

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-209Б-24

Студент: Осипов М.Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 18.11.25

Москва, 2025

Постановка задачи

Вариант 5.

Отсортировать массив целых чисел при помощи четно-нечетной сортировки Бетчера.

Общий метод и алгоритм решения

Использованные системные вызовы:

- **pthread_create()** - Создает новый поток, который выполняет функцию worker с передачей данных data
 - **pthread_join()** - Блокирует выполнение основного потока до завершения указанного рабочего потока
 - **pthread_mutex_lock() / pthread_mutex_unlock()** - Защищает критическую секцию (операцию обмена элементов) от одновременного доступа нескольких потоков
 - **malloc() / free()** - Динамическое выделение и освобождение памяти
 - **srand() / rand()** - Инициализация генератора случайных чисел и заполнение массива
 - **time()** - Используется для инициализации генератора случайных чисел
1. **Инициализация:** Программа создает массив случайных чисел заданного размера и инициализирует мьютекс для синхронизации
 2. **Запуск потоков:** Для каждой итерации сортировки создаются потоки - сначала для четной фазы (сравнение элементов 0-1, 2-3...), затем для нечетной фазы (сравнение элементов 1-2, 3-4...)
 3. **Параллельная обработка:** Каждый поток обрабатывает свою часть массива, сравнивая пары элементов и выполняя обмен при необходимости с использованием мьютекса
 4. **Синхронизация:** Основной поток ожидает завершения всех рабочих потоков после каждой фазы перед началом следующей
 5. **Проверка результата:** После каждой пары фаз проверяется отсортированность массива, процесс повторяется пока массив не будет полностью отсортирован
 6. **Завершение:** Программа выводит отсортированный массив и освобождает ресурсы

Код программы

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
```

```
#include <time.h>

int *array, size, max_threads;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

typedef struct {

    int phase;

    int thread_id;

} ThreadData;

void swap(int i, int j) {

    int tmp = array[i];

    array[i] = array[j];

    array[j] = tmp;

}

void* worker(void *arg) {

    ThreadData *data = (ThreadData*)arg;

    int start = data->thread_id * 2 + data->phase;

    int step = max_threads * 2;

    for (int i = start; i < size - 1; i += step) {

        if (array[i] > array[i + 1]) {

            pthread_mutex_lock(&mutex);

            swap(i, i + 1);

            pthread_mutex_unlock(&mutex);

        }

    }

}
```

```

free(data);

return NULL;
}

int is_sorted() {
    for (int i = 0; i < size - 1; i++)
        if (array[i] > array[i + 1]) return 0;
    return 1;
}

void print_array(int *arr, char *label) {
    printf("%s: ", label);
    for (int i = 0; i < size; i++) {
        printf("%d", arr[i]);
        if (i < size - 1) printf(" ");
    }
    printf("\n");
}

void parallel_sort() {
    pthread_t *threads = malloc(max_threads * sizeof(pthread_t));
    int sorted = 0;
    int iterations = 0;

    while (!sorted && iterations < size) {
        // Even phase
        for (int i = 0; i < max_threads; i++) {
            ThreadData *data = malloc(sizeof(ThreadData));
            data->phase = 0;
            data->thread_id = i;
        }
    }
}

```

```
pthread_create(&threads[i], NULL, worker, data);

}

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

// Odd phase

for (int i = 0; i < max_threads; i++) {

    ThreadData *data = malloc(sizeof(ThreadData));

    data->phase = 1;

    data->thread_id = i;

    pthread_create(&threads[i], NULL, worker, data);

}

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

sorted = is_sorted();

iterations++;

}

printf("Сортировка завершена за %d итераций\n", iterations);

free(threads);

}

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

    if (argc != 3) {

        printf("Usage: %s <size> <threads>\n", argv[0]);

        return 1;

    }

    size = atoi(argv[1]);

    max_threads = atoi(argv[2]);

    if (max_threads > size/2) max_threads = size/2;
```

```
if (max_threads < 1) max_threads = 1;

array = malloc(size * sizeof(int));
int *original = malloc(size * sizeof(int));

srand(time(NULL));

for (int i = 0; i < size; i++) {
    array[i] = rand() % 100;
    original[i] = array[i];
}

printf("== Чётно-нечётная сортировка Бетчера ==\n");
printf("Размер массива: %d, Потоков: %d\n\n", size, max_threads);

print_array(original, "Неотсортированный массив");

struct timespec start_time, end_time;
clock_gettime(CLOCK_MONOTONIC, &start_time);

parallel_sort();

clock_gettime(CLOCK_MONOTONIC, &end_time);

long long start_ns = start_time.tv_sec * 1000000000LL + start_time.tv_nsec;
long long end_ns = end_time.tv_sec * 1000000000LL + end_time.tv_nsec;
double execution_time_ms = (end_ns - start_ns) / 1000000.0;

print_array(array, "Отсортированный массив");

printf("Проверка: %s\n", is_sorted() ? "ОСОРТИРОВАНО КОРРЕКТНО" : "НЕ ОСОРТИРОВАНО КОРРЕКТНО");

printf("Время выполнения: %.3f мс\n", execution_time_ms);
```

```

        free(array);

        free(original);

        pthread_mutex_destroy(&mutex);

        return 0;

    }

```

Протокол работы программы

```

$ gcc -pthread main.c

$ ./a.out 10 4

==== Чётно-нечётная сортировка Бетчера ====

Размер массива: 10, Потоков: 4

Неотсортированный массив: 97 10 54 79 3 33 83 10 34 65

Сортировка завершена за 5 итераций

Отсортированный массив: 3 10 10 33 34 54 65 79 83 97

Проверка: ОТСОРТИРОВАНО КОРРЕКТНО

Время выполнения: 2.891 мс

```

Число потоков (размер массива = 50 чисел)	Время выполнения(мс)	Ускорение	Эффективность
1	9.452	1	1
2	10.786	0.88	0.440
4	30.121	0.31	0.078
8	66.414	0.14	0.018

```

$ strace ./a.out 10 4

execve("./a.out", ["../a.out", "10", "4"], 0x7ffe4048eb50 /* 34 vars */) = 0
brk(NULL)                      = 0x5b44a8f49000
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
fstat(3, {st_mode=S_IFREG|0644, st_size=21784, ...}) = 0
mmap(NULL, 21784, PROT_READ, MAP_PRIVATE, 3, 0) = 0x78866e522000

```



```
set_tid_address(0x78866e327a10)      = 55406
set_robust_list(0x78866e327a20, 24)   = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x78866e504690, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x78866e511140}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x78866e504730, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x78866e511140}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
brk(NULL)                      = 0x5b44a8f49000
brk(0x5b44a8f6a000)           = 0x5b44a8f6a000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "== \320\247\321\221\321\202\320\275\320\276-
\320\275\320\265\321\207\321\221\321\202\320\275\320\260\321\217 "..., 72== Чётно-нечётная
сортировка Бетчера ===
) = 72
write(1, "\320\240\320\260\320\267\320\274\320\265\321\200
\320\274\320\260\321\201\321\201\320\270\320\262\320\260: 10,"..., 52Размер массива: 10, Потоков: 4
) = 52
write(1,
"\320\235\320\265\320\276\321\202\321\201\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\2
60\320\275\320\275\321\213"... , 78Неотсортированный массив: 19 68 43 13 53 12 58 55 65 8
) = 78
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x78866db26000
mprotect(0x78866db27000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55407], tls=0x78866e326700, child_tidptr=0x78866e3269d0) =
55407
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x78866d325000
mprotect(0x78866d326000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) =
55408
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x78866cb24000
mprotect(0x78866cb25000, 8388608, PROT_READ|PROT_WRITE) = 0
```

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55409

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x78866c323000

mprotect(0x78866c324000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55410], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55410

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55411

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55412], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55412

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55413

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55414

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55415

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55416

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55417

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55418

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55419

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55420

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55421], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55421

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55422], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55422

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55423

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55424

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55425

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55426

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55427

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55428

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55429

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55430

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55431

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55432

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55433

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55434], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55434

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55435

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55436

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55437], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55437

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55438

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55439

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55440

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55441

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55442

clone(child_stack=0x78866cb22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[55443], tls=0x78866cb23700, child_tidptr=0x78866cb239d0) = 55443

clone(child_stack=0x78866d323fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866d324700, child_tidptr=0x78866d3249d0) = 55444

clone(child_stack=0x78866db24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866db25700, child_tidptr=0x78866db259d0) = 55445

clone(child_stack=0x78866e325fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x78866e326700, child_tidptr=0x78866e3269d0) = 55446

```
write(1, "\320\241\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\272\320\260
\320\267\320\260\320\262\320\265\321\200\321"..., 64Сортировка завершена за 5 итераций

) = 64

write(1,
"\320\236\321\202\321\201\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\260\320\275\320\2
75\321\213\320\271 \320"..., 74Отсортированный массив: 8 12 13 19 43 53 55 58 65 68

) = 74

write(1, "\320\237\321\200\320\276\320\262\320\265\321\200\320\272\320\260:
\320\236\320\242\320\241\320\236\320\240\320\242\320\230"..., 64Проверка: ОТСОРТИРОВАНО
КОРРЕКТНО

) = 64

exit_group(0)          = ?

+++ exited with 0 +++
```

Выход

Я составил и отладил программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними. В результате работы программы выполняет сортировку массива целых чисел при помощи четно-нечетной сортировки Бетчера. Для каждой итерации сортировки создаются потоки. Основной поток ожидает завершения всех рабочих потоков. Для маленьких массивов (50 элементов) обычно выгоднее использовать 1-2 потока. Параллельная сортировка начинает показывать преимущество на массивах от 1000+ элементов.