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-215Б-23

Студент: Закарейшвили Г. М.

Преподаватель: Миронов Е.С. (ПМИ)

Оценка: _____

Дата: 06.03.24

Москва, 2024

Постановка задачи

Вариант 5.

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. Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы. В отчете привести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.
5. Отсортировать массив целых чисел при помощи четно-нечетной сортировки Бетчера.

Общий метод и алгоритм решения

Использованные системные вызовы:

- **malloc** — выделяет память в куче для динамических массивов и структур.
- **free** — освобождает память, ранее выделенную с помощью malloc.
- **pthread_create** — создает новый поток и запускает выполнение функции в этом потоке.
- **pthread_join** — ожидает завершения выполнения указанного потока.
- **pthread_exit** — завершает выполнение текущего потока.
- **gettimeofday** — получает текущее время с точностью до микросекунд.
- **srand** — инициализирует генератор случайных чисел.
- **rand** — генерирует псевдослучайное число.
- **time** — возвращает текущее время в секундах (используется для инициализации генератора случайных чисел).
- **atoi** — преобразует строку в целое число (используется для преобразования аргумента командной строки в число потоков).

Алгоритм работы программы

1. Инициализация и ввод данных:
    - Ввод аргументов с командной строки. Максимальное количество потоков, которые будут использоваться для сортировки.
    - Ввод размера массива.
    - Выделение памяти для массива с помощью malloc.
    - Инициализация генератора случайных чисел с помощью функции srand и заполнение массива случайными числами с помощью функции rand.
2. Вывод исходного массива.
3. Измерение времени выполнения:
    - Записываем текущее время с помощью функции gettimeofday.
4. Многопоточная сортировка:
    - Вызов функции evenodd: создание потоков.
    - Сортировка в потоках. Их завершение pthread_exit.

● После создания всех потоков программа ожидает их завершения с помощью функции pthread_join.

Код программы

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>

#define MIN_MERGE 32
#define MIN(a, b) ((a) < (b) ? (a) : (b))

// данные в поток
typedef struct {
    int *array;
    int left;
    int right;
} ThreadData;


//слияние двух отсортированных подмассивов
void merge(int arr[], int l, int m, int r) {
    int len1 = m - l + 1, len2 = r - m;
    int *left = (int *)malloc(len1 * sizeof(int));
    int *right = (int *)malloc(len2 * sizeof(int));

    for (int i = 0; i < len1; i++)
        left[i] = arr[l + i];
    for (int i = 0; i < len2; i++)
        right[i] = arr[m + 1 + i];

    int i = 0, j = 0, k = l;
    while (i < len1 && j < len2) {
        if (left[i] <= right[j])
            arr[k++] = left[i++];
        else
            arr[k++] = right[j++];
    }

    while (i < len1)
        arr[k++] = left[i++];

    while (j < len2)
        arr[k++] = right[j++];
```

```
    free(left);
    free(right);
}

// моя сортировка по вики
void evenOddSort(int arr[], int left, int right) {
    int n = right - left + 1;
    for (int p = 1; p < n; p *= 2) {
        for (int k = p; k >= 1; k /= 2) {
            for (int j = k % p; j <= n - 1 - k; j += 2 * k) {
                int max_i = (k - 1 < n - j - k - 1) ? k - 1 : n - j - k - 1;
                for (int i = 0; i <= max_i; i++) {
                    int a = i + j;
                    int b = a + k;
                    if ((a / (p * 2)) == (b /  (p * 2))) {
                        if (arr[a + left] > arr[b + left]) {
                            int temp = arr[a + left];
                            arr[a + left] = arr[b + left];
                            arr[b + left] = temp;
                        }
                    }
                }
            }
        }
    }
}


void *evenoddThread(void *arg) {
    ThreadData *data = (ThreadData *)arg;
    int *arr = data->array;
    int left = data->left;
    int right = data->right;

    evenOddSort(arr, left, right);

    pthread_exit(NULL);
}

void evenodd(int arr[], int n, int maxThreads) {
    if (n < 2) return;

    pthread_t *threads = (pthread_t *)malloc(maxThreads * sizeof(pthread_t));
    ThreadData *threadData = (ThreadData *)malloc(maxThreads *
     sizeof(ThreadData));
```

```
    int chunkSize = n / maxThreads;
    for (int i = 0; i < maxThreads; i++) {
        int start = i * chunkSize;
        int end = MIN(start + chunkSize - 1, n - 1);
        if (start >= n) break;

        threadData[i].array = arr;
        threadData[i].left = start;
        threadData[i].right = end;

        pthread_create(&threads[i], NULL, evenoddThread, (void
     *)&threadData[i]);
    }

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

    // слияние частей массива
    for (int size = chunkSize; size < n; size = 2 * size) {
        for (int left = 0; left < n; left += 2 * size) {
            int mid = left + size - 1;
            int right = MIN(left + 2 * size - 1, n - 1);
            if (mid < right) {
                merge(arr, left, mid, right);
            }
        }
    }

    free(threads);
    free(threadData);
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Вы не ввели макс колво потоков\n", argv[0]);
        return 1;
    }

    int maxThreads = atoi(argv[1]);
    int arraySize;

    printf("Введите число элементов массива: ");
    scanf("%d", &arraySize);

    int *arr = (int *)malloc(arraySize * sizeof(int));
    // инициализация генератора случайных чисел
```

```
    srand(time(NULL));

    // генерация случайных чисел для массива
    for (int i = 0; i < arraySize; i++) {
        arr[i] = rand() % 100; // 0-99
    }

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

    struct timeval start, end;
    gettimeofday(&start, NULL);

    evenodd(arr, arraySize, maxThreads);

    gettimeofday(&end, NULL);

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

    long seconds = end.tv_sec - start.tv_sec;
    long microseconds = end.tv_usec - start.tv_usec;
    double elapsed = seconds + microseconds * 1e-6;

    printf("Заняло времени: %.6f секунд\n", elapsed);
    printf("Количество потоков: %d\n", maxThreads);

    free(arr);
    return 0;
}
```

Протокол работы программы

./evenodd 5
Введите число элементов массива: 8
Оригинал:
20 0 44 8 44 47 8 29
Отсортированный:
0 8 8 20 29 44 44 47
Заняло времени: 0.002625 секунд

Strace:

```
root@LAPTOP-CGCBKBHR:/home/OS/OS_labs/lab2/src# strace -f ./evenodd 2
execve("./evenodd", ["./evenodd", "2"], 0x7fffd37c6a20 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffdcbeb000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffe3e65ba0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd7c3590000
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=16555, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd7c354b000
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\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 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) = 0x7fd7c3320000
mprotect(0x7fd7c3348000, 2023424, PROT_NONE) = 0
mmap(0x7fd7c3348000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x7fd7c3348000
mmap(0x7fd7c34dd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7fd7c34dd000
mmap(0x7fd7c3536000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x7fd7c3536000
mmap(0x7fd7c353c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fd7c353c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd7c3310000
arch_prctl(ARCH_SET_FS, 0x7fd7c3310740) = 0
set_tid_address(0x7fd7c3310a10)         = 1845
set_robust_list(0x7fd7c3310a20, 24)     = 0
rseq(0x7fd7c33110e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
mprotect(0x7fd7c3536000, 16384, PROT_READ) = 0
mprotect(0x7fd7c3597000, 4096, PROT_READ) = 0
mprotect(0x7fd7c3588000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
munmap(0x7fd7c354b000, 16555)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x2), ...}, AT_EMPTY_PATH) = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
getrandom("\x3f\x95\x12\xdb\xcf\xdf\xe4\x50", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x7fffdcbeb000
brk(0x7fffdcc0c000)                     = 0x7fffdcc0c000
newfstatat(0, "", {st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x2), ...}, AT_EMPTY_PATH) = 0
ioctl(0, TCGETS, {B38400 opost isig icanon echo ...}) = 0
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\276
\321\215\320\273\320\265"..., 61Введите число элементов массива: ) = 61
read(0, 4
"4\n", 4096)                    = 2
time(NULL)                              = 1741239525 (2025-03-06T08:38:45+0300)
write(1, "\320\236\321\200\320\270\320\263\320\270\320\275\320\260\320\273:\n", 18Оригинал:
) = 18
write(1, "64 36 37 11 \n", 1364 36 37 11
)          = 13
gettimeofday({tv_sec=1741239525, tv_usec=916798}, NULL) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fd7c33b1870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fd7c3362520}, 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) = 0x7fd7c2b00000
mprotect(0x7fd7c2b01000, 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_SETT
LS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fd7c3300910, parent_tid=0x7fd7c3300910,
exit_signal=0, stack=0x7fd7c2b00000, stack_size=0x7fff00, tls=0x7fd7c3300640}, 88) = -1 ENOSYS (Function not implemented)
clone(child_stack=0x7fd7c32ffef0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLO
NE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 1846 attached
, parent_tid=[1846], tls=0x7fd7c3300640, child_tidptr=0x7fd7c3300910) = 1846
[pid  1846] set_robust_list(0x7fd7c3300920, 24 <unfinished ...>
[pid  1845] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1846] <... set_robust_list resumed>) = 0
[pid  1845] <... rt_sigprocmask resumed>NULL, 8) = 0
```

```
[pid  1846] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1845] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid  1846] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1845] <... mmap resumed>)         = 0x7fd7c22f0000
[pid  1846] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid  1845] mprotect(0x7fd7c22f1000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  1846] <... openat resumed>)       = 3
[pid  1845] <... mprotect resumed>)     = 0
[pid  1846] newfstatat(3, "",  <unfinished ...>
[pid  1845] futex(0x7fd7c358aa98, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
[pid  1846] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16555, ...}, AT_EMPTY_PATH) = 0
[pid  1846] mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd7c354b000
[pid  1846] close(3)                    = 0
[pid  1846] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) =
0x7fd7ba2f0000
[pid  1846] munmap(0x7fd7ba2f0000, 30474240) = 0
[pid  1846] munmap(0x7fd7c0000000, 36634624) = 0
[pid  1846] mprotect(0x7fd7bc000000, 135168, PROT_READ|PROT_WRITE) = 0
[pid  1846] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
[pid  1846] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
[pid  1846] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
[pid  1846] mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd7c22d0000
[pid  1846] mmap(0x7fd7c22d3000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7fd7c22d3000
[pid  1846] mmap(0x7fd7c22ea000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) =
0x7fd7c22ea000
[pid  1846] mmap(0x7fd7c22ee000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1d000) = 0x7fd7c22ee000
[pid  1846] close(3)                    = 0
[pid  1846] mprotect(0x7fd7c22ee000, 4096, PROT_READ) = 0
[pid  1846] futex(0x7fd7c358aa98, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid  1845] <... futex resumed>)        = 0
[pid  1846] <... futex resumed>)        = 1
[pid  1845] futex(0x7fd7c358aa98, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid  1846] munmap(0x7fd7c354b000, 16555 <unfinished ...>
[pid  1845] <... futex resumed>)        = 0
[pid  1846] <... munmap resumed>)       = 0
[pid  1845] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  1846] futex(0x7fd7c22ef210, FUTEX_WAKE_PRIVATE, 2147483647 <unfinished ...>
[pid  1845] <... rt_sigprocmask resumed>[], 8) = 0
[pid  1845]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETT
LS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fd7c2af0910, parent_tid=0x7fd7c2af0910, exit_signal=0,
stack=0x7fd7c22f0000, stack_size=0x7fff00, tls=0x7fd7c2af0640} <unfinished ...>
[pid  1846] <... futex resumed>)        = 0
[pid  1845] <... clone3 resumed>, 88)   = -1 ENOSYS (Function not implemented)
[pid  1846] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  1845] clone(child_stack=0x7fd7c2aefef0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLO
NE_PARENT_SETTID|CLONE_CHILD_CLEARTID <unfinished ...>
[pid  1846] <... rt_sigprocmask resumed>NULL, 8) = 0
strace: Process 1847 attached
[pid  1846] madvise(0x7fd7c2b00000, 8368128, MADV_DONTNEED <unfinished ...>
[pid  1845] <... clone resumed>, parent_tid=[1847], tls=0x7fd7c2af0640, child_tidptr=0x7fd7c2af0910) = 1847
[pid  1847] set_robust_list(0x7fd7c2af0920, 24 <unfinished ...>
[pid  1845] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1846] <... madvise resumed>)      = 0
[pid  1847] <... set_robust_list resumed>) = 0
[pid  1845] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1846] exit(0 <unfinished ...>
[pid  1845] futex(0x7fd7c3300910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1846, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  1847] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1846] <... exit resumed>)         = ?
[pid  1847] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1845] <... futex resumed>)        = 0
[pid  1846] +++ exited with 0 +++
[pid  1845] futex(0x7fd7c2af0910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1847, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  1847] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  1847] madvise(0x7fd7c22f0000, 8368128, MADV_DONTNEED) = 0
[pid  1847] exit(0)                     = ?
[pid  1847] +++ exited with 0 +++
```

```
<... futex resumed>)                    = 0
gettimeofday({tv_sec=1741239526, tv_usec=3414}, NULL) = 0
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\275\321\213\320\2
71:\n", 32Отсортированный:
) = 32
write(1, "11 36 37 64 \n", 1311 36 37 64
)         = 13
write(1, "\320\227\320\260\320\275\321\217\320\273\320\276 \320\262\321\200\320\265\320\274\320\265\320\275\320\270:
0.0"..., 513аняло времени: 0.086616 секунд
) = 51
write(1, "\320\232\320\276\320\273\320\270\321\207\320\265\321\201\321\202\320\262\320\276
\320\277\320\276\321\202\320\276\320\272\320"..., 39Количество потоков: 2
) = 39
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Наблюдения

Массив из 10 000 000 элементов

Число потоков	Время исполнения (с)	Ускорение	Эффективность
1	16.4	1	1
2	9.2	1.8	0,90
3	6.2	2.6	0,97
4	5.4	3.0	0,75
5	4.6	3.6	0,72
6	4.3	3.8	0,63

Ускорение показывает во сколько раз применение параллельного алгоритма уменьшает время решения задачи по сравнению с последовательным алгоритмом. Ускорение определяется величиной SN=T1 /TN, где T1 - время выполнения на одном потоке, TN - время выполнения на N потоках.

Эффективность - величина EN = SN/N, где SN - ускорение, N - количество используемых потоков.

Вывод

На основе тестирования программы с разным количеством потоков и объемом данных можно сделать следующие выводы:

1. Многопоточность значительно ускоряет выполнение программы при грамотном распределении нагрузки между потоками.

2. Для максимального ускорения следует выбирать оптимальное количество потоков, которое соответствует вычислительным возможностям компьютера (например, числу ядер процессора) и объему задачи.

3. Избыточное количество потоков может снижать эффективность работы из-за накладных расходов на управление потоками и синхронизацию.

Таким образом, многопоточность является эффективным инструментом для повышения

производительности, если её правильно применять в зависимости от аппаратных характеристик и сложности задачи