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: Ле Шон Лыонг

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 19.11.25

Москва, 2025

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.

Рассчитать детерминант матрицы (используя определение детерминанта)

Общий метод и алгоритм решения

Использованные системные вызовы:

• clock_gettime(CLOCK_MONOTONIC, &start) — измерение времени выполнения.

• pthread_create() — создание рабочих потоков.

• pthread_join() — ожидание завершения всех потоков.

• malloc() / free() — динамическое выделение и освобождение памяти.

Алгоритм работы программы:

- **1. Инициализация**
  - Главный поток считывает параметры запуска: максимальное количество потоков и размер матрицы.
  - Выделяется память под матрицу и заполняется псевдослучайными значениями.
  - Вычисляется количество перестановок $n!$, определяющих объём работы.
  - Фактическое число потоков выбирается как минимум между заданным пользователем и количеством доступных перестановок.
  - Диапазон перестановок равномерно распределяется между потоками.
- **2. Параллельное вычисление частичных сумм**
  - Каждому потоку передаётся свой диапазон номеров перестановок.
  - Для каждой перестановки поток:
    - преобразует её номер в саму перестановку (лексикографическая нумерация);
    - вычисляет произведение элементов матрицы по этой перестановке;
    - определяет знак перестановки;
    - прибавляет результат в свою локальную сумму.
  - Потоки работают независимо, так как каждый имеет собственное пространство вычислений.
- **3. Сбор результатов**

- Главный поток ожидает завершения всех рабочих потоков через `pthread_join()`.
- После завершения всех потоков их частичные суммы объединяются в итоговое значение детерминанта.
- Так как потоки не обращаются к общей памяти одновременно, дополнительная синхронизация не требуется.

- ### 4. Вывод результатов

  - Программа выводит:
    - вычисленный детерминант,
    - время выполнения,
    - фактическое количество использованных потоков,
    - PID процесса для анализа потоков через `strace`.

Код программы

lab2_determinant.c

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

int MAX_THREADS = 4;

typedef struct {
    int id;
    int n;
    double *matrix;
    long long start;
    long long end;
    double partial_sum;
} ThreadData;

long long factorial_ll(int n) {
    long long f = 1;
    for (int i = 2; i <= n; i++) {
        f *= i;
    }
    return f;
}

int sign_of_perm(const int *p, int n) {
    int inv = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
```

```
            if (p[i] > p[j]) {
                inv++;
            }
        }
    }
    return (inv % 2 == 0) ? 1 : -1;
}

void index_to_perm(long long idx, int *p, int n) {
    int used[20] = {0};

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

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

void *det_worker(void *arg) {
    ThreadData *data = (ThreadData *)arg;
    int n = data->n;
    double *A = data->matrix;
    int perm[20];
    double sum = 0.0;

    for (long long idx = data->start; idx < data->end; idx++) {
        index_to_perm(idx, perm, n);

        double prod = 1.0;
        for (int r = 0; r < n; r++) {
            prod *= A[r * n + perm[r]];
        }

        int s = sign_of_perm(perm, n);
        sum += s * prod;
    }

    data->partial_sum = sum;
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
```

```
        printf("Usage: %s <max_threads> <matrix_size>\n", argv[0]);
        return 1;
    }

    MAX_THREADS = atoi(argv[1]);
    int n = atoi(argv[2]);

    if (n <= 0 || n > 10) {
        printf("Matrix size n must be between 1 and 10\n");
        return 1;
    }

    if (MAX_THREADS <= 0) {
        printf("Thread count must be positive\n");
        return 1;
    }

    double *A = malloc(n * n * sizeof(double));
    if (!A) {
        perror("malloc");
        return 1;
    }

    srand(42);
    for (int i = 0; i < n * n; i++) {
        A[i] = (rand() % 11) - 5;
    }

    long long total_perms = factorial_ll(n);

    int actual_threads = MAX_THREADS;
    if (actual_threads > total_perms) {
        actual_threads = (int)total_perms;
    }

    pthread_t *threads = malloc(sizeof(pthread_t) * actual_threads);
    ThreadData *thread_data = malloc(sizeof(ThreadData) * actual_threads);
    if (!threads || !thread_data) {
        perror("malloc");
        free(A);
        return 1;
    }

    long long base = total_perms / actual_threads;
    long long rem  = total_perms % actual_threads;

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

    long long current = 0;
    for (int i = 0; i < actual_threads; i++) {
        long long chunk = base + (i < rem ? 1 : 0);

        thread_data[i].id          = i;
```

```
        thread_data[i].n           = n;
        thread_data[i].matrix      = A;
        thread_data[i].start       = current;
        thread_data[i].end         = current + chunk;
        thread_data[i].partial_sum = 0.0;

        current += chunk;

        pthread_create(&threads[i], NULL, det_worker, &thread_data[i]);
    }

    double det = 0.0;
    for (int i = 0; i < actual_threads; i++) {
        pthread_join(threads[i], NULL);
        det += thread_data[i].partial_sum;
    }

    clock_gettime(CLOCK_MONOTONIC, &end_time);

    double time_sec = (end_time.tv_sec - start_time.tv_sec) +
                      (end_time.tv_nsec - start_time.tv_nsec) / 1e9;

    printf("Matrix size: %d\n", n);
    printf("Determinant: %.4f\n", det);
    printf("Time: %lf seconds\n", time_sec);
    printf("Threads used: %d (max: %d)\n", actual_threads, MAX_THREADS);
    printf("Process PID: %d\n", getpid());

    free(A);
    free(threads);
    free(thread_data);

    return 0;
}
```

benchmark.sh

```
#!/bin/bash

echo "Performance benchmark for multithreaded determinant (definition)"
echo "================================================================"

gcc -O2 -o lab2_determinant lab2_determinant.c -lpthread -lm

N=9

echo "Matrix size: $N"
echo ""

for threads in 1 2 3 4 5 6
```

```
do
    echo "Threads: $threads"
    ./lab2_determinant $threads $N | grep -E "(Time:|Threads used:)"
    echo "----------------------------------------"
done
```

Протокол работы программы

Тестирование:

```
on@DESKTOP-F8Q4L97:/mnt/c/Users/Son/LabsOS/lab2$ ./lab2_determinant 4 9
Matrix size: 9
Determinant: -11889494.0000
Time: 0.023543 seconds
Threads used: 4 (max: 4)
Process PID: 317
```

Strace:

```
15:31:51.401012 execve("./lab2_determinant", ["./lab2_determinant", "4", "9"], 0x7fff40a5c6f0 /* 26 vars */) = 0

15:31:51.407827 brk(NULL)               = 0x5e489cf30000

15:31:51.408800 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f009a54c000

15:31:51.409933 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

15:31:51.410799 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

15:31:51.411814 fstat(3, {st_mode=S_IFREG|0644, st_size=59003, ...}) = 0

15:31:51.412842 mmap(NULL, 59003, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f009a53d000

15:31:51.413747 close(3)                = 0

15:31:51.414779 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

15:31:51.415511 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\0"..., 832) = 832

15:31:51.416474 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

15:31:51.417320 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

15:31:51.417889 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

15:31:51.418470 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f009a200000

15:31:51.419047 mmap(0x7f009a228000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f009a228000

15:31:51.419686 mmap(0x7f009a3b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f009a3b0000
```

```
     15:31:51.420369 mmap(0x7f009a3ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f009a3ff000

     15:31:51.420863 mmap(0x7f009a405000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f009a405000

     15:31:51.421349 close(3)                = 0

     15:31:51.421836 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f009a53a000

     15:31:51.422296 arch_prctl(ARCH_SET_FS, 0x7f009a53a740) = 0

     15:31:51.422775 set_tid_address(0x7f009a53aa10) = 358

     15:31:51.423189 set_robust_list(0x7f009a53aa20, 24) = 0

     15:31:51.423662 rseq(0x7f009a53b060, 0x20, 0, 0x53053053) = 0

     15:31:51.424114 mprotect(0x7f009a3ff000, 16384, PROT_READ) = 0

     15:31:51.424601 mprotect(0x5e486c574000, 4096, PROT_READ) = 0

     15:31:51.425131 mprotect(0x7f009a584000, 8192, PROT_READ) = 0

     15:31:51.425575 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

     15:31:51.426059 munmap(0x7f009a53d000, 59003) = 0

     15:31:51.426544 getrandom("\xaa\x21\x20\xa3\x15\xa8\x29\x57", 8, GRND_NONBLOCK) = 8

     15:31:51.427002 brk(NULL)               = 0x5e489cf30000

     15:31:51.427448 brk(0x5e489cf51000)     = 0x5e489cf51000

     15:31:51.427928 rt_sigaction(SIGRT_1, {sa_handler=0x7f009a299530, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f009a245330}, NULL, 8)
= 0

     15:31:51.428412 rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

     15:31:51.429310 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x7f00999ff000

     15:31:51.430008 mprotect(0x7f0099a00000, 8388608, PROT_READ|PROT_WRITE) = 0

     15:31:51.430539 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     15:31:51.431039
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=0x7f009a1ff990,
parent_tid=0x7f009a1ff990, exit_signal=0, stack=0x7f00999ff000, stack_size=0x7fff80,
tls=0x7f009a1ff6c0}strace: Process 359 attached

      => {parent_tid=[359]}, 88) = 359

     [pid   358] 15:31:51.432041 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   359] 15:31:51.432546 rseq(0x7f009a1fffe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid   358] 15:31:51.432992 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   359] 15:31:51.433205 <... rseq resumed>) = 0

     [pid   358] 15:31:51.433401 mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

     [pid   359] 15:31:51.433836 set_robust_list(0x7f009a1ff9a0, 24 <unfinished ...>
```

```
     [pid   358] 15:31:51.434228 <... mmap resumed>) = 0x7f00991fe000

     [pid   359] 15:31:51.434435 <... set_robust_list resumed>) = 0

     [pid   358] 15:31:51.434650 mprotect(0x7f00991ff000, 8388608, PROT_READ|PROT_WRITE
<unfinished ...>

     [pid   359] 15:31:51.435115 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   358] 15:31:51.435589 <... mprotect resumed>) = 0

     [pid   359] 15:31:51.435866 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   358] 15:31:51.436190 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     [pid   358] 15:31:51.436547
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=0x7f00999fe990,
parent_tid=0x7f00999fe990, exit_signal=0, stack=0x7f00991fe000, stack_size=0x7fff80,
tls=0x7f00999fe6c0} => {parent_tid=[360]}, 88) = 360

     strace: Process 360 attached

     [pid   358] 15:31:51.437452 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   360] 15:31:51.437879 rseq(0x7f00999fefe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid   358] 15:31:51.438249 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   360] 15:31:51.438388 <... rseq resumed>) = 0

     [pid   358] 15:31:51.438607 mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

     [pid   360] 15:31:51.438969 set_robust_list(0x7f00999fe9a0, 24 <unfinished ...>

     [pid   358] 15:31:51.439279 <... mmap resumed>) = 0x7f00989fd000

     [pid   360] 15:31:51.439400 <... set_robust_list resumed>) = 0

     [pid   358] 15:31:51.439583 mprotect(0x7f00989fe000, 8388608, PROT_READ|PROT_WRITE) =
0

     [pid   360] 15:31:51.439894 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   358] 15:31:51.440132 rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

     [pid   360] 15:31:51.440401 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   358] 15:31:51.440522 <... rt_sigprocmask resumed>[], 8) = 0

     [pid   358] 15:31:51.440719
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=0x7f00991fd990,
parent_tid=0x7f00991fd990, exit_signal=0, stack=0x7f00989fd000, stack_size=0x7fff80,
tls=0x7f00991fd6c0}strace: Process 361 attached

      => {parent_tid=[361]}, 88) = 361

     [pid   358] 15:31:51.441289 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   361] 15:31:51.441619 rseq(0x7f00991fdfe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid   358] 15:31:51.441855 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   361] 15:31:51.441977 <... rseq resumed>) = 0

     [pid   358] 15:31:51.442127 mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
```

```
     [pid   361] 15:31:51.442359 set_robust_list(0x7f00991fd9a0, 24 <unfinished ...>

     [pid   358] 15:31:51.442585 <... mmap resumed>) = 0x7f00981fc000

     [pid   361] 15:31:51.442704 <... set_robust_list resumed>) = 0

     [pid   358] 15:31:51.442857 mprotect(0x7f00981fd000, 8388608, PROT_READ|PROT_WRITE
<unfinished ...>

     [pid   361] 15:31:51.443083 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   358] 15:31:51.443366 <... mprotect resumed>) = 0

     [pid   361] 15:31:51.443516 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   358] 15:31:51.443648 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     [pid   358] 15:31:51.443954
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=0x7f00989fc990,
parent_tid=0x7f00989fc990, exit_signal=0, stack=0x7f00981fc000, stack_size=0x7fff80,
tls=0x7f00989fc6c0}strace: Process 362 attached

      => {parent_tid=[362]}, 88) = 362

     [pid   362] 15:31:51.444498 rseq(0x7f00989fcfe0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid   358] 15:31:51.444733 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     [pid   362] 15:31:51.445157 <... rseq resumed>) = 0

     [pid   358] 15:31:51.445337 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   362] 15:31:51.445479 set_robust_list(0x7f00989fc9a0, 24 <unfinished ...>

     [pid   358] 15:31:51.445777 futex(0x7f009a1ff990,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 359, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

     [pid   362] 15:31:51.446018 <... set_robust_list resumed>) = 0

     [pid   362] 15:31:51.446160 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

     [pid   359] 15:31:51.454903 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     [pid   359] 15:31:51.455899 madvise(0x7f00999ff000, 8368128, MADV_DONTNEED) = 0

     [pid   359] 15:31:51.456694 exit(0)     = ?

     [pid   358] 15:31:51.458020 <... futex resumed>) = 0

     [pid   359] 15:31:51.458571 +++ exited with 0 +++

     [pid   358] 15:31:51.459012 futex(0x7f00999fe990,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 360, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

     [pid   360] 15:31:51.460227 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     [pid   360] 15:31:51.461472 madvise(0x7f00991fe000, 8368128, MADV_DONTNEED) = 0

     [pid   360] 15:31:51.462237 exit(0 <unfinished ...>

     [pid   361] 15:31:51.463151 rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

     [pid   360] 15:31:51.463981 <... exit resumed>) = ?

     [pid   361] 15:31:51.464469 <... rt_sigprocmask resumed>NULL, 8) = 0

     [pid   358] 15:31:51.464880 <... futex resumed>) = 0

     [pid   360] 15:31:51.465343 +++ exited with 0 +++
```

```
[pid   358] 15:31:51.465637 futex(0x7f00991fd990,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 361, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid   361] 15:31:51.466386 madvise(0x7f00989fd000, 8368128, MADV_DONTNEED <unfinished
...>

[pid   362] 15:31:51.467017 rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

[pid   361] 15:31:51.467624 <... madvise resumed>) = 0

[pid   362] 15:31:51.467847 <... rt_sigprocmask resumed>NULL, 8) = 0

[pid   361] 15:31:51.468138 exit(0 <unfinished ...>

[pid   362] 15:31:51.468636 madvise(0x7f00981fc000, 8368128, MADV_DONTNEED <unfinished
...>

[pid   361] 15:31:51.469186 <... exit resumed>) = ?

[pid   362] 15:31:51.469464 <... madvise resumed>) = 0

[pid   358] 15:31:51.469797 <... futex resumed>) = 0

[pid   361] 15:31:51.470061 +++ exited with 0 +++

[pid   358] 15:31:51.470320 futex(0x7f00989fc990,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 362, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid   362] 15:31:51.470722 exit(0)     = ?

[pid   362] 15:31:51.471311 +++ exited with 0 +++

15:31:51.471425 <... futex resumed>)    = 0

15:31:51.471553 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

15:31:51.471922 write(1, "Matrix size: 9\n", 15) = 15

15:31:51.472610 write(1, "Determinant: -11889494.0000\n", 28) = 28

15:31:51.473246 write(1, "Time: 0.043624 seconds\n", 23) = 23

15:31:51.473608 write(1, "Threads used: 4 (max: 4)\n", 25) = 25

15:31:51.473920 getpid()                = 358

15:31:51.474212 write(1, "Process PID: 358\n", 17) = 17

15:31:51.474494 exit_group(0)           = ?

15:31:51.475083 +++ exited with 0 +++
```

Число потоков	Время выполнения (мс)	Ускорение	Эффективность
1	79,85	1	1
2	44,00	1,81	0,905
3	31,30	2,55	0,850
4	23,54	3,39	0,848
5	20,20	3,95	0,790

6	17,30	4,61	0,768

Эксперимент показал, что при увеличении числа потоков время выполнения заметно уменьшается. На 4 потоках достигается максимальное ускорение — примерно 3,4 раза по сравнению с однопоточным режимом. При дальнейшем увеличении потоков эффективность снижается из-за накладных расходов, но ускорение всё ещё растёт. Алгоритм хорошо масштабируется до количества физических ядер процессора.

Вывод

В ходе работы были освоены основные приёмы создания и управления потоками с использованием библиотеки pthread, а также методы измерения времени выполнения и анализа поведения программы через strace. На практике реализована многопоточная программа вычисления детерминанта по определению. Потоки работали корректно и независимо, что позволило обойтись без дополнительных механизмов синхронизации. Эксперимент подтвердил уменьшение времени выполнения при увеличении числа потоков.