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: Слетюрин Кирилл Сергеевич

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 5

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1 Репозиторий

https://github.com/kirill483/OS

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

Нужно было отсортировать массив с помощью четно-нечетной сортировки Бетчера

В ходе выполнения лабораторной работы я использовала следующие системные вызовы:

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

5 Исходный код

lab2.cpp

```
#include <lab2.h>

void cmp(int* b, int* c){
    int tmp;
    if(*b > *c){
        tmp = *b;
        *b = *c;
        *c = tmp;
    }
}

void shuffle(void* args){
    someargs* arg = (someargs*) args;
    int m = (- arg->l + arg->r + 1) / 2;
    for(int i = arg->l, j = arg->l; i < arg->r || j + m <= arg->r; i += 2, j++){
        arg->tmp->d[j] = arg->v->d[i];
        arg->tmp->d[j + m] = arg->v->d[i + 1];
    }
    for(int i = arg->l, j = arg->l; i <= arg->r || j <= arg->r; i++, j++){
        arg->v->d[i] = arg->tmp->d[j];
    }
}

void unshuffle(void* args){
    someargs* arg = (someargs*) args;
    int m = (- arg->l + arg->r + 1) / 2;
    for(int i = arg->l, j = arg->l; i < arg->l + m || j + 1 <= arg->r; j += 2, i++){
        arg->tmp->d[j] = arg->v->d[i];
        arg->tmp->d[j + 1] = arg->v->d[i + m];
    }
    for(int i = arg->l, j = arg->l; i <= arg->r || j <= arg->r; i++, j++){
        arg->v->d[i] = arg->tmp->d[j];
    }
}

void* sort_2(void* args){
    someargs* arg = (someargs*) args;
    if(arg->l == arg->r){
        return  NULL;
    }
    someargs u = (someargs){.v = arg->v, .l = arg->l, .r = arg->r, .tmp = arg->tmp};
    shuffle((void*) &u);
    int m = (arg->l + arg->r + 1) / 2;
    someargs u1 = {.v = arg->v, .l = arg->l, .r = m - 1, .tmp = arg->tmp};
    sort_2((void*) &u1);
    someargs u2 = (someargs){.v = arg->v, .l = m, .r = arg->r, .tmp = arg->tmp};
    sort_2((void*) &u2);
    unshuffle((void*) &u);
    for(int i = arg->l ; i < arg->r; i++){
```

```
50          cmp(&arg->v->d[i], &arg->v->d[i + 1]);
51      }
52  }
53
54  void sort_1(vector* v, int maxcount){
55      int bi = bi_vector(v);
56      vector tmp;
57      init_vector(&tmp, lenth_vector(v));
58      time_t begin = time(NULL);
59      pthread_t threads[maxcount];
60      someargs args[maxcount];
61      int i;
62      for(i = 2; maxcount > 0; i *= 2){
63          maxcount -= i;
64      }
65      int j0 = lenth_vector(v)/(i/2) * 2 - 1;
66      int i0 = (j0 + 1) / 2 * -maxcount;
67      int z = -1;
68      int z0 = 0;
69      bool fl = false;
70      for(int j = 1; j <= lenth_vector(v); j = j * 2 + 1){
71          for(int i = 0; i <= lenth_vector(v) - j - 1; i = 1 + j + i
    ){
72              if(j >= j0 && i >= i0 && fl == false){
73                  fl =  true;
74                  i0 = i;
75              }
76              if(fl){
77                  someargs u = (someargs){.v = v, .l = i, .r = i + j
    , .tmp = &tmp};
78                  shuffle((void*)&u);
79                  int m = (i + i + j + 1) / 2;
80                  z++;
81                  args[z] = (someargs){.v = v, .l = i, .r = m - 1, .
    tmp = &tmp};
82                  pthread_create(&threads[z], NULL, sort_2, (void*)
    &args[z]);
83                  z++;
84                  args[z] = (someargs){.v = v, .l = m, .r = i + j, .
    tmp = &tmp};
85                  pthread_create(&threads[z], NULL, sort_2, (void*)
    &args[z]);
86              }
87              if(!fl){
88                  someargs u = (someargs){.v = v, .l = i, .r = i + j
    , .tmp = &tmp};
89                  sort_2((void*) &u);
90              }
91          }
92          if(fl){
93              for(int i = z0; i <= z; i++){
94                  pthread_join(threads[i], NULL);
95              }
96              z0 = z + 1;
97              for(int i = i0; i <= lenth_vector(v) - j - 1; i = 1 +
    j + i){
98                  someargs u = (someargs){.v = v, .l = i, .r = i + j
    , .tmp = &tmp};
99                  unshuffle((void*) &u);
```

```
                for(int k = i ; k < i + j; k++){
                    cmp(&v->d[k], &v->d[k + 1]);
                }
            }
            i0 = 0;
        }
    }
    debi_vector(v, bi);
    time_t end = time(NULL);
   // printf("Time %ld\n",end - begin);
    free(tmp.d);
}
```

utils.cpp

```
#include "utils.h"

void init_vector(vector* v, int k){
    v->d = (int*) malloc(sizeof(int) * k);
    v->size = 0;
    v->capacity = k;
}

void vector_resize(vector* v){
    v->capacity *= 2;
    v->d = (int*) realloc(v->d, sizeof(int) * v->capacity);
}

void push_vector(vector* v, int n){
    if(v->capacity == v->size){
        vector_resize(v);
    }
    v->d[v->size] = n;
    v->size++;
}

void pop_vector(vector* v){
    v->size--;
}

int lenth_vector(vector* v){
    return v->size;
}

void printf_vector(vector* v){
    for(int i = 0; i < lenth_vector(v); i++){
        printf("%d ", v->d[i]);
    }
    printf("\n");
}

int bi_vector(vector* v){
    float n = lenth_vector(v);
    int k = 0;
    while(n > 1){
        n /= 2;
        k++;
    }
    k = pow(2, k) - lenth_vector(v);
    for(int i = 0; i < k; i++){
```

```
        push_vector(v, INT_MAX);
    }
    return k;
}

void debi_vector(vector* v, int k){
    for(int i = 0; i < k; i++){
        pop_vector(v);
    }
}
```

main.cpp

```
#include "utils.h"
#include "lab2.h"

int main(){
    vector v;
    int maxcount;
    scanf("%d", &maxcount);
    init_vector(&v, 4);
    push_vector(&v, 1);
    push_vector(&v, -1);
    push_vector(&v, 10);
    push_vector(&v, 0);
    sort_1(&v, maxcount);
    printf_vector(&v);
    free(v.d);
}
```

6 Тесты

```
#include <gtest/gtest.h>
#include <utils.h>
#include <lab2.h>

bool check(int maxcount, int size){
    vector v;
    init_vector(&v, size);
    for(int i = 0; i < size; i++){
        push_vector(&v, rand() % 100);
    }
    sort_1(&v, maxcount);
    for(int i = 0; i < v.size - 1; i++){
        if(v.d[i] > v.d[i + 1]){
            free(v.d);
            return false;
        }
    }
    free(v.d);
    return true;
}

TEST(t_0_pthreads_100_size, test1){
    ASSERT_TRUE(check(0, 100));
}

TEST(t_2_pthreads_10_size, test2){
    ASSERT_TRUE(check(2, 10));
}

TEST(t_4_pthreads_1000_size, test3){
    ASSERT_TRUE(check(4, 1000));
}

TEST(t_8_pthreads_100_size, test4){
    ASSERT_TRUE(check(8, 10000));
}

TEST(t_16_pthreads_100_size, test5){
    ASSERT_TRUE(check(16, 100000));
}

TEST(t_1_pthreads_1_size, test6){
    ASSERT_TRUE(check(1, 1));
}

TEST(t_12_pthreads_56473_size, test7){
    ASSERT_TRUE(check(0, 56470));
}

int main(int argc, char **argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

7 Демонстрация работы программы

```
kirill@DESKTOP-7E05ERB:/mnt/c/Users/Kirill/OS/lab2/build$ ./main
6
-1 0 1 10
```

8 Запуск тестов

```
kirill@DESKTOP-7E05ERB:/mnt/c/Users/Kirill/OS/lab2/build$ ./Test
[==========] Running 7 tests from 7 test suites.
[----------] Global test environment set-up.
[----------] 1 test from t_0_pthreads_100_size
[ RUN      ] t_0_pthreads_100_size.test1
[       OK ] t_0_pthreads_100_size.test1 (0 ms)
[----------] 1 test from t_0_pthreads_100_size (0 ms total)

[----------] 1 test from t_2_pthreads_10_size
[ RUN      ] t_2_pthreads_10_size.test2
[       OK ] t_2_pthreads_10_size.test2 (0 ms)
[----------] 1 test from t_2_pthreads_10_size (0 ms total)

[----------] 1 test from t_4_pthreads_1000_size
[ RUN      ] t_4_pthreads_1000_size.test3
[       OK ] t_4_pthreads_1000_size.test3 (1 ms)
[----------] 1 test from t_4_pthreads_1000_size (1 ms total)

[----------] 1 test from t_8_pthreads_100_size
[ RUN      ] t_8_pthreads_100_size.test4
[       OK ] t_8_pthreads_100_size.test4 (21 ms)
[----------] 1 test from t_8_pthreads_100_size (21 ms total)

[----------] 1 test from t_16_pthreads_100_size
[ RUN      ] t_16_pthreads_100_size.test5
[       OK ] t_16_pthreads_100_size.test5 (205 ms)
[----------] 1 test from t_16_pthreads_100_size (205 ms total)

[----------] 1 test from t_1_pthreads_1_size
[ RUN      ] t_1_pthreads_1_size.test6
[       OK ] t_1_pthreads_1_size.test6 (0 ms)
[----------] 1 test from t_1_pthreads_1_size (0 ms total)

[----------] 1 test from t_12_pthreads_56473_size
[ RUN      ] t_12_pthreads_56473_size.test7
[       OK ] t_12_pthreads_56473_size.test7 (116 ms)
[----------] 1 test from t_12_pthreads_56473_size (116 ms total)

[----------] Global test environment tear-down
[==========] 7 tests from 7 test suites ran. (343 ms total)
[  PASSED  ] 7 tests.
```

9 Выводы

В результате выполнения данной лабораторной работы была написана программа на языке C для сортировки массива с помощью четно-нечетной сортировки Бетчера. Приобретены практические навыки в управлении потоками в ОС и обеспечении синхронизации между потоками.