Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

Лабораторная работа №2 по курсу “Операционные системы”

Студент: Иванов Андрей Кириллович

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 4

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1 Репозиторий

https://github.com/kumaroid/osLabs

2 Цель работы

Приобретение практических навыков в:

- Управлении потоками в ОС
- Обеспечении синхронизации между потоками

3 Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.

4 Описание работы программы

Отсортировать массив целых чисел при помощи TimSort

В ходе выполнения лабораторной работы я использовал следующие системные вызовы:

- pthread_create() - создание потока
- pthread_join() - ожидание завершения потока

5 Исходный код

timsort.hpp

```
#pragma once
#include <bits/stdc++.h>

void timSort(int *arr, int n, int maxThreads);
void printArray(int arr[], int n);
```

timsort.cpp

```
#include "timsort.hpp"
#include <cmath>

const int RUN = 4;

struct Args1 {
    int* arr;
    int left;
    int right;
};

struct Args2 {
    int* arr;
    int l;
    int m;
    int r;
};

void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d  ", arr[i]);
    printf("\n");
}

void* insertionSort(void* args)
{
    const auto &ar = *(Args1*)args;
    int *arr = ar.arr;
    int left = ar.left;
    int right = ar.right;
    for (int i = left + 1; i <= right; i++) {
        int temp = arr[i];
        int j = i - 1;
        while (j >= left && arr[j] > temp) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }

    // sleep(50);
    pthread_exit(0);
}

void* merge(void* args)
{
    Args2 arg = *(Args2*)args;
    int* arr = arg.arr;
```

```
50      int m = arg.m;
51      int r = arg.r;
52      int l = arg.l;
53
54      int len1 = m - l + 1, len2 = r - m;
55      int left[len1], right[len2];
56      for (int i = 0; i < len1; i++)
57          left[i] = arr[l + i];
58      for (int i = 0; i < len2; i++)
59          right[i] = arr[m + 1 + i];
60
61      int i = 0;
62      int j = 0;
63      int k = l;
64
65      while (i < len1 && j < len2) {
66          if (left[i] <= right[j]) {
67              arr[k] = left[i];
68              i++;
69          }
70          else {
71              arr[k] = right[j];
72              j++;
73          }
74          k++;
75      }
76
77      while (i < len1) {
78          arr[k] = left[i];
79          k++;
80          i++;
81      }
82
83      while (j < len2) {
84          arr[k] = right[j];
85          k++;
86          j++;
87      }
88
89      // sleep(50);
90      pthread_exit(0);
91  }
92
93  void timSort(int *arr, int n, int maxThreads)
94  {
95
96      int threadCount = std::min(n / RUN + 1, maxThreads);
97      if (maxThreads == -1) {
98          threadCount = n / RUN + 1;
99      }
100     pthread_t tid[threadCount];
101     int j = 0;
102     for (int i = 0; i < n; i += RUN)
103     {
104         Args1* arg = new Args1;
105         arg->arr = arr;
106         arg->left = i;
107         arg->right = std::min((i + RUN - 1), (n - 1));
108         // insertionSort((void*)(&arg));
```

```
109         if (j < threadCount) {
110             pthread_create(&tid[j++], nullptr, insertionSort, (
    void*)(arg));
111         } else {
112             while (j > 0) {
113                 pthread_join(tid[--j], NULL);
114             }
115             ++j;
116             pthread_create(&tid[j++], nullptr, insertionSort, (
    void*)(arg));
117         }
118     }
119 
120     while (j > 0) {
121         pthread_join(tid[--j], NULL);
122     }
123 
124     std::vector<pthread_t> tid2(threadCount);
125     j = 0;
126     for (int size = RUN; size < n; size = 2 * size) {
127         for (int left = 0; left < n; left += 2 * size) {
128             int mid = left + size - 1;
129             int right = std::min((left + 2 * size - 1), (n - 1));
130             if (mid < right) {
131                 Args2* arg = new Args2;
132                 arg->arr = arr;
133                 arg->l = left;
134                 arg->m = mid;
135                 arg->r = right;
136                 // merge((void*)arg);
137                 if (j < threadCount) {
138                     pthread_create(&tid2[j++], nullptr, merge, (
    void*)(arg));
139                 } else {
140                     while (j > 0) {
141                         pthread_join(tid2[--j], NULL);
142                     }
143                     pthread_create(&tid2[j++], nullptr, merge, (
    void*)(arg));
144                 }
145             }
146         }
147     }
148 
149     while (j > 0) {
150         pthread_join(tid2[--j], NULL);
151     }
152 }
```

6 Тесты

```cpp
#include <iostream>
#include <algorithm>
#include <gtest/gtest.h>

#include "timsort.hpp"


void TestParent(int* base_arr) {
    int n = sizeof(base_arr) / sizeof(base_arr[0]);
    int arr_1[n], arr_2[n];
    std::copy(base_arr, base_arr + n, arr_1);
    std::copy(base_arr, base_arr + n, arr_2);

    timSort(arr_1, n, 5);
    std::sort(arr_2, arr_2 + n);

    for (int i = 0; i < n; ++i) {
        ASSERT_EQ(arr_1[i], arr_2[i]);
    }
}

TEST(SortTest, ONE) {
    int arr[] = { -2, 7, 15, -14, 0, 15, 0, 7, 234, 3, -89, 9, -12, 21,
                      -7, -4, -13, 5, 8, -14, 12, 54, 1, 0, 127, 40, 55};
    TestParent(arr);
}

TEST(SortTest, TWO) {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                      17, 18, 19, 20, 21, 22, 23, 24, 25, 26,};
    TestParent(arr);
}

TEST(SortTest, THREE) {
    int arr[] = {1, 2, -34, 45, -66, 2, 96, -9, 33, 1, 1, 90, 44, -34, -3, 4, 35, 0};
    TestParent(arr);
}
```

7 Запуск тестов

```
andrew@DESKTOP-K3DH39N:~/MAI/osLabs/build/tests$ ./lab2_test
Running main() from /home/andrew/MAI/osLabs/build/_deps/googletest-src/googletest/s
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from SortTest
[ RUN      ] SortTest.ONE
[       OK ] SortTest.ONE (150 ms)
[ RUN      ] SortTest.TWO
[       OK ] SortTest.TWO (0 ms)
[ RUN      ] SortTest.THREE
[       OK ] SortTest.THREE (0 ms)
[----------] 3 tests from SortTest (150 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (187 ms total)
[  PASSED  ] 3 tests.
```

8 Выводы

В результате выполнения данной лабораторной работы была написана программа на языке C++ для сортировки массива методом TimSort, обрабатывающая данные в многопоточном режиме. Были получены практические навыки в управлении потоками в ОС и обеспечении синхронизации между потоками.