Отчет по лабораторной работе № 3 по курсу
Операционные системы

Студент группы М8О-206Б-21 Синюков Антон Сергеевич, № по списку 19

Контакты www, e-mail, icq, skype vk.com/antonckya

Работа выполнена: « 11 » февраля 2023 г.

Преподаватель: Миронов Евгений

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 202 __ г., итоговая оценка ___

Подпись преподавателя _____

1. **Тема:** Управление потоками в ОС

2. **Цель работы:** Приобретение практических навыков в управление потоками в ОС и обеспечение синхронизации между потоками.

3. **Задание** (*вариант № 3*): Отсортировать массив целых чисел при помощи параллельной сортировки слиянием.

4. **Оборудование**:
ЭВМ _____, процессор _____, имя узла сети _____с ОП _____Мб,
НМД _____Мб. Терминал _____адрес _____. Принтер _____
Другие устройства _____

5. **Программное обеспечение**:
Операционная система семейства _____, наименование _____версия _____
интерпретатор команд _____версия _____
Система программирования _____версия _____
Редактор текстов _____версия _____
Утилиты операционной системы _____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных _____

6. **Идея, метод, алгоритм** решение задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Массив делится на количесво частей, равное количеству данных потоков. Каждый поток сортирует свою часть с помощью сортировки слиянием. На выходе получается N отсортированных кусков внутри массива, которые параллельно сливаются, получая отсортированный массив.

7. **Сценарий выполнения работы** (план работы, первоначальный текст программы в черновике [можно на отдельном листе] и тесты либо соображения по тестированию)

1. Изучить работу с потоками POSIX: создание и их использование в ОС Linux с помощью pthread.h
2. Написать программу main.c для выполнения поставленной задачи.

3. Скомпилировать и протестировать программу.

8. **Выводы:** : Ранее мне уже приходилось использовать потоки во время написания личных проектов и редких домашних заданий в школе. Но тогда я использовал язык программирования Python, в котором потоки создаются и используются довольно просто. Теперь я осознал, насколько сложные технологии и процессы скрываются за многопоточным программированием. Лабораторная работа получилась очень интересная, хоть и сложная.

Подпись студента _____

9. Код

main.c

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// gcc -o a.out main.c -lpthread

int array[1000000];

typedef struct {
    int left;
    int right;
} left_right;

typedef struct {
    int left;
    int mid;
    int right;
} left_mid_right;

void merge(int left, int mid, int right){
    //printf("merged: %d %d %d\n", left, mid, right);
    int it1 = left, it2 = mid + 1;
    int res[1000000];
    for (int i = left; i <= right; i++){
        res[i] = array[i];
    }
    for (int i = left; i <= right; i++){
        if (it1 > mid){
            array[i] = res[it2];
            it2++;
        } else if (it2 > right){
            array[i] = res[it1];
            it1++;
        } else if (res[it2] < res[it1]){
            array[i] = res[it2];
            it2++;
        } else {
            array[i] = res[it1];
            it1++;
        }
    }
}

void merge_sort(int left, int right){
    if (left >= right)
        return;
```

```
    int mid = (left + right) / 2;
    merge_sort(left, mid);
    merge_sort(mid + 1, right);
    merge(left, mid, right);
}

void * mt_merge_sort(void * args){
    left_right *lr = args;
    int left = lr->left;
    int right = lr->right;
    //printf("l: %d, r: %d\n", left, right);
    merge_sort(left, right);
    pthread_exit(NULL);
}

void * mt_merge(void * args){
    left_mid_right *lmr = args;
    int left = lmr->left;
    int right = lmr->right;
    int mid = lmr->mid;
    //printf("l: %d, m: %d, r: %d\n", left, mid, right);
    merge(left, mid, right);
    pthread_exit(NULL);
}

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

    srand(time(NULL));
    //printf("randomized array:\n");

    long array_size = atol(argv[1]);
    int thread_count = atoi(argv[2]);

    for (int i = 0; i < array_size; i++){
        array[i] = 1 + rand() % 1000;
    }
    /*
    for (int i = 0; i < array_size; i++){
        printf("%d ", array[i]);
    }
    printf("\n\n"); */

    pthread_t *threads = malloc(thread_count * sizeof(pthread_t));
    left_right *lr_threads = malloc(thread_count * sizeof(left_right));
    left_mid_right *lmr_threads = malloc(thread_count * sizeof(left_mid_right));

    for (int i = 0; i < thread_count; i++) {
        int left = i * (array_size / thread_count);
        int right = (i + 1) * (array_size / thread_count);
        if (i == thread_count - 1){
```

```
            right = array_size;
        }
        lr_threads[i].left = left;
        lr_threads[i].right = right - 1;
        pthread_create(&threads[i], NULL, mt_merge_sort, &(lr_threads[i]));
    }
    for (int i = 0; i < thread_count; i++) {
        pthread_join(threads[i], NULL);
    }

    /*
    for (int i = 0; i < thread_count; i++) {
        printf("%d %d\n", lr_threads[i].left, lr_threads[i].right);
    }
    printf("\nsemi-sorted array: \n");
    for (int i = 0; i < array_size; i++){
        printf("%d ", array[i]);
    }
    printf("\n");
    */

    while (thread_count != 1){
        if (thread_count == 2){
            break;
        }
        thread_count = thread_count / 2;
        for (int i = 0; i < thread_count; i++) {
            int left = lr_threads[2 * i].left;
            int right = lr_threads[2 * i + 1].right;
            int mid = lr_threads[2 * i].right;
            lr_threads[i].left = left;
            lr_threads[i].right = right;
            left_mid_right lmr;
            lmr_threads[i].left = left;
            lmr_threads[i].mid = mid;
            lmr_threads[i].right = right - 1;
            //printf("lmr: %d %d %d\n", lmr.left, lmr.mid, lmr.right);
            pthread_create(&threads[i], NULL, mt_merge, &(lmr_threads[i]));
        }
        for (int i = 0; i < thread_count; i++) {
            pthread_join(threads[i], NULL);
        }
    }

    merge(lr_threads[0].left, lr_threads[0].right, array_size - 1);

    free(threads);
    free(lr_threads);
    free(lmr_threads);
```

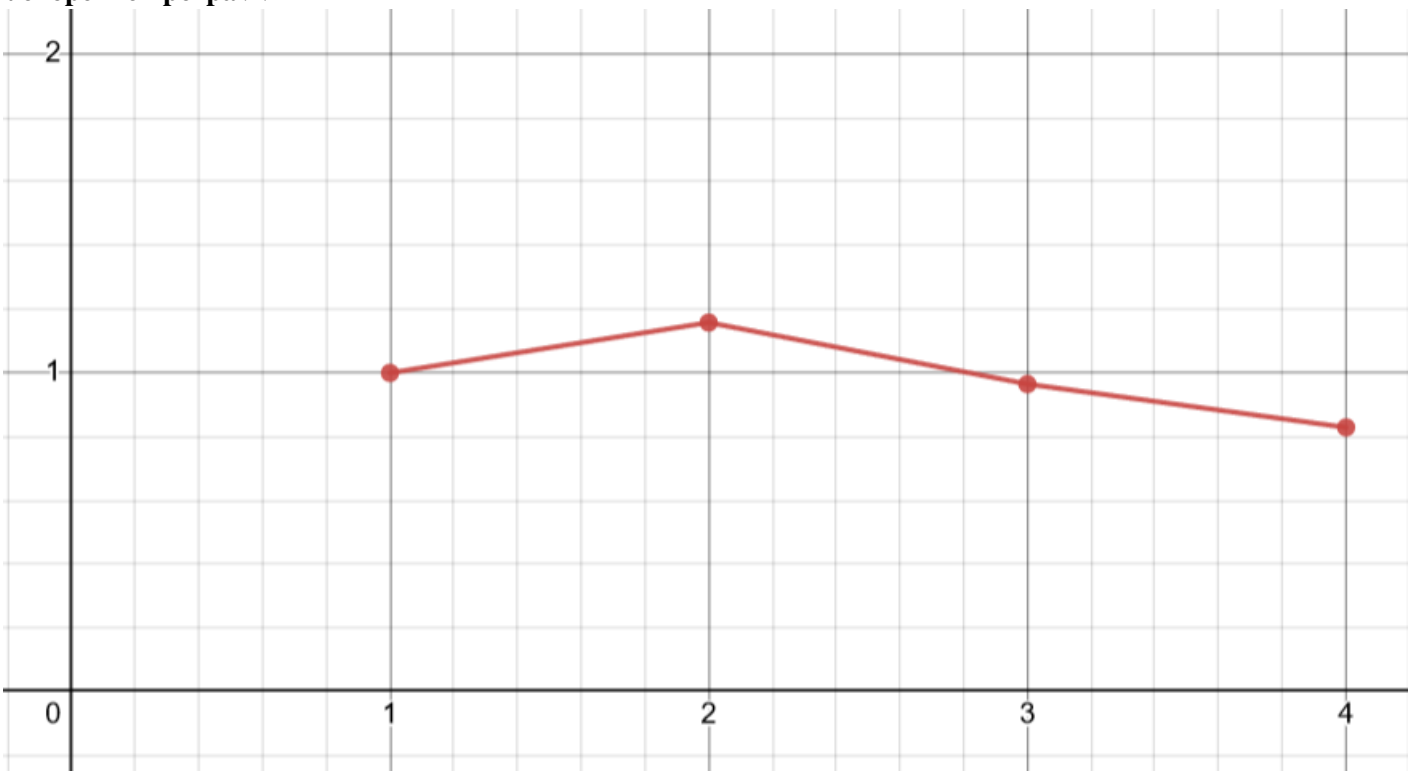
```
    /*
    printf("\nsorted array: \n");
    for (int i = 0; i < array_size; i++){
        if (array[i] != 0){
            printf("%d ", array[i]);
        }
    }
    printf("\n");
    */

    return 0;
}
```

10. **Протокол strace**

```
root@Anton-Sinyukov:/mnt/c/Users/sinyu/Desktop/os_3# strace ./a.out 10000 2
execve("./a.out", ["./a.out", "10000", "2"], 0x7ffd3b3bb300 /* 20 vars */) = 0
brk(NULL)                               = 0x56434697b000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe2ccf7660) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f65d0d28000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=16355, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16355, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f65d0d24000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f65d0afc000
mmap(0x7f65d0b24000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000
mmap(0x7f65d0cb9000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f65
mmap(0x7f65d0d11000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000
mmap(0x7f65d0d17000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f65d0af9000
arch_prctl(ARCH_SET_FS, 0x7f65d0af9740) = 0
set_tid_address(0x7f65d0af9a10)         = 722
set_robust_list(0x7f65d0af9a20, 24)     = 0
rseq(0x7f65d0afa0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f65d0d11000, 16384, PROT_READ) = 0
mprotect(0x5643450d1000, 4096, PROT_READ) = 0
mprotect(0x7f65d0d62000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f65d0d24000, 16355)           = 0
getrandom("\x5b\xcd\x39\x1f\xde\xee\xe0\xd4", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x56434697b000
brk(0x56434699c000)                     = 0x56434699c000
rt_sigaction(SIGRT_1, {sa_handler=0x7f65d0b8d8f0, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_R
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f65d02f8000
mprotect(0x7f65d02f9000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
ck=0x7f65d02f8000, stack_size=0x7fff00, tls=0x7f65d0af8640} => {parent_tid=[723]}, 88) = 723
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f65cfaf7000
mprotect(0x7f65cfaf8000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
ck=0x7f65cfaf7000, stack_size=0x7fff00, tls=0x7f65d02f7640} => {parent_tid=[724]}, 88) = 724
```

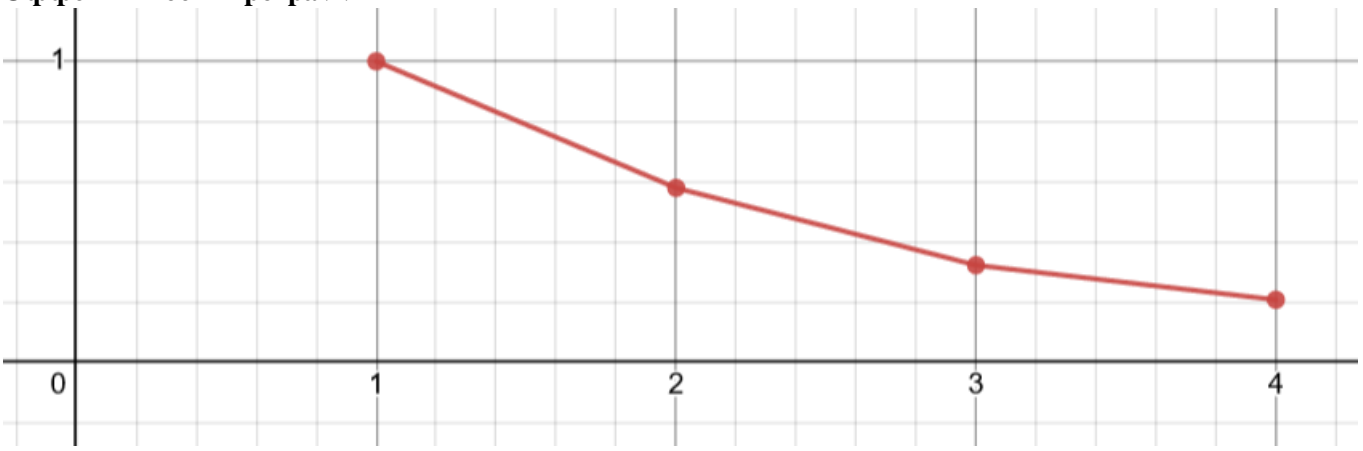
```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
ck=0x7f65cfaf7000, stack_size=0x7fff00, tls=0x7f65d02f7640} => {parent_tid=[724]}, 88) = 724
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
futex(0x7f65d0af8910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 723, NULL, FUTEX_BITSET_MATCH_ANY) =
futex(0x7f65d02f7910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 724, NULL, FUTEX_BITSET_MATCH_ANY) =
exit_group(0)                           = ?
+++ exited with 0 +++
```

11. Ускорение программы



По оси X - количество потоков, по оси Y - ускорение

12. Эффективность программы



По оси X - количество потоков, по оси Y - эффективность

Данные значения вызваны неэффективным вторым этапом программы - слиянием отсортированных частей. Вместо одного слияния всего массива один раз в конце количество слияний зависит от входного количества потоков как O(n), и чем их больше - тем больше будет слияний (при 2 потоках 1 слияние вызывает ускорение, а, например при 100 потоках их количество равно 99, при этом разбитых на 7 этапов с вызывами системных вызовов на инициализацию потоков).