Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №2 по курсу «Операционные системы»

Группа: М8О-213Б-23

Студент: Аксельрод А.М.

Преподаватель: Бахарев В.Д.

Оценка: ____________________

Дата: 24.12.24

Москва, 2024

Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке 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.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчёте привести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 4. Отсортировать массив целых чисел при помощи TimSort.

Общий метод и алгоритм решения

Использованные системные вызовы:

- int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr); – инициализация мьютекса.
- int pthread_mutex_lock(pthread_mutex_t *mutex); – блокировка мьютекса.
- int pthread_mutex_unlock(pthread_mutex_t *mutex); – разблокировка мьютекса.
- int pthread_mutex_destroy(pthread_mutex_t *mutex); – удаление мьютекса.
- ssize_t write(int_fd, const void* buf, size_t n); – записывает n байт из буфера в файл fd.
- int pthread_create (pthread_t *tid, const pthread_attr_t *tattr, void *(*start_routine)(void *), void *arg); – создаёт поток со стартовой функцией и с заданными аргументами.
- int pthread_join (pthread_t tid, void ** status); – блокирует вызывающий поток, пока указанный поток не завершится.
- void exit(int status); – завершает программу.

Программа запускается с аргументом — числом потоков, которое управляет количеством одновременно работающих потоков. В начале инициализируется мьютекс для синхронизации доступа к данным. Потом создаётся структура TASK, которая используется для передачи информации о диапазонах данных, с которыми будет работать каждый поток. Массив случайных чисел делится на несколько диапазонов, и каждому потоку передаются начальные и конечные индексы этих диапазонов. Количество элементов в каждом диапазоне определяется отношением числа элементов к числу потоков.

Если количество активных потоков достигает максимального значения, программа ожидает завершения всех текущих потоков перед запуском новых. После завершения всех вычислений выполняется ожидание завершения оставшихся запущенных потоков.

Время, затраченное на выполнение сортировки, измеряется с помощью функции get_time_in_milliseconds, которая использует gettimeofday для получения времени в миллисекундах. Время выполнения выводится на экран.

В конце мьютекс уничтожается.

Число логических ядер:

```
$ nproc --all
```

4

Число потоков	Время выполнения(мс)	Ускорение	Эффективность
1	26232	1.00	1.00
2	14729	1.78	0.89
3	12216	2.15	0.72
4	**8486**	**3.09**	0.77
8	9936	2.64	0.33
16	18315	1.43	0.09

Код программы

Main.c

```
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/time.h>

typedef struct TASK {
    int low;
    int high;
    int busy;
    int* a;
} TASK;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void insertion_sort(int* a, int left, int right) {
    for (int i = left + 1; i <= right; i++) {
        int temp = a[i];
        int j = i - 1;
        while (j >= left && a[j] > temp) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = temp;
    }
}

void merge(int* a, int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int* left_array = (int*)malloc(n1 * sizeof(int));
    int* right_array = (int*)malloc(n2 * sizeof(int));

    for (int i = 0; i < n1; i++) {
        left_array[i] = a[left + i];
    }
    for (int i = 0; i < n2; i++) {
        right_array[i] = a[mid + 1 + i];
    }

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {
        if (left_array[i] <= right_array[j]) {
            a[k++] = left_array[i++];
        } else {
            a[k++] = right_array[j++];
        }
    }

    while (i < n1) {
        a[k++] = left_array[i++];
    }

    while (j < n2) {
        a[k++] = right_array[j++];
    }

    free(left_array);
    free(right_array);
}

void timsort(int* a, int n) {
    int run = 32;
    for (int i = 0; i < n; i += run) {
        insertion_sort(a, i, (i + run - 1 < n - 1) ? i + run - 1 : n - 1);
    }
    for (int size = run; size < n; size = 2 * size) {
        for (int left = 0; left < n; left += 2 * size) {
            int mid = left + size - 1;
            int right = (left + 2 * size - 1 < n - 1) ? left + 2 * size - 1 : n - 1;

            if (mid < right) {
                merge(a, left, mid, right);
            }
        }
    }
}

void* timsort_thread(void* arg) {
    TASK* task = (TASK*)arg;
    timsort(task->a + task->low, task->high - task->low + 1);

    pthread_mutex_lock(&mutex);
    task->busy = 0;
    pthread_mutex_unlock(&mutex);

    return NULL;
}

long get_time_in_milliseconds() {
    struct timeval tv;
    gettimeofday(&tv, NULL);
    return tv.tv_sec * 1000 + tv.tv_usec / 1000;
}
```

```
void write_message(int fd, const char* message) {
    size_t len = 0;
    while (message[len] != '\0') {
        len++;
    }
    write(fd, message, len);
}

void write_number_message(int fd, const char* prefix, long number, const char*
suffix) {
    char buffer[128];
    char num_buffer[64];
    size_t prefix_len = 0, suffix_len = 0, num_len = 0;

    while (prefix[prefix_len] != '\0') {
        prefix_len++;
    }
    while (suffix[suffix_len] != '\0') {
        suffix_len++;
    }

    long temp = number;
    if (temp == 0) {
        num_buffer[num_len++] = '0';
    } else {
        while (temp > 0) {
            num_buffer[num_len++] = (temp % 10) + '0';
            temp /= 10;
        }
    }

    for (size_t i = 0; i < num_len / 2; i++) {
        char tmp = num_buffer[i];
        num_buffer[i] = num_buffer[num_len - 1 - i];
        num_buffer[num_len - 1 - i] = tmp;
    }

    size_t total_len = prefix_len + num_len + suffix_len;
    size_t pos = 0;

    for (size_t i = 0; i < prefix_len; i++) {
        buffer[pos++] = prefix[i];
    }
    for (size_t i = 0; i < num_len; i++) {
        buffer[pos++] = num_buffer[i];
    }
    for (size_t i = 0; i < suffix_len; i++) {
        buffer[pos++] = suffix[i];
    }

    write(fd, buffer, total_len);
}

int main(int argc, char** argv) {
    int max_array_elements = 100000000;
    int max_threads;

    if (argc < 2) {
        write_message(STDERR_FILENO, "usage: program thread_count\n");
        exit(EXIT_SUCCESS);
    }

    if (argc == 2) {
        max_threads = atoi(argv[1]);
    }

    time_t raw_time;

    write_number_message(STDERR_FILENO, "array[", max_array_elements,
"]\n");
    write_number_message(STDERR_FILENO, "threads[", max_threads, "]\n");

    int* array = (int*)malloc(sizeof(int) * max_array_elements);

    srand((unsigned)time(NULL));
    for (int i = 0; i < max_array_elements; i++)
        array[i] = rand();

    write_message(STDERR_FILENO, "array randomized\n");

    pthread_t* threads = (pthread_t*)malloc(sizeof(pthread_t) * max_threads);
    TASK* tasklist = (TASK*)malloc(sizeof(TASK) * max_threads);

    int len = max_array_elements / max_threads;
    int low = 0;

    long start_time_ms = get_time_in_milliseconds();
```

```
time(&raw_time);
write_message(STDERR_FILENO, "now time is: ");
write_message(STDERR_FILENO, ctime(&raw_time));

for (int i = 0; i < max_threads; i++, low += len) {
    TASK* task = &tasklist[i];
    task->a = array;
    task->busy = 1;
    task->low = low;
    task->high = (i == max_threads - 1) ? max_array_elements - 1 : low + len - 1;

    pthread_create(&threads[i], 0, timsort_thread, task);
}

for (int i = 0; i < max_threads; i++)
    pthread_join(threads[i], NULL);

pthread_mutex_lock(&mutex);
TASK* taskm = &tasklist[0];
for (int i = 1; i < max_threads; i++) {
    TASK* task = &tasklist[i];
    merge(taskm->a, taskm->low, task->low - 1, task->high);
}
pthread_mutex_unlock(&mutex);

long end_time_ms = get_time_in_milliseconds();

time(&raw_time);
write_message(STDERR_FILENO, "now time is: ");
```

```
    write_message(STDERR_FILENO, ctime(&raw_time));

    write_number_message(STDERR_FILENO, "array sorted in ", end_time_ms - start_time_ms, " ms\n");

    free(tasklist);
    free(threads);
    free(array);

    pthread_mutex_destroy(&mutex);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ gcc main.c
$ ./a.out
usage: program thread_count

$ ./a.out 1
array[100000000]
threads[1]
array randomized
now time is: Tue Dec 24 01:47:59 2024
now time is: Tue Dec 24 01:48:25 2024
array sorted in 26232 ms

$ ./a.out 2
array[100000000]
threads[2]
```

```
array randomized
now time is: Tue Dec 24 01:48:33 2024
now time is: Tue Dec 24 01:48:47 2024
array sorted in 14729 ms

$ ./a.out 3
array[100000000]
threads[3]
array randomized
now time is: Tue Dec 24 01:48:54 2024
now time is: Tue Dec 24 01:49:05 2024
array sorted in 12216 ms

$ ./a.out 4
array[100000000]
threads[4]
array randomized
now time is: Tue Dec 24 01:49:11 2024
now time is: Tue Dec 24 01:49:19 2024
array sorted in 8486 ms

$ ./a.out 8
array[100000000]
threads[8]
array randomized
now time is: Tue Dec 24 01:49:26 2024
now time is: Tue Dec 24 01:49:35 2024
array sorted in 9936 ms
```

```
$ ./a.out 16
array[100000000]
threads[16]
array randomized
now time is: Tue Dec 24 01:49:42 2024
now time is: Tue Dec 24 01:50:01 2024
array sorted in 18315 ms
```

Strace:

```
$ strace -f ./a.out 4
execve("./a.out", ["./a.out", "4"], 0x7ffe639ff830 /* 68 vars */) = 0
brk(NULL) = 0x5b0b498a0000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc843af880) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbf5609000
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=58791, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 58791, PROT_READ, MAP_PRIVATE, 3, 0) = 0x78cbf55fa000
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\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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
```

```
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x78cbf5200000

mprotect(0x78cbf5228000, 2023424, PROT_NONE) = 0

mmap(0x78cbf5228000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x78cbf5228000

mmap(0x78cbf53bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x78cbf53bd000

mmap(0x78cbf5416000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x78cbf5416000

mmap(0x78cbf541c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x78cbf541c000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbf55f7000

arch_prctl(ARCH_SET_FS, 0x78cbf55f7740) = 0

set_tid_address(0x78cbf55f7a10) = 13296

set_robust_list(0x78cbf55f7a20, 24) = 0

rseq(0x78cbf55f80e0, 0x20, 0, 0x53053053) = 0

mprotect(0x78cbf5416000, 16384, PROT_READ) = 0

mprotect(0x5b0b4984d000, 4096, PROT_READ) = 0

mprotect(0x78cbf5643000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x78cbf55fa000, 58791) = 0

write(2, "array[100000000]\n", 17array[100000000]) = 17

write(2, "threads[4]\n", 11threads[4]) = 11

getrandom("\x09\xb5\xc8\xe9\x57\x44\xb9\x0f", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x5b0b498a0000

brk(0x5b0b498c1000) = 0x5b0b498c1000
```

```
mmap(NULL, 400003072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbdd400000

write(2, "array randomized\n", 17array randomized) = 17

write(2, "now time is: ", 13now time is: ) = 13

openat(AT_FDCWD, "/etc/localtime", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=1535, ...}, AT_EMPTY_PATH) = 0

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=1535, ...}, AT_EMPTY_PATH) = 0

read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 1535

lseek(3, -927, SEEK_CUR) = 608

read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 927

close(3) = 0

write(2, "Tue Dec 24 01:50:33 2024\n", 25Tue Dec 24 01:50:33 2024) = 25

rt_sigaction(SIGRT_1, {sa_handler=0x78cbf5291870, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x78cbf5242520}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x78cbdca00000

mprotect(0x78cbdca01000, 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|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x78cbdd200910, parent_tid=0x78cbdd200910, exit_signal=0, stack=0x78cbdca00000, stack_size=0x7fff00, tls=0x78cbdd200640}strace: Process 13315 attached

=> {parent_tid=[13315]}, 88) = 13315

[pid 13315]

rseq(0x78cbdd200fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 13296] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 13315] <... rseq resumed>) = 0
```

[pid 13296] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 13315] set_robust_list(0x78cbdd200920, 24 <unfinished ...>

[pid 13296] <... mmap resumed>) = 0x78cbdc000000

[pid 13315] <... set_robust_list resumed>) = 0

[pid 13296] mprotect(0x78cbdc001000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

[pid 13315] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

[pid 13296] <... mprotect resumed>) = 0

[pid 13315] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 13296] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

[pid 13296] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x78cbdc800910, parent_tid=0x78cbdc800910, exit_signal=0, stack=0x78cbdc000000, stack_size=0x7fff00, tls=0x78cbdc800640}strace: Process 13316 attached

<unfinished ...>

[pid 13316] rseq(0x78cbdc800fe0, 0x20, 0, 0x53053053) = 0

[pid 13296] <... clone3 resumed> => {parent_tid=[13316]}, 88) = 13316

[pid 13316] set_robust_list(0x78cbdc800920, 24 <unfinished ...>

[pid 13296] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

[pid 13316] <... set_robust_list resumed>) = 0

[pid 13296] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 13316] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

[pid 13296] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 13316] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 13296] <... mmap resumed>) = 0x78cbdb600000

[pid 13296] mprotect(0x78cbdb601000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 13296] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

```
[pid 13296]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLO
NE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, child_tid=0x78cbdbe00910,
parent_tid=0x78cbdbe00910, exit_signal=0, stack=0x78cbdb600000,
stack_size=0x7fff00, tls=0x78cbdbe00640}strace: Process 13317 attached

<unfinished ...>

[pid 13317] rseq(0x78cbdbe00fe0, 0x20, 0, 0x53053053) = 0

[pid 13317] set_robust_list(0x78cbdbe00920, 24) = 0

[pid 13317] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 13296] <... clone3 resumed> => {parent_tid=[13317]}, 88) = 13317

[pid 13296] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 13296] mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x78cbdac00000

[pid 13296] mprotect(0x78cbdac01000, 8388608,
PROT_READ|PROT_WRITE) = 0

[pid 13296] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

[pid 13296]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLO
NE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, child_tid=0x78cbdb400910,
parent_tid=0x78cbdb400910, exit_signal=0, stack=0x78cbdac00000,
stack_size=0x7fff00, tls=0x78cbdb400640}strace: Process 13318 attached

<unfinished ...>

[pid 13318] rseq(0x78cbdb400fe0, 0x20, 0, 0x53053053) = 0

[pid 13318] set_robust_list(0x78cbdb400920, 24 <unfinished ...>

[pid 13296] <... clone3 resumed> => {parent_tid=[13318]}, 88) = 13318

[pid 13318] <... set_robust_list resumed>) = 0

[pid 13296] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

[pid 13318] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 13296] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 13296] futex(0x78cbdd200910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 13315, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
```

```
[pid 13317] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x78cbd2c00000

[pid 13317] munmap(0x78cbd2c00000, 20971520) = 0

[pid 13317] munmap(0x78cbd8000000, 46137344) = 0

[pid 13317] mprotect(0x78cbd4000000, 135168, PROT_READ|PROT_WRITE) = 0

[pid 13315] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x78cbcc000000

[pid 13315] munmap(0x78cbd0000000, 67108864) = 0

[pid 13315] mprotect(0x78cbcc000000, 135168, PROT_READ|PROT_WRITE) = 0

[pid 13318] mmap(0x78cbd0000000, 67108864, PROT_NONE,

MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0 <unfinished ...>

[pid 13316] mmap(0x78cbd0000000, 67108864, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0 <unfinished ...>

[pid 13318] <... mmap resumed>) = 0x78cbc8000000

[pid 13318] mprotect(0x78cbc8000000, 135168, PROT_READ|PROT_WRITE) = 0

[pid 13316] <... mmap resumed>) = 0x78cbc4000000

[pid 13316] mprotect(0x78cbc4000000, 135168, PROT_READ|PROT_WRITE) = 0

[pid 13317] mprotect(0x78cbd4021000, 4096, PROT_READ|PROT_WRITE) = 0

[pid 13318] mprotect(0x78cbc8021000, 4096, PROT_READ|PROT_WRITE) = 0

[pid 13315] mprotect(0x78cbcc021000, 4096, PROT_READ|PROT_WRITE) = 0

[pid 13316] mprotect(0x78cbc4021000, 4096, PROT_READ|PROT_WRITE) = 0

[pid 13317] mmap(NULL, 135168, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbf55d6000

[pid 13317] munmap(0x78cbf55d6000, 135168) = 0

[pid 13317] mprotect(0x78cbd4022000, 131072, PROT_READ|PROT_WRITE) = 0

[pid 13315] mprotect(0x78cbcc022000, 131072, PROT_READ|PROT_WRITE) = 0

[pid 13316] mprotect(0x78cbc4022000, 131072, PROT_READ|PROT_WRITE) = 0

[pid 13318] mprotect(0x78cbc8022000, 131072, PROT_READ|PROT_WRITE) = 0
```

```
        [pid 13317] mmap(NULL, 266240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbf55b6000

        [pid 13317] munmap(0x78cbf55b6000, 266240) = 0

        [pid 13317] mprotect(0x78cbd4042000, 262144, PROT_READ|PROT_WRITE) = 0

        [pid 13315] mprotect(0x78cbcc042000, 262144, PROT_READ|PROT_WRITE) = 0

        [pid 13318] mprotect(0x78cbc8042000, 262144, PROT_READ|PROT_WRITE
<unfinished ...>

        [pid 13317] mmap(NULL, 528384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

        [pid 13316] mprotect(0x78cbc4042000, 262144, PROT_READ|PROT_WRITE
<unfinished ...>

        [pid 13317] <... mmap resumed>) = 0x78cbf5576000

        [pid 13318] <... mprotect resumed>) = 0

        [pid 13317] munmap(0x78cbf5576000, 528384) = 0

        [pid 13317] mprotect(0x78cbd4082000, 524288, PROT_READ|PROT_WRITE) = 0

        [pid 13316] <... mprotect resumed>) = 0

        [pid 13315] mprotect(0x78cbcc082000, 524288, PROT_READ|PROT_WRITE) = 0

        [pid 13318] mprotect(0x78cbc8082000, 524288, PROT_READ|PROT_WRITE
<unfinished ...>

        [pid 13317] mmap(NULL, 1052672, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbf54f6000

        [pid 13318] <... mprotect resumed>) = 0

        [pid 13317] munmap(0x78cbf54f6000, 1052672) = 0

        [pid 13317] mprotect(0x78cbd4102000, 1048576, PROT_READ|PROT_WRITE) =
0

        [pid 13316] mprotect(0x78cbc4082000, 524288, PROT_READ|PROT_WRITE) = 0

        [pid 13315] mprotect(0x78cbcc102000, 1048576, PROT_READ|PROT_WRITE) =
0

        [pid 13317] mmap(NULL, 2101248, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbda800000

        [pid 13318] mprotect(0x78cbc8102000, 1048576, PROT_READ|PROT_WRITE) =
0

        [pid 13317] munmap(0x78cbda800000, 2101248) = 0
        [pid 13317] mprotect(0x78cbd4202000, 2097152, PROT_READ|PROT_WRITE) =
0

        [pid 13316] mprotect(0x78cbc4102000, 1048576, PROT_READ|PROT_WRITE) =
0

        [pid 13315] mprotect(0x78cbcc202000, 2097152, PROT_READ|PROT_WRITE) =
0

        [pid 13317] mmap(NULL, 4198400, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbda600000

        [pid 13318] mprotect(0x78cbc8202000, 2097152, PROT_READ|PROT_WRITE) =
0

        [pid 13317] munmap(0x78cbda600000, 4198400) = 0

        [pid 13317] mprotect(0x78cbd4402000, 4194304, PROT_READ|PROT_WRITE) =
0

        [pid 13316] mprotect(0x78cbc4202000, 2097152, PROT_READ|PROT_WRITE) =
0

        [pid 13315] mprotect(0x78cbcc402000, 4194304, PROT_READ|PROT_WRITE) =
0

        [pid 13317] mmap(NULL, 8392704, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbda200000

        [pid 13318] mprotect(0x78cbc8402000, 4194304, PROT_READ|PROT_WRITE) =
0

        [pid 13317] munmap(0x78cbda200000, 8392704) = 0

        [pid 13317] mprotect(0x78cbd4802000, 8388608, PROT_READ|PROT_WRITE) =
0

        [pid 13316] mprotect(0x78cbc4402000, 4194304, PROT_READ|PROT_WRITE) =
0

        [pid 13315] mprotect(0x78cbcc802000, 8388608, PROT_READ|PROT_WRITE) =
0

        [pid 13317] mmap(NULL, 16781312, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbd9a00000

        [pid 13318] mprotect(0x78cbc8802000, 8388608, PROT_READ|PROT_WRITE) =
0

        [pid 13317] munmap(0x78cbd9a00000, 16781312) = 0

        [pid 13317] mprotect(0x78cbd5002000,

        16777216, PROT_READ|PROT_WRITE) = 0
```

```
[pid 13316] mprotect(0x78cbc4802000, 8388608, PROT_READ|PROT_WRITE) =
0

[pid 13315] mprotect(0x78cbcd002000, 16777216, PROT_READ|PROT_WRITE) =
0

[pid 13317] mmap(NULL, 33558528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 13318] mprotect(0x78cbc9002000, 16777216, PROT_READ|PROT_WRITE) =
0

[pid 13317] <... mmap resumed>) = 0x78cbd8a00000

[pid 13316] mprotect(0x78cbc5002000, 16777216, PROT_READ|PROT_WRITE) =
0

[pid 13315] mmap(NULL, 33558528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbd1e00000

[pid 13318] mmap(NULL, 33558528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbc1e00000

[pid 13315] munmap(0x78cbd1e00000, 33558528) = 0

[pid 13315] mmap(NULL, 67112960, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbbdc00000

[pid 13316] mmap(NULL, 33558528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbd1e00000

[pid 13318] munmap(0x78cbc1e00000, 33558528 <unfinished ...>

[pid 13317] munmap(0x78cbd8a00000, 33558528) = 0

[pid 13318] <... munmap resumed>) = 0

[pid 13318] mmap(NULL, 67112960, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 13317] mmap(NULL, 67112960, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 13318] <... mmap resumed>) = 0x78cbb9a00000

[pid 13317] <... mmap resumed>) = 0x78cbb5800000

[pid 13316] munmap(0x78cbd1e00000, 33558528) = 0

[pid 13316] mmap(NULL, 67112960, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbb1600000

[pid 13315] munmap(0x78cbbdc00000, 67112960) = 0

[pid 13315] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
```

[pid 13315] madvise(0x78cbdca00000, 8368128, MADV_DONTNEED) = 0

[pid 13315] exit(0) = ?

[pid 13296] <... futex resumed>) = 0

[pid 13315] +++ exited with 0 +++

[pid 13296] futex(0x78cbdc800910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 13316, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 13318] munmap(0x78cbb9a00000, 67112960) = 0

[pid 13318] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 13318] madvise(0x78cbdac00000, 8368128, MADV_DONTNEED) = 0

[pid 13318] exit(0) = ?

[pid 13318] +++ exited with 0 +++

[pid 13317] munmap(0x78cbb5800000, 67112960) = 0

[pid 13317] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 13317] madvise(0x78cbdb600000, 8368128, MADV_DONTNEED) = 0

[pid 13317] exit(0) = ?

[pid 13317] +++ exited with 0 +++

[pid 13316] munmap(0x78cbb1600000, 67112960) = 0

[pid 13316] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 13316] madvise(0x78cbdc000000, 8368128, MADV_DONTNEED) = 0

[pid 13316] exit(0) = ?

[pid 13316] +++ exited with 0 +++

<... futex resumed>) = 0

mmap(NULL, 100003840, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbbe000000

mmap(NULL, 100003840, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbb8000000

munmap(0x78cbbe000000, 100003840) = 0

munmap(0x78cbb8000000, 100003840) = 0

mmap(NULL, 200003584, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbb8000000 mmap(NULL, 100003840, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbb2000000

munmap(0x78cbb8000000, 200003584) = 0

munmap(0x78cbb2000000, 100003840) = 0

mmap(NULL, 300003328, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbb2000000

mmap(NULL, 100003840, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78cbac000000

munmap(0x78cbb2000000, 300003328) = 0

munmap(0x78cbac000000, 100003840) = 0

write(2, "now time is: ", 13now time is:) = 13

newfstatat(AT_FDCWD, "/etc/localtime", {st_mode=S_IFREG|0644, st_size=1535, ...}, 0) = 0

write(2, "Tue Dec 24 01:50:43 2024\n", 25Tue Dec 24 01:50:43 2024) = 25

write(2, "array sorted in 10343 ms\n", 25array sorted in 10343 ms) = 25

munmap(0x78cbdd400000, 400003072) = 0

exit_group(0) = ?

+++ exited with 0 +++

Вывод

В ходе выполнения этой лабораторной работы я научилась писать многопоточные программы на Си. Я реализовала механизмы создания и синхронизации потоков при помощи библиотеки pthread.h. также я использовала мьютексы для синхронизации доступа к данным. В результате были собраны данные со временем сортировки массива с разным количеством потоков, в ходе их анализа было выявлено, что наибольшее ускорение – при количестве потоков, равном числу логических ядер.