МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №4

Выполнила студентка группы Б17-501: Баранова Дарья

Проверил: Ктитров С.В

5

Москва, 2020

Задача

Вариант D–2

Тема «Данные потоков»

Разработать программу, иллюстрирующую средства, позволяющие по ключу обращаться к данным, специфичным для каждого потока. Программа должна позволять вводить разнотипные данные в указанный поток и считывать данные указанного потока.

Выполнение

Чтобы иллюстрировать возможность создания разнотипных данных с разными значениями в разных потоках по одному ключу, будем разделять потоки на четные и нечетные и присваивать им имя “Even thread” или “Odd thread”. Программой создается несколько потоков (по умолчанию выбрано 5). Для каждого можно задать данные типов int, float, char, pid, bool (чтобы добавить тип bool, была подключена дополнительная библиотека).

В переменную типа int будем помещать номер потока, в переменную типа pid – ID конкретного потока, в переменную типа char – имя потока – “Even thread”/”Odd thread”, в переменную типа float – число либо 1.01, либо 2.22, в переменную типа bool – true/false. После задания данных потоками вызывается функция получения этих данных для вывода на экран.

Таким образом, программа выводит на экран данные специфичные для каждого потока:

- parameter – это значение переменной типа int. Она задаёт четность потока
- name – значение переменной типа char
- data– значение переменной типа float
- bool_data – значение переменной типа bool
- thread_id – значение переменной типа pthread_t

Пример работы:

```
super@super-Inspiron-5558:~/Desktop/unik/D$ cc -pthread main.c -o main
super@super-Inspiron-5558:~/Desktop/unik/D$ ./main
parameter = 1, name = Odd thread, data = 1.010000, bool_data = 1, thread id = 139903277364992
parameter = 0, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139903285757696
parameter = 3, name = Odd thread, data = 1.010000, bool_data = 1, thread id = 139903056738048
parameter = 2, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139903268972288
parameter = 4, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139903048345344
super@super-Inspiron-5558:~/Desktop/unik/D$ ./main
parameter = 0, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139969798752000
parameter = 1, name = Odd thread, data = 1.010000, bool_data = 1, thread id = 139969790359296
parameter = 2, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139969695840000
parameter = 4, name = Even thread, data = 2.220000, bool_data = 0, thread id = 139969773573888
parameter = 3, name = Odd thread, data = 1.010000, bool_data = 1, thread id = 139969781966592
super@super-Inspiron-5558:~/Desktop/unik/D$
```

Код программы:

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <pthread.h>
5.  #include <stdbool.h>
6.
7.  //зададим создаваемых количество потоков
8.  #define NUM_THREADS     5
9.
10. static  pthread_key_t key;
11.
12. /*  собственный тип данных. Укажем в нем переменные разных типов, которыми
13.     будем пользоваться
14. */
15. typedef struct data_bloc {
16.     pthread_t TID_TYPE;
17.     int       INT_TYPE;
18.     float     FLOAT_TYPE;
19.     bool      BOOL_TYPE;
20.     char*     STRING_TYPE;
21. } data_t;
22.
23. void put_msg() {
24.     // выведем данные для каждого потока
25.     printf("parameter = %u, name = %s, data = %f, bool_data = %d, thread
       id = %ld\n",
26.         ((data_t*)pthread_getspecific(key))->INT_TYPE,
27.         ((data_t*)pthread_getspecific(key))->STRING_TYPE,
28.         ((data_t*)pthread_getspecific(key))->FLOAT_TYPE,
29.         ((data_t*)pthread_getspecific(key))->BOOL_TYPE,
30.         ((data_t*)pthread_getspecific(key))->TID_TYPE);
31. }
32.
33. static pthread_once_t once = PTHREAD_ONCE_INIT;
34.
35. static void destructor(void* db) { // деструктор собственных данных
36.     data_t *p = (data_t*)db;
37.     free(p);
38. }
39.
40. static void once_creator(void) { // создаёт единый на процесс ключ для
       данных data_
41.     pthread_key_create(&key, destructor);
42. }
43.
44. void* thread_proc(void *data) { // функция потока
45.     int param = (int)(intptr_t)data;
```

```
46.     pthread_once(&once, once_creator);  // гарантируем единичность
    создания ключа
47.
48.
49.     pthread_setspecific(key, malloc(sizeof(data_t)));
50.     data_t *db = pthread_getspecific(key);
51.
52.     db->INT_TYPE = param;
53.     db->TID_TYPE = pthread_self();
54.
55. /* будем помещать разные данные в разные потоки -
56.    например, поделим потоки на четные/нечетные
57. */
58.     if (param & 1) {
59.         db->STRING_TYPE = "Odd thread";
60.         db->FLOAT_TYPE = 1.01;
61.         db->BOOL_TYPE = true;
62.     } else {
63.         db->STRING_TYPE = "Even thread";
64.         db->FLOAT_TYPE = 2.22;
65.         db->BOOL_TYPE = false;
66.     }
67.     put_msg();
68.     return NULL;
69. }
70.
71. int main( int argc, char **argv, char **envp ) {
72.     pthread_t threads[NUM_THREADS];
73.     for (int i = 0; i < NUM_THREADS; i++) {
74.         // создаем поток и передаем в него его номер i
75.         pthread_create(&threads[i], NULL, thread_proc,
    (void*)(intptr_t)(i));
76.     }
77.     for (int i = 0; i < NUM_THREADS; i++) {
78.         //ждем завершения потоков
79.         pthread_join(threads[i], NULL);
80.     }
81.     return(EXIT_SUCCESS);
82. }
83.
```