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:

- clock_gettime(CLOCK_MONOTONIC, &t_start) — измерение времени выполнения
- pthread_create() — создание рабочих потоков
- pthread_join() — ожидание завершения потоков
- malloc() / free() — динамическое выделение и освобождение памяти

Алгоритм:

1. Подготовка и настройка

- Программа получает два параметра: ограничение по потокам 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>

int THREAD_CAP = 4;

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 free[15] = {0};
    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 (!free[j]) {
                if (cnt == pos) {
                    output[i] = j;
                    free[j] = 1;
                    break;
                }
                cnt++;
            }
        }
    }
}

void* compute_slice(void *ptr) {
    WorkerInfo *wi = (WorkerInfo*)ptr;
    int n = wi->dim;
    double *m = wi->mat;
    int current_seq[15];
    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;
    return NULL;
}
```

```
int main(int argc, char *argv[]) {
    if (argc < 3) {
        fprintf(stderr, "Формат: %s <макс_потоков> <размер_матрицы>\n", argv[0]);
        return 1;
    }

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

    if (mat_size < 1 || mat_size > 10) {
        fprintf(stderr, "Ошибка: размер матрицы от 1 до 10\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) {
        fprintf(stderr, "Не удалось выделить память\n");
        return 1;
    }

    unsigned int seed = time(NULL) + getpid();
    srand(seed);

    printf("Случайная матрица %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) {
```

```
    fprintf(stderr, "Ошибка выделения памяти под потоки\n");
    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\n", i);
        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("\nРезультат вычисления:\n");
printf("  Определитель:   %12.4f\n", final_det);
printf("  Время работы:   %12.6f с\n", elapsed);
printf("  Использовано потоков: %4d из %d\n", real_threads, THREAD_CAP);
printf("  ID процесса:    %12d\n", getpid());

free(M);
```

```
    free(thread_list);
    free(work_info);

    return 0;
}
```

test_performance.sh

```
#!/bin/bash

echo "Тестирование параллельного определителя"
echo "========================================"

gcc -O2 -pthread -o determinant_calculator determinant_calculator.c -lm

SIZE=7

echo "Матрица: ${SIZE}×${SIZE}"
echo ""

echo "Потоки  | Время (с)    | Ускорение | Эффективность"
echo "--------|--------------|-----------|--------------"

BASE_TIME=0
for t in 1 2 3 4 6 8
do
    RESULT=$(./determinant_calculator $t $SIZE 2>/dev/null)

    TIME_STR=$(echo "$RESULT" | grep "Время работы:")
    TIME=$(echo "$TIME_STR" | awk '{print $3}')

    if [ -z "$TIME" ]; then
        TIME_STR=$(echo "$RESULT" | grep "Time:")
        TIME=$(echo "$TIME_STR" | awk '{print $2}')
    fi

    if [ -z "$TIME" ]; then
        echo "Ошибка: не удалось извлечь время для $t потоков"
        echo "Вывод программы был:"
        echo "$RESULT"
        continue
    fi

    TIME=${TIME/,/.}

    if [ $t -eq 1 ]; then
        BASE_TIME=$TIME
        SPEEDUP=1.00
        EFF=1.000
```

```
        printf "   %2d   |  %-10s  |  %.2fx    |  %.3f\n" $t $TIME $SPEEDUP $EFF
    else
        SPEEDUP=$(awk -v base="$BASE_TIME" -v time="$TIME" 'BEGIN {printf "%.2f",
base/time}')
        EFF=$(awk -v speedup="$SPEEDUP" -v t="$t" 'BEGIN {printf "%.3f", speedup/t}')
        printf "   %2d   |  %-10s  |  %.2fx    |  %.3f\n" $t $TIME $SPEEDUP $EFF
    fi
done
```

Протокол работы программы

Тестирование:

$ gcc -O2 -pthread -o determinant_calculator determinant_calculator.c -lm

$ ./determinant_calculator 4 4

Случайная матрица 4×4:

5 -6 6 1

6 -6 -4 -4

-7 6 -1 1

6 -1 6 -3

Количество перестановок: 24

Результат вычисления:

Определитель: 680.0000

Время работы: 0.000659 с

Использовано потоков: 4 из 4

ID процесса: 1324

Strace:

```
$ strace -f ./determinant_calculator 4 5 2>&1
execve("./determinant_calculator", ["./determinant_calculator", "4", "5"],
0x7ffe3caf55a8 /* 27 vars */) = 0
brk(NULL)                               = 0x61e13f331000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x71e14520b000
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) = 0x71e145200000
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\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) = 0x71e144e00000
     mmap(0x71e144e28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x71e144e28000
     mmap(0x71e144fb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x71e144fb0000
     mmap(0x71e144fff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x71e144fff000
     mmap(0x71e145005000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x71e145005000
     close(3)                                = 0
     mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x71e1451fd000
     arch_prctl(ARCH_SET_FS, 0x71e1451fd740) = 0
     set_tid_address(0x71e1451fda10)         = 1364
     set_robust_list(0x71e1451fda20, 24)     = 0
     rseq(0x71e1451fe060, 0x20, 0, 0x53053053) = 0
     mprotect(0x71e144fff000, 16384, PROT_READ) = 0
     mprotect(0x61e1085e6000, 4096, PROT_READ) = 0
     mprotect(0x71e145243000, 8192, PROT_READ) = 0
     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
     munmap(0x71e145200000, 41635)           = 0
     getrandom("\x95\xf3\xe1\x22\x3b\xe3\xff\x47", 8, GRND_NONBLOCK) = 8
     brk(NULL)                               = 0x61e13f331000
     brk(0x61e13f352000)                     = 0x61e13f352000
     getpid()                                = 1364
     fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
     write(1, "\320\241\320\273\321\203\321\207\320\260\320\271\320\275\320\260\321\217
\320\274\320\260\321\202\321\200\320\270\321\206\320"..., 40Случайная матрица 5×5:
     ) = 40
     write(1, "     5   -3    2   -6    6 \n", 28     5   -3    2   -6    6
     ) = 28
     write(1, "    -3   -2   -5    4    6 \n", 28    -3   -2   -5    4    6
     ) = 28
     write(1, "    -1   -7   -4   -3    0 \n", 28    -1   -7   -4   -3    0
     ) = 28
     write(1, "    -1    3   -1    6   -7 \n", 28    -1    3   -1    6   -7
     ) = 28
     write(1, "    -7   -7    6    0    5 \n", 28    -7   -7    6    0    5
     ) = 28
     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: 120
     ) = 52
     rt_sigaction(SIGRT_1, {sa_handler=0x71e144e99530, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x71e144e45330}, 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) =
0x71e1445ff000
```

```
mprotect(0x71e144600000, 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|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x71e144dff990,
parent_tid=0x71e144dff990, exit_signal=0, stack=0x71e1445ff000, stack_size=0x7fff80,
tls=0x71e144dff6c0}strace: Process 1365 attached
 => {parent_tid=[1365]}, 88) = 1365
[pid  1365] rseq(0x71e144dfffe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  1364] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1365] <... rseq resumed>)         = 0
[pid  1364] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1365] set_robust_list(0x71e144dff9a0, 24 <unfinished ...>
[pid  1364] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid  1365] <... set_robust_list resumed>) = 0
[pid  1364] <... mmap resumed>)         = 0x71e143dfe000
[pid  1365] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1364] mprotect(0x71e143dff000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  1365] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364] <... mprotect resumed>)     = 0
[pid  1365] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  1364] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  1365] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364] <... rt_sigprocmask resumed>[], 8) = 0
[pid  1365] madvise(0x71e1445ff000, 8368128, MADV_DONTNEED <unfinished ...>
[pid  1364]
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=0x71e1445fe990,
parent_tid=0x71e1445fe990, exit_signal=0, stack=0x71e143dfe000, stack_size=0x7fff80,
tls=0x71e1445fe6c0} <unfinished ...>
[pid  1365] <... madvise resumed>)      = 0
[pid  1365] exit(0)                     = ?
strace: Process 1366 attached
[pid  1364] <... clone3 resumed> => {parent_tid=[1366]}, 88) = 1366
[pid  1366] rseq(0x71e1445fefe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  1365] +++ exited with 0 +++
[pid  1364] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1366] <... rseq resumed>)         = 0
[pid  1364] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1366] set_robust_list(0x71e1445fe9a0, 24 <unfinished ...>
[pid  1364] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid  1366] <... set_robust_list resumed>) = 0
[pid  1364] <... mmap resumed>)         = 0x71e1435fd000
[pid  1366] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1364] mprotect(0x71e1435fe000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  1366] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364] <... mprotect resumed>)     = 0
[pid  1366] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  1364] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  1366] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364] <... rt_sigprocmask resumed>[], 8) = 0
[pid  1364]
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=0x71e143dfd990,
parent_tid=0x71e143dfd990, exit_signal=0, stack=0x71e1435fd000, stack_size=0x7fff80,
tls=0x71e143dfd6c0} <unfinished ...>
```

```
[pid  1366] madvise(0x71e143dfe000, 8368128, MADV_DONTNEED) = 0
strace: Process 1367 attached
[pid  1366] exit(0 <unfinished ...>
[pid  1364] <... clone3 resumed> => {parent_tid=[1367]}, 88) = 1367
[pid  1367] rseq(0x71e143dfdfe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  1366] <... exit resumed>)         = ?
[pid  1364] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1367] <... rseq resumed>)         = 0
[pid  1364] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1366] +++ exited with 0 +++
[pid  1364] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid  1367] set_robust_list(0x71e143dfd9a0, 24 <unfinished ...>
[pid  1364] <... mmap resumed>)         = 0x71e142dfc000
[pid  1367] <... set_robust_list resumed>) = 0
[pid  1364] mprotect(0x71e142dfd000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  1367] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1364] <... mprotect resumed>)     = 0
[pid  1367] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  1367] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  1364] <... rt_sigprocmask resumed>[], 8) = 0
[pid  1367] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1364]
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=0x71e1435fc990,
parent_tid=0x71e1435fc990, exit_signal=0, stack=0x71e142dfc000, stack_size=0x7fff80,
tls=0x71e1435fc6c0} <unfinished ...>
[pid  1367] madvise(0x71e1435fd000, 8368128, MADV_DONTNEED) = 0
[pid  1367] exit(0strace: Process 1368 attached
)                       = ?
[pid  1364] <... clone3 resumed> => {parent_tid=[1368]}, 88) = 1368
[pid  1368] rseq(0x71e1435fcfe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  1364] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  1367] +++ exited with 0 +++
[pid  1368] <... rseq resumed>)         = 0
[pid  1364] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  1368] set_robust_list(0x71e1435fc9a0, 24 <unfinished ...>
[pid  1364] futex(0x71e1435fc990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1368, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  1368] <... set_robust_list resumed>) = 0
[pid  1368] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid  1368] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  1368] madvise(0x71e142dfc000, 8368128, MADV_DONTNEED) = 0
[pid  1368] exit(0)                     = ?
[pid  1364] <... futex resumed>)        = 0
[pid  1368] +++ exited with 0 +++
write(1, "\n", 1
)                       = 1
write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202
\320\262\321\213\321\207\320\270\321\201\320\273\320"..., 41Результат вычисления:
) = 41
```

```
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) = 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) = 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) = 55
getpid()                                = 1364
write(1, "  ID \320\277\321\200\320\276\321\206\320\265\321\201\321\201\320\260: "..., 39) = 39
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
Определитель:     -22949.0000
Время работы: 0.033597 с
Использовано потоков:     4 из 4
ID процесса:              1364
```

```
$ ./test_performance.sh
Тестирование параллельного определителя
========================================
Матрица: 7×7

Потоки  | Время (с)    | Ускорение | Эффективность
--------|--------------|-----------|--------------
    1   |  0.000977    |  1.00x    |  1.000
    2   |  0.000826    |  1.18x    |  0.590
    3   |  0.000583    |  1.68x    |  0.560
    4   |  0.000703    |  1.39x    |  0.347
    6   |  0.000758    |  1.29x    |  0.215
    8   |  0.000971    |  1.01x    |  0.126
```

С ростом числа потоков до 3 наблюдается увеличение ускорения (до 1.68x), однако эффективность резко падает (с 1.0 до 0.56), что свидетельствует о значительных накладных расходах на создание и синхронизацию потоков. При дальнейшем увеличении числа потоков (4–8) ускорение снижается, а эффективность продолжает падать до 0.126, что объясняется ограниченным параллелизмом задачи (малый объем вычислений для матрицы 7×7) и доминированием издержек многопоточности.

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.