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-209БВ-24

Студент: Попов П.А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 19.12.25

Москва, 2024

Постановка задачи

Вариант 9.

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.

Вариант 9: рассчитать детерминант матрицы (используя определение детерминанта).

Общий метод и алгоритм решения

Использованные системные вызовы:

- pthread_create() – создание нового потока
- pthread_join() – ожидание завершения потока
- pthread_cancel() – отмена выполнения потока
- clock_gettime() – получение точного времени
- getpid() – получение ID процесса
- time() – получение текущего времени

Алгоритм работы программы:

1. Подготовка и настройка
    - Программа получает параметры: максимальное число потоков и размер квадратной матрицы
    - Выделяется динамическая память под матрицу заданного размера
    - Матрица заполняется псевдослучайными целыми числами в диапазоне от -8 до 7
    - Вычисляется общее число перестановок (факториал размера матрицы)
    - Определяется реальное количество рабочих потоков (минимум между ограничением и числом перестановок)
    - Работа равномерно делится между потоками по диапазонам индексов перестановок
2. Параллельные вычисления
    - Запускаются независимые потоки, каждый получает свой блок перестановок для обработки
    - Внутри каждого потока:
        - Для каждого индекса из своего диапазона генерируется соответствующая перестановка
        - Вычисляется произведение матричных элементов согласно этой перестановке
        - Определяется знак перестановки через подсчет инверсий
        - Накопительно суммируются взвешенные произведения
    - Потоки работают без взаимодействия, каждый имеет собственные данные
3. Объединение результатов

- Основной поток ожидает завершения всех рабочих потоков
- Собирает частичные суммы от каждого потока
- Суммирует все частичные результаты в итоговое значение определителя
- Синхронизация не требуется, так как потоки работают с непересекающимися данными

4. Формирование отчета
   - Вычисляется и выводится время выполнения параллельной части
   - Отображается полученное значение определителя
   - Показывается фактическое количество задействованных потоков
   - Выводится идентификатор процесса для возможного мониторинга потоков средствами ОС

Код программы

determinant_calculator.c

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int THREAD_CAP = 4;
int USE_FIXED_SEED = 0;
unsigned int CUSTOM_SEED = 0;

typedef struct {
    int worker_id;
    int dim;
    double *mat;
    long long from_idx;
    long long to_idx;
    double worker_total;
} WorkerInfo;

long long get_fact(int x) {
    long long res = 1;
    int i;
    for (i = 2; i <= x; i++) res = res * i;
    return res;
}

int get_parity(int *seq, int len) {
    int swaps = 0;
    int i, j;
    for (i = 0; i < len; i++) {
        for (j = i + 1; j < len; j++) {
```

```
            if (seq[i] > seq[j]) swaps++;
        }
    }
    return (swaps % 2) ? -1 : 1;
}

void generate_seq(long long idx, int *output, int n) {
    int *used = (int*)calloc(n, sizeof(int));
    if (!used) {
        fprintf(stderr, "Ошибка выделения памяти в generate_seq\n");
        return;
    }

    int i, j;
    for (i = 0; i < n; i++) {
        int pos = idx % (n - i);
        idx = idx / (n - i);

        int cnt = 0;
        for (j = 0; j < n; j++) {
            if (!used[j]) {
                if (cnt == pos) {
                    output[i] = j;
                    used[j] = 1;
                    break;
                }
                cnt++;
            }
        }
    }

    free(used);
}

void* compute_slice(void *ptr) {
    WorkerInfo *wi = (WorkerInfo*)ptr;
    int n = wi->dim;
    double *m = wi->mat;

    int *current_seq = (int*)malloc(n * sizeof(int));
    if (!current_seq) {
        fprintf(stderr, "Поток %d: не удалось выделить память\n", wi->worker_id);
        wi->worker_total = 0.0;
        return NULL;
    }

    double slice_sum = 0.0;
    long long k;
```

```
    for (k = wi->from_idx; k < wi->to_idx; k++) {
        generate_seq(k, current_seq, n);

        double term = 1.0;
        int row;
        for (row = 0; row < n; row++) {
            term = term * m[row * n + current_seq[row]];
        }

        slice_sum += get_parity(current_seq, n) * term;
    }

    wi->worker_total = slice_sum;
    free(current_seq);
    return NULL;
}

void print_usage(const char *prog_name) {
    printf("Использование: %s <макс_потоков> <размер_матрицы> [опции]\n", prog_name);
    printf("\nОпции:\n");
    printf("  -f [SEED]    Фиксированный seed (по умолчанию 42)\n");
    printf("  -s SEED      Конкретный seed значение\n");
    printf("  -h           Справка\n");
}

int main(int argc, char *argv[]) {
    if (argc < 3) {
        print_usage(argv[0]);
        return 1;
    }

    THREAD_CAP = atoi(argv[1]);
    int mat_size = atoi(argv[2]);

    for (int i = 3; i < argc; i++) {
        if (strcmp(argv[i], "-h") == 0) {
            print_usage(argv[0]);
            return 0;
        }
        else if (strcmp(argv[i], "-f") == 0) {
            USE_FIXED_SEED = 1;
            if (i + 1 < argc && argv[i + 1][0] != '-') {
                CUSTOM_SEED = atoi(argv[i + 1]);
                i++;
            } else {
                CUSTOM_SEED = 42;
            }
        }
        else if (strcmp(argv[i], "-s") == 0) {
```

```
            USE_FIXED_SEED = 1;
            if (i + 1 < argc) {
                CUSTOM_SEED = atoi(argv[i + 1]);
                i++;
            } else {
                fprintf(stderr, "Ошибка: опция -s требует значения seed\n");
                return 1;
            }
        }
        else {
            fprintf(stderr, "Неизвестная опция: %s\n", argv[i]);
            print_usage(argv[0]);
            return 1;
        }
    }

    if (mat_size < 1 || mat_size > 12) {
        fprintf(stderr, "Ошибка: размер матрицы должен быть от 1 до 12\n");
        return 1;
    }

    if (THREAD_CAP < 1) {
        fprintf(stderr, "Ошибка: число потоков должно быть > 0\n");
        return 1;
    }

    double *M = (double*)calloc(mat_size * mat_size, sizeof(double));
    if (M == NULL) {
        perror("calloc");
        return 1;
    }

    if (USE_FIXED_SEED) {
        srand(CUSTOM_SEED);
        printf("Используется фиксированный seed: %u\n", CUSTOM_SEED);
    } else {
        unsigned int seed = time(NULL) ^ getpid();
        srand(seed);
        printf("Используется случайный seed: %u\n", seed);
    }

    printf("\nМатрица %d×%d:\n", mat_size, mat_size);
    int i, j;
    for (i = 0; i < mat_size; i++) {
        printf("  ");
        for (j = 0; j < mat_size; j++) {
            M[i * mat_size + j] = (rand() % 16) - 8;
            printf("%4.0f ", M[i * mat_size + j]);
        }
```

```
    printf("\n");
}

long long perm_count = get_fact(mat_size);
printf("\nКоличество перестановок: %lld\n", perm_count);

int real_threads = THREAD_CAP;
if (real_threads > perm_count) real_threads = perm_count;

pthread_t *thread_list = (pthread_t*)malloc(real_threads * sizeof(pthread_t));
WorkerInfo *work_info = (WorkerInfo*)malloc(real_threads * sizeof(WorkerInfo));

if (thread_list == NULL || work_info == NULL) {
    perror("malloc");
    free(M);
    return 1;
}

long long base_load = perm_count / real_threads;
long long extra_load = perm_count % real_threads;
long long cursor = 0;

struct timespec t_start, t_end;
clock_gettime(CLOCK_MONOTONIC, &t_start);

for (i = 0; i < real_threads; i++) {
    long long load = base_load + (i < extra_load ? 1 : 0);

    work_info[i].worker_id = i;
    work_info[i].dim = mat_size;
    work_info[i].mat = M;
    work_info[i].from_idx = cursor;
    work_info[i].to_idx = cursor + load;
    work_info[i].worker_total = 0.0;

    cursor += load;

    if (pthread_create(&thread_list[i], NULL, compute_slice, &work_info[i]) != 0) {
        fprintf(stderr, "Не удалось создать поток %d: ", i);
        perror(NULL);

        for (int j = 0; j < i; j++) {
            pthread_cancel(thread_list[j]);
        }
        free(M); free(thread_list); free(work_info);
        return 1;
    }
}
```

```
    double final_det = 0.0;
    for (i = 0; i < real_threads; i++) {
        pthread_join(thread_list[i], NULL);
        final_det += work_info[i].worker_total;
    }

    clock_gettime(CLOCK_MONOTONIC, &t_end);

    double elapsed = (t_end.tv_sec - t_start.tv_sec) +
                     (t_end.tv_nsec - t_start.tv_nsec) / 1e9;

    printf("\n=== РЕЗУЛЬТАТЫ ===\n");
    printf("  Определитель:   %12.4f\n", final_det);
    printf("  Время работы:   %12.6f c\n", elapsed);
    printf("  Использовано потоков: %4d из %d\n", real_threads, THREAD_CAP);
    printf("  Производительность:   %12.0f перестановок/c\n",
           (elapsed > 0) ? perm_count / elapsed : 0);
    printf("  ID процесса:    %12d\n", getpid());

    free(M);
    free(thread_list);
    free(work_info);

    return 0;
}
```

Протокол работы программы

Тестирование:

./determinant_calculator 6 6

Используется случайный seed: 1766205017

Матрица 6×6:

-2 0 -1 -5 -6 -7

2 -7 -3 0 -4 2

1 5 6 2 7 0

7 -3 3 5 -4 -1

7 -2 -8 -1 3 -7

-7 -7 1 0 -4 3

Количество перестановок: 720

=== РЕЗУЛЬТАТЫ ===

Определитель: 153270.0000

Время работы: 0.001008 с

Использовано потоков: 6 из 6

Производительность: 714418 перестановок/с

ID процесса: 1158

$ ./determinant_calculator 6 6 -f

Используется фиксированный seed: 42

Матрица 6×6:

```
-2  -4  -7   1  -4  -2
 5   4  -1   7  -6  -1
-3   0  -8  -6  -4   0
 2   3   3  -8  -7   1
 7   3  -7   0   5  -8
-7  -5  -4  -6   4   0
```

Количество перестановок: 720

=== РЕЗУЛЬТАТЫ ===

Определитель: -51832.0000

Время работы: 0.001051 с

Использовано потоков: 6 из 6

Производительность: 684930 перестановок/с

ID процесса: 1183

$ ./determinant_calculator 6 6 -f 6

Используется фиксированный seed: 6

Матрица 6×6:

```
 5   1   0  -7   4  -3
```

-2 -5 2 -6 7 3

-3 -3 -6 0 -8 4

-1 0 7 6 4 6

5 -5 4 -4 -3 7

-1 -6 0 7 -4 -3

Количество перестановок: 720

=== РЕЗУЛЬТАТЫ ===

Определитель: 177146.0000

Время работы: 0.000979 с

Использовано потоков: 6 из 6

Производительность: 735779 перестановок/с

ID процесса: 1194

Strace:

```
execve("./determinant_calculator", ["./determinant_calculator", "4", "6", "-f"], 0x7ffcd3d459a8 /* 28 vars */) = 0
brk(NULL)                               = 0x5eb31ed78000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7eeb8341a000
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=41635, ...}) = 0
mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7eeb8340f000
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\220\243\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
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7eeb83000000
mmap(0x7eeb83028000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7eeb83028000
mmap(0x7eeb831b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000)
= 0x7eeb831b0000
mmap(0x7eeb831ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7eeb831ff000
mmap(0x7eeb83205000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7eeb83205000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7eeb8340c000
arch_prctl(ARCH_SET_FS, 0x7eeb8340c740) = 0
set_tid_address(0x7eeb8340ca10)         = 1267
set_robust_list(0x7eeb8340ca20, 24)     = 0
```

```
rseq(0x7eeb8340d060, 0x20, 0, 0x53053053) = 0
mprotect(0x7eeb831ff000, 16384, PROT_READ) = 0
mprotect(0x5eb2e15dc000, 4096, PROT_READ) = 0
mprotect(0x7eeb83452000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7eeb8340f000, 41635)           = 0
getrandom("\x66\xe4\x14\xbf\x73\xfb\x55\x08", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5eb31ed78000
brk(0x5eb31ed99000)                     = 0x5eb31ed99000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "\320\230\321\201\320\277\320\276\320\273\321\214\320\267\321\203\320\265\321\202\321\201\321\217
\321\204\320\270\320\272\321"..., 61Используется фиксированный seed: 42
) = 61
write(1, "\n\320\234\320\260\321\202\321\200\320\270\321\206\320\260 6\303\2276:\n", 22
Матрица 6×6:
) = 22
write(1, "    -2   -4   -7    1   -4   -2 "..., 33    -2   -4   -7    1   -4   -2
) = 33
write(1, "     5    4   -1    7   -6   -1 "..., 33     5    4   -1    7   -6   -1
) = 33
write(1, "    -3    0   -8   -6   -4    0 "..., 33    -3    0   -8   -6   -4    0
) = 33
write(1, "     2    3    3   -8   -7    1 "..., 33     2    3    3   -8   -7    1
) = 33
write(1, "     7    3   -7    0    5   -8 "..., 33     7    3   -7    0    5   -8
) = 33
write(1, "    -7   -5   -4   -6    4    0 "..., 33    -7   -5   -4   -6    4    0
) = 33
write(1, "\n\320\232\320\276\320\273\320\270\321\207\320\265\321\201\321\202\320\262\320\276
\320\277\320\265\321\200\320\265\321\201"..., 52
Количество перестановок: 720
) = 52
rt_sigaction(SIGRT_1, {sa_handler=0x7eeb83099530, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7eeb83045330}, 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) =
0x7eeb827ff000
mprotect(0x7eeb82800000, 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_SYSVSE
M|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7eeb82fff990,
parent_tid=0x7eeb82fff990, exit_signal=0, stack=0x7eeb827ff000, stack_size=0x7fff80, tls=0x7eeb82fff6c0} =>
{parent_tid=[1268]}, 88) = 1268
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7eeb81ffe000
mprotect(0x7eeb81fff000, 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_SYSVSE
M|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7eeb827fe990,
parent_tid=0x7eeb827fe990, exit_signal=0, stack=0x7eeb81ffe000, stack_size=0x7fff80, tls=0x7eeb827fe6c0} =>
{parent_tid=[1269]}, 88) = 1269
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7eeb817fd000
mprotect(0x7eeb817fe000, 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=0x7eeb81ffd990, parent_tid=0x7eeb81ffd990, exit_signal=0, stack=0x7eeb81ffd000, stack_size=0x7fff80, tls=0x7eeb81ffd6c0} => {parent_tid=[1270]}, 88) = 1270
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7eeb80ffc000
        mprotect(0x7eeb80ffd000, 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=0x7eeb817fc990, parent_tid=0x7eeb817fc990, exit_signal=0, stack=0x7eeb80ffc000, stack_size=0x7fff80, tls=0x7eeb817fc6c0} => {parent_tid=[1271]}, 88) = 1271
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        write(1, "\n", 1
        )                       = 1
        write(1, "=== \320\240\320\225\320\227\320\243\320\233\320\254\320\242\320\220\320\242\320\253 ===\n", 29=== РЕЗУЛЬТАТЫ ===
        ) = 29
        write(1, "  \320\236\320\277\321\200\320\265\320\264\320\265\320\273\320\270\321\202\320\265\320\273\321\214: -"..., 43  Определитель:    -51832.0000
        ) = 43
        write(1, "  \320\222\321\200\320\265\320\274\321\217 \321\200\320\260\320\261\320\276\321\202\321\213:      "..., 45  Время работы:      0.007061 с
        ) = 45
        write(1, "  \320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\320\260\320\275\320\276\320\277\320\276\321"..., 55  Использовано потоков:    4 из 4
        ) = 55
        write(1, "  \320\237\321\200\320\276\320\270\320\267\320\262\320\276\320\264\320\270\321\202\320\265\320\273\321\214\320\275\320\276"..., 83  Производительность:          101972 перестановок/с
        ) = 83
        getpid()                        = 1267
        write(1, "  ID \320\277\321\200\320\276\321\206\320\265\321\201\321\201\320\260:          "..., 39  ID процесса: 1267
        ) = 39
        exit_group(0)                   = ?
        +++ exited with 0 +++
```

Потоки	Время (с)	Ускорение	Эффективность
1	0.016974	1.00	1.000
2	0.009177	1.84	0.920
3	0.006219	2.72	0.906
4	0.006284	2.70	0.675
6	0.005054	3.35	0.558
8	0.005880	2.88	0.360

С ростом числа потоков до 3 ускорение растет до 2.72×, но эффективность падает с 1.0 до 0.906. При 4-8 потоках ускорение достигает максимума 3.35× (6 потоков), затем снижается, а

эффективность падает до 0.360. Это вызвано накладными расходами на создание потоков, конкуренцией за ресурсы CPU и ограниченным параллелизмом задачи для матрицы 8×8.

Вывод

В ходе лабораторной работы были освоены практические навыки управления потоками с использованием библиотеки pthread, а также методы анализа производительности программы при помощи strace. На примере реализации параллельного вычисления определителя матрицы подтверждена возможность сокращения времени выполнения за счёт распараллеливания, однако низкая эффективность при большом количестве потоков показала важность учёта накладных расходов на организацию многопоточности. Полученный опыт позволяет сделать вывод о целесообразности применения многопоточных вычислений для задач с достаточным объёмом независимых операций.