Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу
«Операционные системы»

Тема работы
“Потоки”

Студент: Андреев Александр Олегович
Группа: М8О-206Б-20
Вариант: 2
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Постановка задачи

Задача: реализовать параллельный алгоритм быстрой сортировки.

Исходный код

l3.c

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>
unsigned current_number, number;
pthread_t* threads;
pthread_mutex_t mutex;

typedef struct
{
   unsigned i, j;
   int *array;
} thread_data;

void* quicksort (void* arguments)
{
   thread_data* data = (thread_data*) arguments;
   unsigned i = data->i, j = data->j;
   int x = data->array[(i + j) / 2];
   while (i <= j)
   {
      while (data->array[i] < x)
      {
         i++;
      }
      while (data->array[j] > x)
      {
         j--;
      }
      if (i <= j)
      {
         if (data->array[i] > data->array[j])
         {
            data->array[i] += data->array[j];
            data->array[j] = data->array[i] - data->array[j];
            data->array[i] -= data->array[j];
```

```
            }
            if (i == INT_MAX)
            {
               break;
            }
            i++;
            if (!j)
            {
               break;
            }
            j--;
         }
      }
      if ((i < data->j) && (data->i < j))
      {
         thread_data a = {i, data->j, data->array}, b = {data->i, j, data->array};
         if (current_number){
                int index;
            if (index = (pthread_mutex_lock(&mutex)) != 0){
               printf("There is some problems with locking mutex\n");
                        printf("Code of error is %d\n", index);
                        exit(1);
                }
            current_number--;
            if (index = (pthread_mutex_unlock(&mutex)) != 0)
              {
                  printf("There is some priblems with unlocking mutex\n");
                  printf("Code of error is %d\n", index);
                  exit(1);
            }
            int local_number = current_number;
            if ((data->j - i) >= (j - data->i))
            {
               if (index = (pthread_create(&threads[number - current_number - 1],
NULL, quicksort, &b)) != 0)
               {
                  printf("Can't create the thread\n");
                  printf("Code of error is %d\n", index);
                  exit(1);
               }
               quicksort(&a);
            }
```

```
            else
            {
               if (index = (pthread_create(&threads[number - current_number - 1],
NULL, quicksort, &a)) != 0)
               {
                  printf("Can't create the thread\n");
                  printf("Code of error is %d\n", index);
                  exit(1);
               }
               quicksort(&b);
            }
            if ((index = pthread_join(threads[number - local_number - 1], NULL)) !=
0)
            {
               printf("Can't join the thread\n");
               printf("Code of error is %d\n", index);
               exit(1);
            }
         }
         else
         {
            quicksort(&a);
            quicksort(&b);
         }
      }
      else
      {
         if (i < data->j)
         {
            thread_data a = {i, data->j, data->array};
            quicksort(&a);
         }
         else if (data->i < j)
         {
            thread_data a = {data->i, j, data->array};
            quicksort(&a);
         }
      }
      return NULL;
}

int main(int argc, char* argv[])
```

```
{
   if ((argc != 3) || (atoi(argv[1]) < 0) || (atoi(argv[2]) < 1))
   {
      printf("Syntax should be like this: ./[executable_file_name] [(non-negative)
number_of_threads] [(positive) size_of_array]\n");
      exit(1);
   }
   number = strtol(argv[1], NULL, 10);
   unsigned size = strtol(argv[2], NULL, 10);
   if (number > size)
   {
      printf("The size of array is less than number of threads, but it can't be with
parallel quick sort, so number of threads equals size of array now\n");
      number = size;
   }
   current_number = number;
   printf("Input elements of the array\n");
   int *array = (int*) malloc(size * sizeof(int));
   bool sorted = true;
   for (int i = 0; i < size; i++)
   {
      scanf("%i", &array[i]);
      if ((i) && (sorted) && (array[i] < array[i - 1]))
      {
         sorted = false;
      }
   }
   if (sorted)
   {
      printf("Array is sorted yet, this is the end of the program\n");
      for (int i = 0; i < size; i++)
      {
         printf("%i ", array[i]);
      }
      printf("\n");
        free(array);
      return 0;
   }
   int index;
   if (index = (pthread_mutex_init(&mutex, NULL)) != 0)
   {
      printf("There is some problems with initializing mutex\n");
```

```
        printf("%i\n", index);
        free(array);
        exit(1);
    }
    threads = (pthread_t*)malloc(number * sizeof(pthread_t));
    thread_data a = {0, size - 1, array};
    quicksort(&a);
    printf("Sorted array:\n");
    for (int i = 0; i < size; i++)
    {
        printf("%i ", array[i]);
    }
    printf("\n");
    free(array);
    free(threads);
    if (index = (pthread_mutex_destroy(&mutex)) != 0)
    {
        printf("There is some problems with destroying mutex\n");
        printf("%i\n", index);
        exit(1);
    }
    return 0;
}
```

Демонстрация работы программы

Тест 1.

Неправильные аргументы вызова

missclick3@missclick3:~/Desktop/OSLabs/lab3$ ./a.out -1 -2

Syntax should be like this: ./[executable_file_name] [(non-negative) number_of_threads] [(positive) size_of_array]

Тест 2.

Количество нитей больше количества элементов в массиве

missclick3@missclick3:~/Desktop/OSLabs/lab3$ ./a.out 8 6

The size of array is less than number of threads, but it can't be with parallel quick sort, so number of threads equals size of array now

Input elements of the array

88 12 -182 200 123 7

Sorted array:

-182 7 12 88 123 200

Тест 3.

Массив уже отсортирован

missclick3@missclick3:~/Desktop/OSLabs/lab3$ ./a.out 3 6

Input elements of the array

1 2 3 4 5 6

Array is sorted already, this is the end of the program

1 2 3 4 5 6

Тест 4.

Программа параллельно сортирует массив из 6элементов с помощью 3 потоков.

missclick3@missclick3:~/Desktop/OSLabs/lab3$ ./a.out 3 6

Input elements of the array

-100 12 7 0 9 6

Sorted array:

-100 0 6 7 9 12

Выводы

Благодаря данной лабораторной работе я ознакомился с тем, что из себя представляют потоки в операционной системе Ubuntu. Я узнал некоторые полезные системные вызовы, научился основам пользования мьютексом, а не очень сложное задание лишь помогло мне в этом.