6-61б
Тип задания	лабораторная работа
Тема лабораторной работы	Многопоточное программирование

Студент _____ **Никифорова И. А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Федорук В.Г.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2019 г.

Оглавление

Задание на лабораторную работу

Разработать программу, реализующую обработку текстовых файлов и функционирующую в рамках 3-х потоков.

Корневой поток является управляющим и принимает в качестве аргументов имена 2-х файлов. В начале своей работы он порождает 2 потока, передавая каждому по одному имени файла.

Первый порожденный поток осуществляет побайтное считывание файла и вывод с небольшой задержкой прочитанных байт (по одному на строке) в верхнем регистре на стандартный вывод.

Второй порожденный поток осуществляет побайтное считывание файла и вывод с небольшой задержкой прочитанных байт (по одному на строке) в нижнем регистре на стандартный вывод.

Порожденные потоки функционируют параллельно.

Управляющий поток считывает со стандартного ввода строки, содержащие имена новых текстовых файлов и заставляет порожденные потоки немедленно переходить на обработку новых файлов по следующей схеме: второй порожденный поток переходит к обработке файла, ранее обрабатываемого первым потоком; первый порожденный поток начинает обрабатывать файл, имя которого ему передает управляющий поток.

Описание структуры программы и реализованных способов взаимодействия потоков

Программа состоит из функций *lower, upper, remove_enter, main* и набора глобальных переменных. Далее все части будут рассмотрены подробнее.

В начале программы объявляются глобальные переменные, с помощью которых организовано взаимодействие между потоками (листинг 1). *filename1* и *filename2* необходимы для установления имен файлов для функций *lower* и *upper* соответственно. *name_changed_1* и *name_changed_2* - флаги, оповещающие функции об изменениях названий файлов *filename1* и *filename2* соответственно. *end* - флаг, устанавливаемый, если была получена команда о завершении программы.

Листинг 1. Глобальные переменные, через которые общаются потоки

```
1. char* filename1 = NULL;
2. char* filename2 = NULL;
3. char name_changed_1 = 0;
4. char name_changed_2 = 0;
5. char end = 0;
```

Функция *remove_enter* (листинг 2) не относится непосредственно к логике задачи. Она заменяет на нулевой байт лишний перевод строки в считываемых функцией *getline* названиях файлов.

Листинг 2. Дополнительная функция *remove_enter*

```
1. void remove_enter(char* name)
2.     name[strlen(name)-1] = '\0';
3. }
```

Функции *lower* (листинг 3) и *upper* (листинг 4) выполняют приведение всех символов заданного в соответствующей им глобальной переменной файла к нужному регистру (нижнему или верхнему соответственно). Они отличаются только названиями глобальных переменных, описанных выше.

Каждая из перечисленных выше функций работает, пока не будет установлен общий для функций флаг окончания выполнения *end.* В цикле проверки данного флага (строка 4, листинги 3 и 4) каждая функция сначала открывает файл по имени, записанном в глобальной переменной *filename1* (*lower*) или *filename2* (*upper*).

Далее функция производит чтение из файла во внутреннем цикле (строка 17, листинги 3 и 4), который может быть остановлен только по окончании

файла (*read_num* на каждой итерации сохраняет количество прочитанных символов, если эта переменная станет равна нулю, то файл завершен) или установке одного из флагов *name_changed_1* (*lower*) / *name_changed_2* (*upper*) или *end*. Эта проверка флагов необходима, чтобы переключение на новый файл или остановка программы происходили сразу же после команды к данным действиям.

Далее в функции происходит обнуление флага изменений в имени файла и закрытие файла для корректной работы.

Если за время работы программы файл был исчерпан, а его имя не изменилось, на следующей итерации внешнего цикла (строка 4, листинги 3 и 4) он будет вновь открыт и прочитан.

Листинг 3. Функция *lower*

```
1. void* lower(void* attr) {
2.     // пока внешним потоком не будет
3.     // установлен флаг окончания выполнения
4.     while (!end) {
5.         // открытие файла для чтения
6.         FILE* in = fopen(filename2, "r");
7.         if (in == NULL) {
8.             return (void*)-1;
9.         }
10.
11.            // чтение по одному символу и вывод с переходом на новую
12.            // строку, пока файл не закончится или вне потока
13.            // не будет установлен один из флагов name_changed или end
14.            char symbs[2];
15.            symbs[1] = '\n';
16.            size_t read_num = 1;
17.            while (read_num >= 1 && name_changed_2 == 0 && end == 0) {
18.                read_num = fread(&symbs[0], 1, 1, in);
19.                symbs[0] = tolower(symbs[0]);
20.                write(1, &symbs, 2);
21.                sleep(1);
22.            }
23.
24.            // снимаем флаг изменения имени файла
25.            name_changed_2 = 0;
26.
27.            // закрытие файла
28.            int is_closed = fclose(in);
29.            if (is_closed == EOF) {
30.                return (void*)-1;
31.            }
32.        }
33.
34.        pthread_exit(0);
35.    }
```

Листинг 4. Функция *upper*

```
void* upper(void* attr) {
    // пока внешним потоком не будет
    // установлен флаг окончания выполнения
    while (!end) {
        // открытие файла для чтения
        FILE* in = fopen(filename1, "r");
        if (in == NULL) {
            return (void*)-1;
        }

            // чтение по одному символу и вывод с переходом на новую
            // строку, пока файл не закончится или вне потока
            // не будет установлен один из флагов name_changed или end
            char symbs[2];
            symbs[1] = '\n';
            size_t read_num = 1;
            while (read_num >= 1 && name_changed_1 == 0 && end == 0) {
                read_num = fread(&symbs[0], 1, 1, in);
                symbs[0] = toupper(symbs[0]);
                write(1, &symbs, 2);
                sleep(1);
            }

            // снимаем флаг изменения имени файла
            name_changed_1 = 0;

            // закрытие файла
            int is_closed = fclose(in);
            if (is_closed == EOF) {
                return (void*)-1;
            }
        }

        pthread_exit(0);
    }
```

Функция main (листинг 5) решает задачу управления потоками и считывания новых имен файлов. Она создаёт потоки для функций *lower* и *upper* и ожидает их завершения. Схема взаимодействия потоков представлена на рисунке 1.

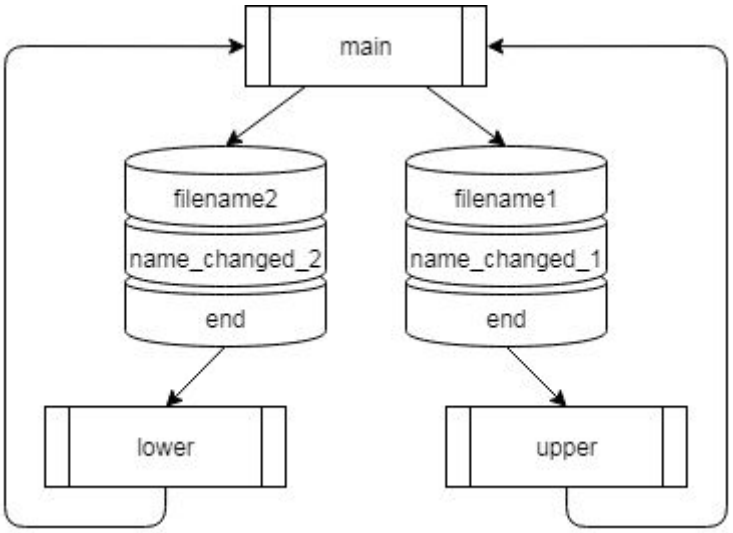

Рис. 1. Взаимодействие потоков. Поток функции main порождает два потока: для функции lower и для функции upper. Порожденные потоки получают информацию об изменениях от корневого с помощью специальных глобальных переменных. Корневой получает информацию от порожденных только в том случае, если они завершены, т.к. ожидает их завершения.

Листинг 5. Функция *main*

```
1. int main() {
2.   // чтение названий файла
3.    size_t size_of_line = 0;
4.    ssize_t read_num = getline(&filename1, &size_of_line, stdin);
5.    read_num = getline(&filename2, &size_of_line, stdin);
6.
7.    // удаление лишних \n из строк с названиями файлов
8.    remove_enter(filename1);
9.    remove_enter(filename2);
10.
11.      // задание атрибутов создаваемых потоков
12.      pthread_attr_t pattr;
13.      pthread_attr_init(&pattr);
14.      pthread_attr_setscope(&pattr, PTHREAD_SCOPE_SYSTEM);
15.      pthread_attr_setdetachstate(&pattr, PTHREAD_CREATE_JOINABLE);
16.
17.      // создание и запуск потоков
18.      pthread_t pid1, pid2;
19.      pthread_create(&pid1, &pattr, upper, filename1);
20.      pthread_create(&pid2, &pattr, lower, filename2);
21.
22.      // чтение новых имен файлов, пока это возможно
23.      // и изменение их в глобальных переменных
24.      char* filename_swp = NULL;
25.      while (getline(&filename_swp, &size_of_line, stdin) > 0) {
26.          remove_enter(filename_swp);
27.          if (strcmp(filename_swp,"stop") == 0) {
28.              end = 1;
29.              free(filename_swp);
30.              free(filename1);
```

```
31.             free(filename2);
32.             pthread_attr_destroy(&pattr);
33.             pthread_exit(0);
34.         }
35.         strcpy(filename2, filename1);
36.         strcpy(filename1, filename_swp);
37.         name_changed_1 = 1;
38.         name_changed_2 = 1;
39.     }
40.
41.     // ожидание завершения дочерних потоков
42.     int retval1 = 0, retval2 = 0;
43.     pthread_join(pid1, (void*)&retval1);
44.     pthread_join(pid2, (void*)&retval2);
45.
46.     // освобождение выделенной памяти
47.     free(filename1);
48.     free(filename2);
49.     pthread_attr_destroy(&pattr);
50.     pthread_exit(0);
51. }
```

Описание основных используемых структур данных

В программе использовалась только структура данных массив для хранения строк как массивов символов в языке СИ.

Блок-схема программы

Блок-схема функции main представлена на рисунках 2 и 3, функции lower - на рисунке 4. Блок-схема функции remove_enter не приводится, так как она не связана непосредственно с логикой работы программы. Блок-схема функции upper также не приводится, так как она аналогична блок-схеме функции lower.

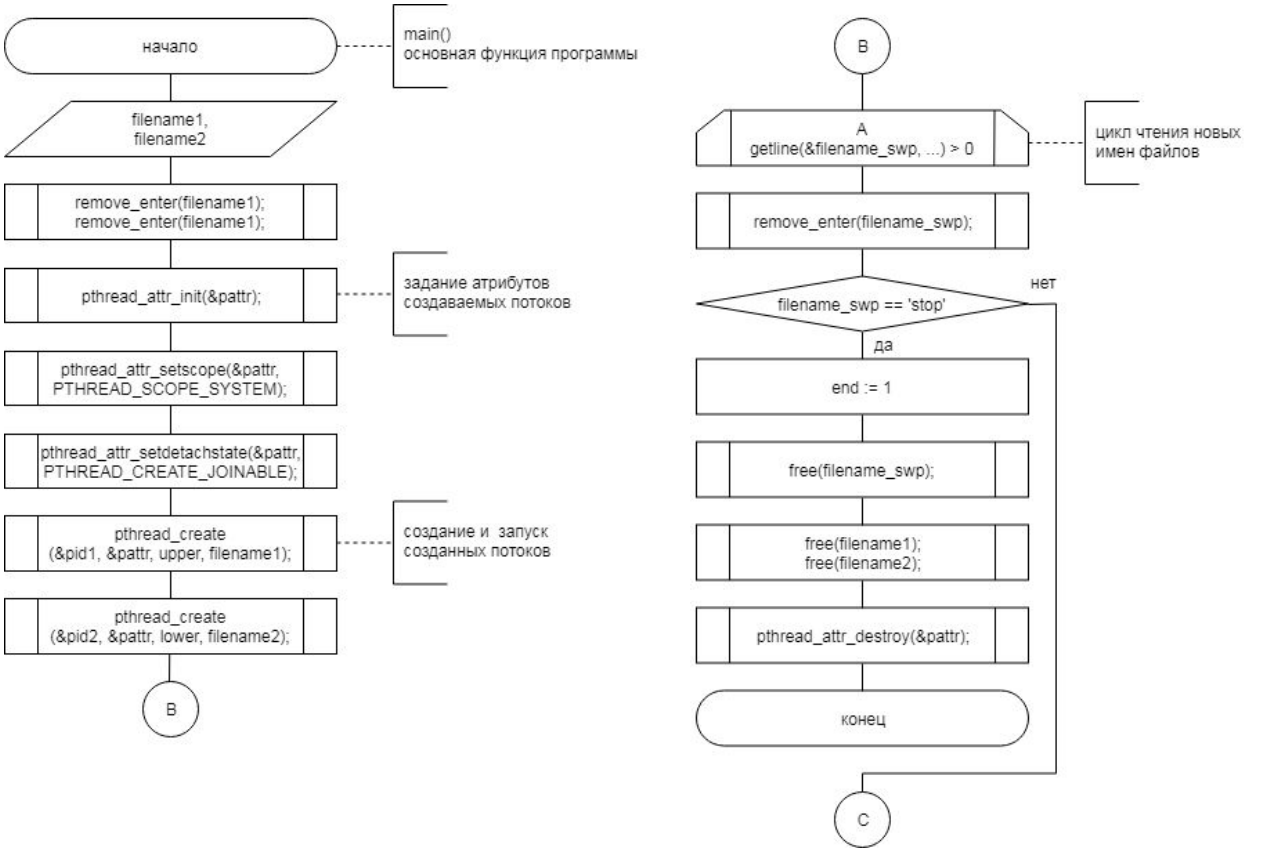

Рис. 2. Блок-схема функции main, начало

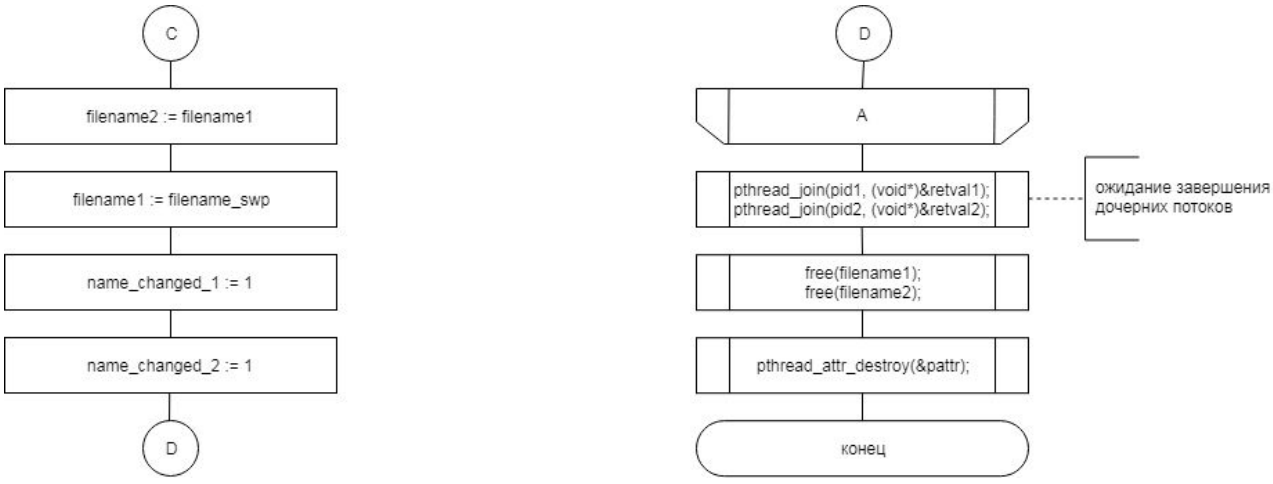

Рис. 3. Блок-схема функции main, окончание

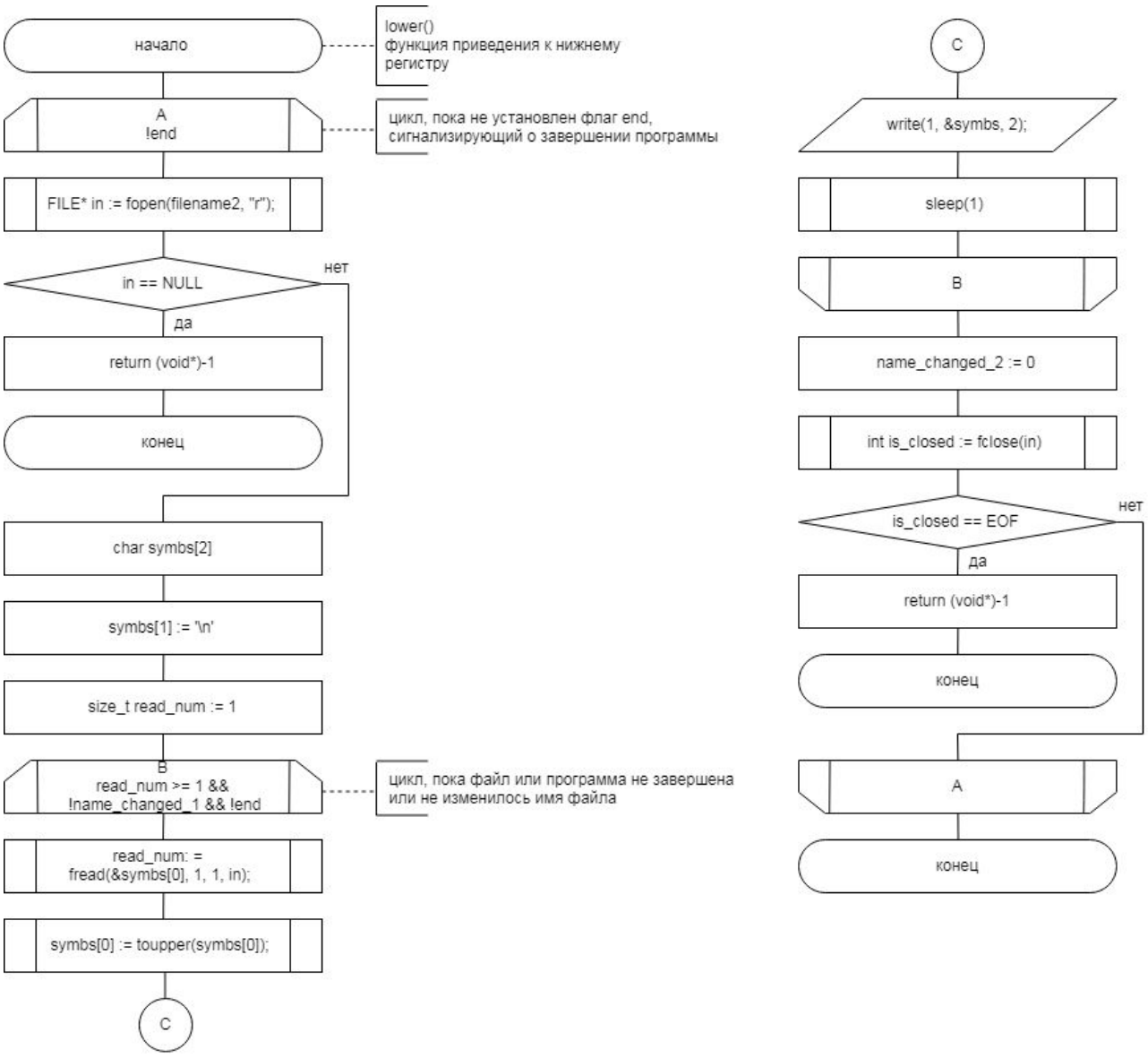

Рис. 4. Блок-схема функции lower

Примеры результатов работы программы

Результаты работы программы представлены в таблице 1, где номер строки соответствует времени ввода указанных в строке входных данных или вывода выходных. Файл *a.in* содержит только буквы “A” в различных регистрах. Аналогично файлы *b.in* и *c.in* содержат буквы “B” и “C” соответственно.

Таблица 1. Примеры входных и выходных данных для программы

Входные данные	Результат работы
1. a.in 2. b.in 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. c.in 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. stop	1. 2. 3. b 4. A 5. b 6. A 7. b 8. A 9. A 10. b 11. A 12. b 13. A 14. b 15. A 16. b 17. C 18. a 19. C 20. a 21. a 22. C 23. C 24. a 25. C 26. a 27. C 28. a 29. C 30. a 31. C
1. b.in 2. c.in 3. 4. 5. 6. 7. 8. stop	1. 2. 3. B 4. c 5. B 6. c 7. B 8. c

Текст программы

Полный текст программы с комментариями приведен в листинге 6.

Листинг 6. Полный текст программы

```
1.    #include <stdio.h>
2.    #include <stdlib.h>
3.    #include <unistd.h>
4.    #include <ctype.h>
5.    #include <errno.h>
6.    #include <string.h>
7.    #include <pthread.h>
8.
9.    char* filename1 = NULL;
10.   char* filename2 = NULL;
11.   char name_changed_1 = 0;
12.   char name_changed_2 = 0;
13.   char end = 0;
14.
15.   // перевод в нижний регистр
16.   void* lower(void* attr) {
17.      // пока внешним потоком не будет
18.      // установлен флаг окончания выполнения
19.      while (!end) {
20.          // открытие файла для чтения
21.          FILE* in = fopen(filename2, "r");
22.          if (in == NULL) {
23.              return (void*)-1;
24.          }
25.
26.          // чтение по одному символу и вывод с переходом на новую
27.          // строку, пока файл не закончится или вне потока
28.          // не будет установлен один из флагов name_changed или end
29.          char symbs[2];
30.          symbs[1] = '\n';
31.          size_t read_num = 1;
32.          while (read_num >= 1 && name_changed_2 == 0 && end == 0) {
33.              read_num = fread(&symbs[0], 1, 1, in);
34.              symbs[0] = tolower(symbs[0]);
35.              write(1, &symbs, 2);
36.              sleep(1);
37.          }
38.
39.          // снимаем флаг изменения имени файла
40.          name_changed_2 = 0;
41.
42.          // закрытие файла
43.          int is_closed = fclose(in);
44.          if (is_closed == EOF) {
45.              return (void*)-1;
46.          }
47.      }
48.
49.      pthread_exit(0);
```

```
50.   }
51.
52.   // перевод в верхний регистр
53.   void* upper(void* attr) {
54.      // пока внешним потоком не будет
55.      // установлен флаг окончания выполнения
56.      while (!end) {
57.          // открытие файла для чтения
58.          FILE* in = fopen(filename1, "r");
59.          if (in == NULL) {
60.              return (void*)-1;
61.          }
62.
63.          // чтение по одному символу и вывод с переходом на новую
64.          // строку, пока файл не закончится или вне потока
65.          // не будет установлен один из флагов name_changed или end
66.          char symbs[2];
67.          symbs[1] = '\n';
68.          size_t read_num = 1;
69.          while (read_num >= 1 && name_changed_1 == 0 && end == 0) {
70.              read_num = fread(&symbs[0], 1, 1, in);
71.              symbs[0] = toupper(symbs[0]);
72.              write(1, &symbs, 2);
73.              sleep(1);
74.          }
75.
76.          // снимаем флаг изменения имени файла
77.          name_changed_1 = 0;
78.
79.          // закрытие файла
80.          int is_closed = fclose(in);
81.          if (is_closed == EOF) {
82.              return (void*)-1;
83.          }
84.      }
85.
86.      pthread_exit(0);
87.   }
88.
89.   // удаление считанного перевода строки на конце названия файле
90.   void remove_enter(char* name) {
91.      name[strlen(name)-1] = '\0';
92.   }
93.
94.   // контроль потоков и считывание новых имен файлов
95.   int main() {
96.     // чтение названий файла
97.      size_t size_of_line = 0;
98.      ssize_t read_num = getline(&filename1, &size_of_line, stdin);
99.      read_num = getline(&filename2, &size_of_line, stdin);
100.
101.     // удаление лишних \n из строк с названиями файлов
102.     remove_enter(filename1);
103.     remove_enter(filename2);
104.
105.     // задание атрибутов создаваемых потоков
106.     pthread_attr_t pattr;
```

```
107.     pthread_attr_init(&pattr);
108.     pthread_attr_setscope(&pattr, PTHREAD_SCOPE_SYSTEM);
109.     pthread_attr_setdetachstate(&pattr, PTHREAD_CREATE_JOINABLE);
110.
111.     // создание и запуск потоков
112.     pthread_t pid1, pid2;
113.     pthread_create(&pid1, &pattr, upper, filename1);
114.     pthread_create(&pid2, &pattr, lower, filename2);
115.
116.     // чтение новых имен файлов, пока это возможно
117.     // и изменение их в глобальных переменных
118.     char* filename_swp = NULL;
119.     while (getline(&filename_swp, &size_of_line, stdin) > 0) {
120.         remove_enter(filename_swp);
121.         if (strcmp(filename_swp,"stop") == 0) {
122.             end = 1;
123.             free(filename_swp);
124.             free(filename1);
125.             free(filename2);
126.             pthread_attr_destroy(&pattr);
127.             pthread_exit(0);
128.         }
129.         strcpy(filename2, filename1);
130.         strcpy(filename1, filename_swp);
131.         name_changed_1 = 1;
132.         name_changed_2 = 1;
133.     }
134.
135.     // ожидание завершения дочерних потоков
136.     int retval1 = 0, retval2 = 0;
137.     pthread_join(pid1, (void*)&retval1);
138.     pthread_join(pid2, (void*)&retval2);
139.
140.     // освобождение выделенной памяти
141.     free(filename1);
142.     free(filename2);
143.     pthread_attr_destroy(&pattr);
144.     pthread_exit(0);
145. }
```