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-210Б-23

Студент: Воронухин Н.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.12.24

Москва, 2024

Постановка задачи

Вариант 1.

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.

Отсортировать массив целых чисел при помощи битонической сортировки.

Общий метод и алгоритм решения

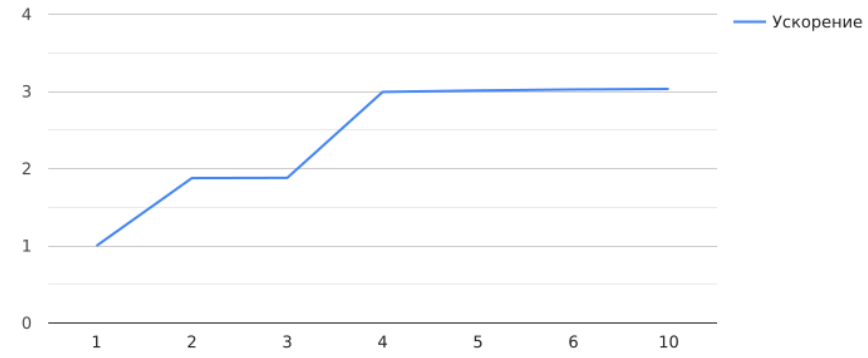
Использованные системные вызовы:

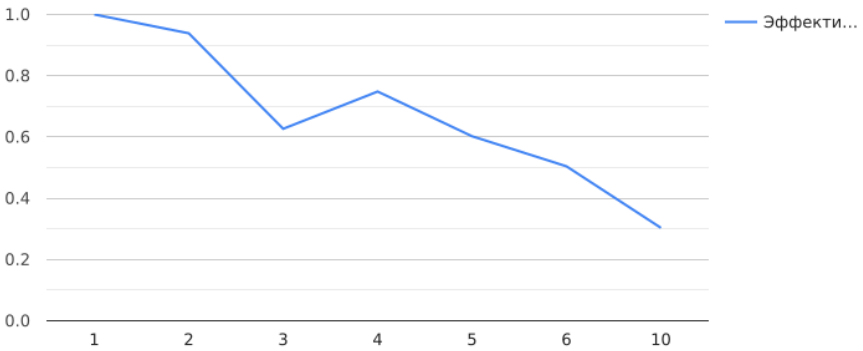
- void exit(int status); - приводит к обычному завершению программы, и величина status возвращается процессу-родителю.
- exit_group – системный вызов, завершающий все потоки процесса.
- clone – используется для создания дочернего процесса или потока в Linux
- futex – используется для создания фьютекса – легковесной версии мьютекса.
- void * mmap(void **start*, size_t *length*, int *prot* , int *flags*, int *fd*, off_t *offset*); - отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым описателем fd, в память, начиная с адреса start. Последний параметр (адрес) необязателен, и обычно бывает равен 0. Настоящее местоположение отраженных данных возвращается самой функцией mmap, и никогда не бывает равным 0.
- int munmap(void **start*, size_t *length*); - удаляет отображение для указанного адресного диапазона

Программа в аргументах командной строки принимает количество потоков, которые необходимо создать. Инициализирует большой массив для сортировки псевдослучайных чисел. Сортирует его с помощью рекурсивной битонической сортировки, на определенных уровнях которой создаются дополнительные потоки, по сути, делящие массив на несколько, сортируя соответствующие части. Размер массива ограничен числами, являющимися степенью 2, как и количество потоков.

Влияние количества потоков на скорость выполнения программы:

Число потоков	Время исполнения (мс)	Ускорение	Эффективность
1	25364	1	1
2	13511	1,877285175	0,938642588
3	13489	1,880346949	0,626782316
4	8473	2,993508793	0,748377198
5	8421	3,011993825	0,602398765
6	8384	3,02528626	0,504214377
10	8364	3,032520325	0,303252033





Код программы

Lab_2.c

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

const int ARRAY_SIZE = 1024 * 1024 * 16;

enum ErrorCodes {
    WRONG_ARGS = 1,
    WRONG_THREADS,
```

```
};

typedef enum Direction {
    INCREASING,
    DECREASING,
    ALLOC_ERR
} Direction;

typedef struct thread_data {
    int *arr;
    int start;
    int count;
    Direction dir;
    int spawn;
} thread_data;

void cmp_swap(int*, int, int, Direction);
void bitsort_merge(int*, int, int, Direction);
void* bitsort(void*);
int greatest_power_of_two_in_num(int);
void exit_err(int, char *, int);


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

    if (argc != 2) {
        char mes[] = "Invalid number of arguments.\n";
        exit_err(WRONG_ARGS, mes, sizeof(mes));
    }
    int* arr = (int*)malloc(ARRAY_SIZE * sizeof(int));
    if (arr == NULL) {
        char mes[] = "Memory allocation error";
```

```
        exit_err(ALLOC_ERR, mes, sizeof(mes));
    }
    for (int i = 0; i < ARRAY_SIZE; ++i) {
        arr[i] = rand() % 100000;
    }
    int threads_num = atoi(argv[1]);
    if (threads_num < 1) {
        char mes[] = "Invalid number of threads.\n";
        exit_err(WRONG_THREADS, mes, sizeof(mes));
    }
    int p_thr = 0;
    p_thr = greatest_power_of_two_in_num(threads_num);
    if (threads_num & (threads_num - 1) != 0) {
        threads_num = 1 << p_thr;
    }
    thread_data d = {arr, 0, ARRAY_SIZE, INCREASING, p_thr};
    bitsort(&d);
    free(arr);
}

void exit_err(int code, char* mes, int mes_size) {
    write(STDERR_FILENO, mes, mes_size);
    exit(code);
}

void* bitsort(void* args) {
    thread_data* data = (thread_data*)args;
    if (data->count > 1) {
        if (data->spawn > 0) {
            int middle = data->count / 2;
            pthread_t thread_1, thread_2;
            thread_data left = {
                data->arr,
```

```
            data->start,
            middle,
            INCREASING,
            data->spawn - 1,
        };
        if (pthread_create(&thread_1, NULL, bitsort, (void *)&left)){
            char mes[] = "Failed to create thread.\n";
            write(STDERR_FILENO, mes, sizeof(mes));
            bitsort((void*)&left);

            thread_data right = {
                data->arr,
                data->start + middle,
                middle,
                DECREASING,
                data->spawn - 1,
            };
            bitsort((void *)&right);
        }
        else {
            thread_data right = {
                data->arr,
                data->start + middle,
                middle,
                DECREASING,
                data->spawn - 1,
            };
            bitsort((void *)&right);
            pthread_join(thread_1, NULL);
        }
    }
    else {
        int middle = data->count / 2;
```

```
                thread_data left = {data->arr, data->start, middle, INCREASING, data->spawn};
                bitsort((void*)&left);
                thread_data right = {data->arr, data->start + middle, middle, DECREASING, data->spawn};
                bitsort((void*)&right);
            }
            bitsort_merge(data->arr, data->start, data->count, data->dir);
        }
    }

    void bitsort_merge(int* arr, int start, int count, Direction dir) {
        if (count > 1) {
            int middle = count / 2;
            for (int i = start; i < start + middle; ++i) {
                cmp_swap(arr, i, i + middle, dir);
            }
            bitsort_merge(arr, start, middle, dir);
            bitsort_merge(arr, start + middle, middle, dir);
        }
    }

    void cmp_swap(int *arr, int left, int right, Direction dir) {
        if ((arr[left] > arr[right] && dir == INCREASING) ||
        (arr[left] < arr[right] && dir == DECREASING)) {
            int tmp = arr[left];
            arr[left] = arr[right];
            arr[right] = tmp;
        }
    }
    int greatest_power_of_two_in_num(int n) {
        int p = 0;
        while (n > 1) {
            p += 1;
```

```
            n /= 2;
        }
        return p;
    }
```

Протокол работы программы

Strace:

```
execve("./lab_2", ["./lab_2", "4"], 0x7ffd3c6c5460 /* 64 vars */) = 0
brk(NULL)                               = 0x55726bbba000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f265a2e2000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=92170, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 92170, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f265a2cb000
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\0\20t\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1922136, ...}, 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, 1970000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f265a0ea000
mmap(0x7f265a110000, 1396736, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f265a110000
mmap(0x7f265a265000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17b000) = 0x7f265a265000
mmap(0x7f265a2b8000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1ce000) = 0x7f265a2b8000
mmap(0x7f265a2be000, 53072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f265a2be000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f265a0e7000
arch_prctl(ARCH_SET_FS, 0x7f265a0e7740) = 0
set_tid_address(0x7f265a0e7a10)         = 3054
```

```
     set_robust_list(0x7f265a0e7a20, 24)     = 0

     rseq(0x7f265a0e8060, 0x20, 0, 0x53053053) = 0

     mprotect(0x7f265a2b8000, 16384, PROT_READ) = 0

     mprotect(0x557269e89000, 4096, PROT_READ) = 0

     mprotect(0x7f265a314000, 8192, PROT_READ) = 0

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

     munmap(0x7f265a2cb000, 92170)           = 0

     getrandom("\x54\x46\xd1\x69\xdd\x5a\xf4\x87", 8, GRND_NONBLOCK) = 8

     brk(NULL)                               = 0x55726bbba000

     brk(0x55726bbdb000)                     = 0x55726bbdb000

     mmap(NULL, 67112960, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f26560e6000

     rt_sigaction(SIGRT_1, {sa_handler=0x7f265a170720, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f265a126050}, 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) =
0x7f26558e5000

     mprotect(0x7f26558e6000, 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|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f26560e5990,
parent_tid=0x7f26560e5990, exit_signal=0, stack=0x7f26558e5000, stack_size=0x7fff80,
tls=0x7f26560e56c0}strace: Process 3055 attached

      => {parent_tid=[3055]}, 88) = 3055

     [pid  3055] rseq(0x7f26560e5fe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid  3054] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid  3055] <... rseq resumed>)         = 0

     [pid  3054] <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid  3055] set_robust_list(0x7f26560e59a0, 24 <unfinished ...>

     [pid  3054] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     [pid  3055] <... set_robust_list resumed>) = 0

     [pid  3054] <... mmap resumed>)         = 0x7f26550e4000

     [pid  3055] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid  3054] mprotect(0x7f26550e5000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     [pid  3055] <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid  3054] <... mprotect resumed>)     = 0

     [pid  3055] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
```

```
     [pid  3054] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

     [pid  3055] <... mmap resumed>)         = 0x7f26548e3000

     [pid  3054] <... rt_sigprocmask resumed>[], 8) = 0

     [pid  3055] mprotect(0x7f26548e4000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     [pid  3054]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f26558e4990,
parent_tid=0x7f26558e4990, exit_signal=0, stack=0x7f26550e4000, stack_size=0x7fff80,
tls=0x7f26558e46c0} <unfinished ...>

     [pid  3055] <... mprotect resumed>)     = 0

     [pid  3054] <... clone3 resumed> => {parent_tid=[3056]}, 88) = 3056

     [pid  3055] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE,
-1, 0 <unfinished ...>

     [pid  3054] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid  3055] <... mmap resumed>)         = 0x7f264c8e3000

     [pid  3054] <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid  3055] munmap(0x7f264c8e3000, 57790464strace: Process 3056 attached

      <unfinished ...>

     [pid  3056] rseq(0x7f26558e4fe0, 0x20, 0, 0x53053053) = 0

     [pid  3056] set_robust_list(0x7f26558e49a0, 24) = 0

     [pid  3056] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

     [pid  3055] <... munmap resumed>)       = 0

     [pid  3055] munmap(0x7f2654000000, 9318400) = 0

     [pid  3055] mprotect(0x7f2650000000, 135168, PROT_READ|PROT_WRITE) = 0

     [pid  3055] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     [pid  3055]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f26550e3990,
parent_tid=0x7f26550e3990, exit_signal=0, stack=0x7f26548e3000, stack_size=0x7fff80,
tls=0x7f26550e36c0}strace: Process 3057 attached

      <unfinished ...>

     [pid  3057] rseq(0x7f26550e3fe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid  3055] <... clone3 resumed> => {parent_tid=[3057]}, 88) = 3057

     [pid  3057] <... rseq resumed>)         = 0

     [pid  3055] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid  3057] set_robust_list(0x7f26550e39a0, 24 <unfinished ...>

     [pid  3055] <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid  3057] <... set_robust_list resumed>) = 0

     [pid  3057] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
```

```
[pid  3054] futex(0x7f26558e4990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 3056, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid  3055] futex(0x7f26550e3990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 3057, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid  3056] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid  3056] madvise(0x7f26550e4000, 8368128, MADV_DONTNEED) = 0

[pid  3056] exit(0)                     = ?

[pid  3054] <... futex resumed>)        = 0

[pid  3056] +++ exited with 0 +++

[pid  3057] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid  3057] madvise(0x7f26548e3000, 8368128, MADV_DONTNEED) = 0

[pid  3057] exit(0)                     = ?

[pid  3057] +++ exited with 0 +++

[pid  3055] <... futex resumed>)        = 0

[pid  3055] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid  3055] madvise(0x7f26558e5000, 8368128, MADV_DONTNEED) = 0

[pid  3055] exit(0)                     = ?

[pid  3055] +++ exited with 0 +++

munmap(0x7f26560e6000, 67112960)        = 0

exit_group(0)                           = ?

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

Вывод

При выполнении этой работы я получил практические навыки в создании и управлении потоками посредством системных вызовов, и в организации межпоточного взаимодействия, и рассмотрел влияние количества потоков на скорость выполнения программы.