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-213Б-23

Студент: Гига М.Я.

Преподаватель: Бахарев В.Д

Оценка: _____

Дата: 03.11.24

Москва, 2024

Постановка задачи

Вариант 9.

Составить программу на языке C (C++), обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной систем (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы. Программа должна рассчитать детерминант матрицы (используя определение детерминанта).

Общий метод и алгоритм решения

Использованные системные функции:

- int write(int fd, void* buf, size_t count); – записывает count байт из buf в fd.
- int pthread_create(pthread_t *restrict thread, const pthread_attr_t *restrict attr, void *(*start_routine)(void *), void *restrict arg); – создает новый поток выполняющий функцию start_routine с аргументом restrict arg, и записывает Id созданного потока в thread. Вызов возвращает 0, если успешен иначе код ошибки.
- int pthread_join(pthread_t thread, void **retval); – ожидает завершения потока с id thread, и если retval не равен NULL, записывает результат работы функции потока в retval. Вызов возвращает 0, если успешен иначе код ошибки.
- pid_t fork(void); – создает дочерний процесс (используется как основа для pthread_create)

Программа main считывает количество потоков thread_number из аргументов командной строки и вальсирует значение. Далее создается и инициализируются переменная mat типа vector<vector<double>>. Данная переменная а также количество потоков передуются в функцию determinant. Эта функция проверяет является ли матрица квадратной и вызывает функцию det, передавая ей матрицу и количество потоков.

Функция det принимает матрицу mat и количество доступных потоков rem. Если матрица состоит из одного элемента, этот элемент возвращается. Создаются векторы dets для хранения детерминантов матриц меньшего размера и args для передачи аргументов в функции. В каждый элемент args записывается ссылка на матрицу, индекс обрабатываемого столба, и указатель на результат. Далее если rem < 2, то для каждого столбца вызывается функция calc с параметром из args. В противном в многопоточной части функции создается вектор th для хранения id потоков. И вычисляется возможное количество потоков которые можно выделить для вычисления детерминантов каждой из матриц меньшего размера. Проходя по всем столбцам матрицы если в th нет свободных ячеек, то ждем завершения одного из них с помощью pthread_join. В свободную ячейку записывается id потока созданного через pthread_create и вызывающего calc. После прохождения всех столбцов функция ждет завершения всех потоков. На этом завершается многопоточная часть функции. Функция возвращает сумму полученных определителей, умноженных на значение первой строки соответствующего столбца, и на $(-1)^i$, где i – номер столбца.

Функция calc принимает указатель на структуру Arg, содержащую матрицу mat, номер столбца i, указатель на результат *res и количество потоков cnt. Она создает из Arg.mat новую

матрицу mat исключением первой строки и столбца Arg.i. Далее она записывает в Arg.res результат вызова det(mat, Arg.cnt).

Код программы

main.cpp

```
#include <pthread.h>
#include <unistd.h>
#include <vector>
#include <string>

using std::string;
using std::vector;
// variant 9 Рассчитать детерминант матрицы (используя определение
детерминанта)

inline size_t min(size_t a, size_t b) {
    return a < b ? a : b;
}

inline ssize_t print(int fd, string s) {
    return write(fd, s.c_str(), s.size());
}

typedef struct Args {
    vector<vector<double>> *mat;
    double *det;
    size_t i;
    size_t cnt;
} args_t;

double det(vector<vector<double>> &mat, size_t off);

void *calc(void *ptr) {
    args_t *args = (args_t *)ptr;
    size_t ex = args->i;
    size_t s = args->mat->size();
    vector<vector<double>> mat(s - 1);
    for (size_t i = 0; i < s - 1; i++) {
        mat[i].resize(s - 1);
        for (size_t j = 0; j < ex; j++)
            mat[i][j] = (*(args->mat))[i + 1][j];
        for (size_t j = ex; j < mat.size(); j++)
            mat[i][j] = (*(args->mat))[i + 1][j + 1];
    }
    *(args->det) = det(mat, 0);
    return NULL;
}

double det(vector<vector<double>> &mat, size_t rem) {
    if (mat.size() == 1)
        return mat[0][0];

    vector<double> dets(mat.size());
    vector<args_t *> args(mat.size());
    for (size_t i = 0; i < mat.size(); i++) {
        args[i] = (args_t *)malloc(sizeof(args_t));
        if (args[i] == NULL) {
            print(STDOUT_FILENO, "ERROR: buy more ram\n");
            exit(-1);
        }
```

```
        args[i]->mat = &mat;
        args[i]->det = &dets[i];
        args[i]->i = i;
        args[i]->cnt = 0;
    }

    if (rem > 1) {
        vector<pthread_t> th(min(rem, mat.size()));
        rem -= th.size();
        size_t first = rem % (th.size() - 1);
        size_t rest = rem / (th.size() - 1);
        if (rest == 1) {
            first = rem;
            rest = 0;
        }
        size_t j = 0;
        for (size_t i = 0; i < mat.size(); i++) {
            if (mat[0][i] == 0)
                continue;
            args[i]->cnt = (j % th.size()) == 0 ? first : rest;
            if (j >= th.size() && pthread_join(th[j % th.size()], NULL)) {
                print(STDOUT_FILENO, "ERROR: failed to join thread\n");
                exit(-1);
            }
            if (pthread_create(&th[j % th.size()], NULL, calc, args[i])) {
                print(STDOUT_FILENO, "ERROR: failed to create thread\n");
                exit(-1);
            }
            j++;
        }
        for (size_t i = 0; i < min(th.size(), j); i++) {
            if (pthread_join(th[i % th.size()], NULL)) {
                print(STDOUT_FILENO, "ERROR: failed to join thread\n");
                exit(-1);
            }
        }
    } else {
        for (size_t i = 0; i < mat.size(); i++)
            if (mat[0][i] != 0)
                calc(args[i]);
    }

    double res = 0;
    for (size_t i = 0; i < dets.size(); i++) {
        free(args[i]);
        double mul = i % 2 == 0 ? 1 : -1;
        res += mul * mat[0][i] * dets[i];
    }
    return res;
}

double determinant(vector<vector<double>> &mat, size_t thread_cnt) {
    for (size_t i = 0; i < mat.size(); i++) {
        if (mat[i].size() != mat.size()) {
            print(STDERR_FILENO, "ERROR: matrix must be square\n");
            return 0;
        }
    }
    return det(mat, thread_cnt);
}

string to_string(long n, size_t d = 1) {
    string s;
    bool neg = n < 0;
```

```
    if (neg)
        n *= -1;

    while (n > 0) {
        s.push_back('0' + (n % 10));
        n /= 10;
    }

    while (s.size() < d)
        s.push_back('0');

    if (neg)
        s.push_back('-');

    for (size_t i = 0; i < s.size() / 2; i++) {
        char tmp = s[i];
        s[i] = s[s.size() - i - 1];
        s[s.size() - i - 1] = tmp;
    }
    return s;
}

string to_string(double n, size_t d = 6) {
    string s;
    long n_i = n;
    s += to_string(n_i);
    if (d > 0) {
        s.push_back('.');
        n = abs(n);
        n -= n_i;
        n *= pow(10, d);
        s += to_string((long)n, d);
    }
    return s;
}

int main(int argc, const char *argw[]) {
    if (argc < 2) {
        print(STDERR_FILENO, "ERROR: number of threads not provided\n");
        return -1;
    }
    char *end;
    size_t thread_number = strtoul(argw[1], &end, 10);
    if (*end != '\0') {
        string message = "ERROR: could not parse \"";
        message += argw[1];
        message += "\" as number\n";
        print(STDERR_FILENO, message);
        return -1;
    }
    if (thread_number == 0) {
        print(STDERR_FILENO, "number of threads must be > 0\n");
        return -1;
    }

    {
        vector<vector<double>> mat = {
            {0, 1, 2, 4, 5, 6},     {2, 3, 1, 0, -1, -2},
            {12, 321, 1, 2, 41, 1}, {12, 31, 1, 2, 41, 1},
            {-12, 32, 1, 2, 0, -1}, {1, 0, 0, 0, 1, 2}};
        double det = determinant(mat, thread_number);
        string res = "|mat| = " + to_string(det) + "\n";
        print(STDOUT_FILENO, res);
    }
```

```
    {
        vector<vector<double>> mat(10, vector<double>(10, 1));
        double det = determinant(mat, thread_number);
        string res = "|mat| = " + to_string(det) + "\n";
        print(STDOUT_FILENO, res);
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

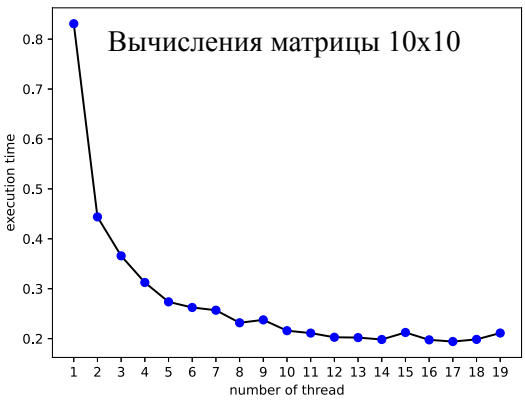
```
$ ./main.out
ERROR: number of threads not provided

$ ./main.out 0
number of threads must be > 0

$ time ./main.out 1
|mat| = 901320.000000
|mat| = 0.000000
./main.out 1  0.59s user 0.01s system 99% cpu 0.598 total

$ time ./main.out 10
|mat| = 901320.000000
|mat| = 0.000000
./main.out 10  1.40s user 0.01s system 691% cpu 0.204 total
```

Анализ зависимости времени выполнения программы от количества потоков



На двух графиках видно, что время выполнения программы значительно уменьшается пока количество потоков не достигнуло размера матрицы n. После этого изменения во времени выполнения незначительны и могут считаться погрешностью измерений. Незначительное изменение времени связано с двумя причинами. Первая причина заключается в том, что после достижения n, для каждой из матриц размера n-1 выделяется собственный поток. Вычисление определителя матрицы n-2 происходит в n-1 раз быстрее в сравнении с матрицей n-1. Для

значительного изменения скорости кол-во процессов должно быть увеличено в n-1 раз. Вторая причина – это системное ограничение на количество параллельно выполняемых потоков.

Strace:

```
$ sudo dtruss -f ./main.out 10
        PID/THRD  SYSCALL(args)                    = return
|mat| = 901320.000000
|mat| = 0.000000
 1514/0x3751:  fork()            = 0 0
 1514/0x3751:  munmap(0x100534000, 0x84000)              = 0 0
 1514/0x3751:  munmap(0x1005B8000, 0x8000)               = 0 0
 1514/0x3751:  munmap(0x1005C0000, 0x4000)               = 0 0
 1514/0x3751:  munmap(0x1005C4000, 0x4000)               = 0 0
 1514/0x3751:  munmap(0x1005C8000, 0x48000)              = 0 0
 1514/0x3751:  munmap(0x100610000, 0x4C000)              = 0 0
 1514/0x3751:  crossarch_trap(0x0, 0x0, 0x0)             = -1 Err#45
 1514/0x3751:  open(".\0", 0x100000, 0x0)                = 3 0
 1514/0x3751:  fcntl(0x3, 0x32, 0x16FB37098)             = 0 0
 1514/0x3751:  close(0x3)                = 0 0
 1514/0x3751:  fsgetpath(0x16FB370A8, 0x400, 0x16FB37088)                = 49 0
 1514/0x3751:  fsgetpath(0x16FB370B8, 0x400, 0x16FB37098)                = 14 0
 1514/0x3751:  csrctl(0x0, 0x16FB374BC, 0x4)             = -1 Err#1
 1514/0x3751:  __mac_syscall(0x1845F7D62, 0x2, 0x16FB37400)              = 0 0
 1514/0x3751:  csrctl(0x0, 0x16FB374AC, 0x4)             = -1 Err#1
 1514/0x3751:  __mac_syscall(0x1845F4B95, 0x5A, 0x16FB37440)             = 0 0
 1514/0x3751:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16FB369A8, 0x16FB369A0,
0x1845F6888, 0xD)          = 0 0
 1514/0x3751:  sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16FB36A58,
0x16FB36A50, 0x0, 0x0)               = 0 0
 1514/0x3751:  open("/\0", 0x20100000, 0x0)              = 3 0
 1514/0x3751:  openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0)
= 4 0
 1514/0x3751:  dup(0x4, 0x0, 0x0)                = 5 0
 1514/0x3751:  fstatat64(0x4, 0x16FB36531, 0x16FB364A0)          = 0 0
 1514/0x3751:  openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0)
= 6 0
 1514/0x3751:  fcntl(0x6, 0x32, 0x16FB36530)             = 0 0
 1514/0x3751:  dup(0x6, 0x0, 0x0)                = 7 0
 1514/0x3751:  dup(0x5, 0x0, 0x0)                = 8 0
 1514/0x3751:  close(0x3)                = 0 0
 1514/0x3751:  close(0x5)                = 0 0
 1514/0x3751:  close(0x4)                = 0 0
 1514/0x3751:  close(0x6)                = 0 0
 1514/0x3751:  __mac_syscall(0x1845F7D62, 0x2, 0x16FB36F20)              = 0 0
 1514/0x3751:  shared_region_check_np(0x16FB36B40, 0x0, 0x0)             = 0 0
 1514/0x3751:  fsgetpath(0x16FB370C0, 0x400, 0x16FB36FE8)                = 82 0
 1514/0x3751:  fcntl(0x8, 0x32, 0x16FB370C0)             = 0 0
 1514/0x3751:  close(0x8)                = 0 0
 1514/0x3751:  close(0x7)                = 0 0
 1514/0x3751:  getfsstat64(0x0, 0x0, 0x2)                = 10 0
 1514/0x3751:  getfsstat64(0x1002C4050, 0x54B0, 0x2)             = 10 0
 1514/0x3751:  getattrlist("/\0", 0x16FB37000, 0x16FB36F70)              = 0 0
 1514/0x3751:  stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/
dyld/dyld_shared_cache_arm64e\0", 0x16FB37360, 0x0)           = 0 0
dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return):
invalid address (0x0) in action #12 at DIF offset 12
 1514/0x3751:  stat64("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/main.out\0",
0x16FB36810, 0x0)            = 0 0
 1514/0x3751:  open("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/main.out\0", 0x0,
0x0)             = 3 0
 1514/0x3751:  mmap(0x0, 0x9528, 0x1, 0x40002, 0x3, 0x0)                 =
0x1002C4000 0
```

```
 1514/0x3751:  fcntl(0x3, 0x32, 0x16FB36928)            = 0 0
 1514/0x3751:  close(0x3)               = 0 0
 1514/0x3751:  munmap(0x1002C4000, 0x9528)              = 0 0
 1514/0x3751:  stat64("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/main.out\0",
0x16FB36D80, 0x0)           = 0 0
 1514/0x3751:  stat64("/usr/lib/libc++.1.dylib\0", 0x16FB35CD0, 0x0)
= -1 Err#2
 1514/0x3751:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libc+
+.1.dylib\0", 0x16FB35C80, 0x0)                 = -1 Err#2
 1514/0x3751:  stat64("/usr/lib/libc++.1.dylib\0", 0x16FB35CD0, 0x0)
= -1 Err#2
(Еще 4 страницы вызовов stat64)
 1514/0x3751:  open("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/main.out\0", 0x0,
0x0)            = 3 0
 1514/0x3751:  __mac_syscall(0x1845F7D62, 0x2, 0x16FB341A0)             = 0 0
 1514/0x3751:  map_with_linking_np(0x16FB33FC0, 0x1, 0x16FB33FF0)
= 0 0
 1514/0x3751:  close(0x3)               = 0 0
 1514/0x3751:  mprotect(0x1002BC000, 0x4000, 0x1)               = 0 0
 1514/0x3751:  open("/dev/dtracehelper\0", 0x2, 0x0)            = 3 0
 1514/0x3751:  ioctl(0x3, 0x80086804, 0x16FB33528)              = 0 0
 1514/0x3751:  close(0x3)               = 0 0
 1514/0x3751:  shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)
= 0 0
 1514/0x3751:  access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0)
= -1 Err#2
 1514/0x3751:  bsdthread_register(0x1848FA0F4, 0x1848FA0E8, 0x4000)
= 1073746399 0
 1514/0x3751:  getpid(0x0, 0x0, 0x0)            = 1514 0
 1514/0x3751:  shm_open(0x184791F41, 0x0, 0x5F74C000)           = 3 0
 1514/0x3751:  fstat64(0x3, 0x16FB33BA0, 0x0)           = 0 0
 1514/0x3751:  mmap(0x0, 0x8000, 0x1, 0x40001, 0x3, 0x0)                =
0x1002CC000 0
 1514/0x3751:  close(0x3)               = 0 0
 1514/0x3751:  csops(0x5EA, 0x0, 0x16FB33CDC)           = 0 0
 1514/0x3751:  ioctl(0x2, 0x4004667A, 0x16FB33C4C)              = 0 0
 1514/0x3751:  mprotect(0x1002DC000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x1002E8000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x1002EC000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x1002F8000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x1002FC000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x100308000, 0x4000, 0x0)               = 0 0
 1514/0x3751:  mprotect(0x1002D4000, 0xC8, 0x1)         = 0 0
 1514/0x3751:  mprotect(0x1002D4000, 0xC8, 0x3)         = 0 0
 1514/0x3751:  mprotect(0x1002D4000, 0xC8, 0x1)         = 0 0
 1514/0x3751:  mprotect(0x10030C000, 0x4000, 0x1)               = 0 0
 1514/0x3751:  mprotect(0x100310000, 0xC8, 0x1)         = 0 0
 1514/0x3751:  mprotect(0x100310000, 0xC8, 0x3)         = 0 0
 1514/0x3751:  mprotect(0x100310000, 0xC8, 0x1)         = 0 0
 1514/0x3751:  mprotect(0x1002D4000, 0xC8, 0x3)         = 0 0
 1514/0x3751:  mprotect(0x1002D4000, 0xC8, 0x1)         = 0 0
 1514/0x3751:  mprotect(0x10030C000, 0x4000, 0x3)               = 0 0
 1514/0x3751:  mprotect(0x10030C000, 0x4000, 0x1)               = 0 0
 1514/0x3751:  issetugid(0x0, 0x0, 0x0)         = 0 0
 1514/0x3751:  getentropy(0x16FB332B8, 0x20, 0x0)               = 0 0
 1514/0x3751:  getattrlist("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/
main.out\0", 0x16FB33B40, 0x16FB33B5C)              = 0 0
 1514/0x3751:  access("/Users/maxgiga/dev/mai/MAI_OS/lab02/src\0", 0x4, 0x0)
= 0 0
 1514/0x3751:  open("/Users/maxgiga/dev/mai/MAI_OS/lab02/src\0", 0x0, 0x0)
= 3 0
 1514/0x3751:  fstat64(0x3, 0x12FE044E0, 0x0)           = 0 0
 1514/0x3751:  csrctl(0x0, 0x16FB33D2C, 0x4)            = -1 Err#1
 1514/0x3751:  fgetattrlist(0x3, 0x16FB33DD0, 0x16FB33D50)              = 0 0
```

```
 1514/0x3751:  __mac_syscall(0x1908A2505, 0x2, 0x16FB33D50)              = 0 0
 1514/0x3751:  fcntl(0x3, 0x32, 0x16FB33A28)             = 0 0
 1514/0x3751:  close(0x3)                = 0 0
 1514/0x3751:  open("/Users/maxgiga/dev/mai/MAI_OS/lab02/src/Info.plist\0",
0x0, 0x0)            = -1 Err#2
 1514/0x3751:  proc_info(0x2, 0x5EA, 0xD)                = 64 0
 1514/0x3751:  csops_audittoken(0x5EA, 0x10, 0x16FB33DB0)                = 0 0
 1514/0x3751:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16FB34108, 0x16FB34100,
0x18800BD3A, 0x15)                = 0 0
 1514/0x3751:  sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16FB34198,
0x16FB34190, 0x0, 0x0)               = 0 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x60000203D220, 0x16FBCF000)
= 1874653184 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x60000203D240, 0x16FC5B000)
= 1875226624 0
 1514/0x375f:  fork()            = 0 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x60000203D260, 0x16FCE7000)
= 1875800064 0
 1514/0x375f:  thread_selfid(0x0, 0x0, 0x0)              = 14175 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x60000203D280, 0x16FD73000)
= 1876373504 0
 1514/0x3760:  fork()            = 0 0
 1514/0x3761:  fork()            = 0 0
 1514/0x3760:  thread_selfid(0x0, 0x0, 0x0)              = 14176 0
 1514/0x3761:  thread_selfid(0x0, 0x0, 0x0)              = 14177 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x60000203D2A0, 0x16FDFF000)
= 1876946944 0
 1514/0x3762:  fork()            = 0 0
 1514/0x3763:  fork()            = 0 0
 1514/0x3762:  thread_selfid(0x0, 0x0, 0x0)              = 14178 0
 1514/0x3763:  thread_selfid(0x0, 0x0, 0x0)              = 14179 0
 1514/0x375f:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x375f:  ulock_wake(0x1000002, 0x16FBCF034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FBCF034, 0xC07)                 = 0 0
 1514/0x3761:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3760:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3760:  ulock_wake(0x1000002, 0x16FC5B034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FC5B034, 0x1F0B)                = 0 0
 1514/0x3763:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3762:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3762:  ulock_wake(0x1000002, 0x16FD73034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FD73034, 0x1D03)                = 0 0
 1514/0x3751:  write(0x1, "|mat| = 901320.000000\n\0", 0x16)             = 22 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028000, 0x16FBCF000)
= 1874653184 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028020, 0x16FC5B000)
= 1875226624 0
 1514/0x3764:  fork()            = 0 0
 1514/0x3764:  thread_selfid(0x0, 0x0, 0x0)              = 14180 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028040, 0x16FCE7000)
= 1875800064 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028060, 0x16FD73000)
= 1876373504 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028080, 0x16FDFF000)
= 1876946944 0
 1514/0x3765:  fork()            = 0 0
 1514/0x3766:  fork()            = 0 0
 1514/0x3765:  thread_selfid(0x0, 0x0, 0x0)              = 14181 0
 1514/0x3766:  thread_selfid(0x0, 0x0, 0x0)              = 14182 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x6000020280A0, 0x16FE8B000)
= 1877520384 0
 1514/0x3767:  fork()            = 0 0
 1514/0x3768:  fork()            = 0 0
 1514/0x3767:  thread_selfid(0x0, 0x0, 0x0)              = 14183 0
```

```
 1514/0x3768:  thread_selfid(0x0, 0x0, 0x0)              = 14184 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x6000020280C0, 0x16FF17000)
= 1878093824 0
 1514/0x3769:  fork()            = 0 0
 1514/0x376a:  fork()            = 0 0
 1514/0x3769:  thread_selfid(0x0, 0x0, 0x0)              = 14185 0
 1514/0x376a:  thread_selfid(0x0, 0x0, 0x0)              = 14186 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x6000020280E0, 0x16FFA3000)
= 1878667264 0
 1514/0x376b:  fork()            = 0 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028100, 0x17002F000)
= 1879240704 0
 1514/0x376b:  thread_selfid(0x0, 0x0, 0x0)              = 14187 0
 1514/0x376c:  fork()            = 0 0
 1514/0x3751:  bsdthread_create(0x1002B98F0, 0x600002028120, 0x1700BB000)
= 1879814144 0
 1514/0x376c:  thread_selfid(0x0, 0x0, 0x0)              = 14188 0
 1514/0x376d:  fork()            = 0 0
 1514/0x376d:  thread_selfid(0x0, 0x0, 0x0)              = 14189 0
 1514/0x376d:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3765:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3766:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x376a:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3767:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3768:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3764:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3764:  ulock_wake(0x1000002, 0x16FBCF034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FBCF034, 0x1D07)                = 0 0
 1514/0x3769:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x3769:  ulock_wake(0x1000002, 0x16FE8B034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FE8B034, 0x1C03)                = 0 0
 1514/0x376c:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x376b:  __disable_threadsignal(0x1, 0x0, 0x0)             = 0 0
 1514/0x376b:  ulock_wake(0x1000002, 0x16FFA3034, 0x0)           = 0 0
 1514/0x3751:  ulock_wait(0x1020002, 0x16FFA3034, 0xF03)                 = 0 0
 1514/0x3751:  write(0x1, "|mat| = 0.000000\n\0", 0x11)          = 17 0
```

Красным обозначены системные вызовы, произведенные процессом. Зеленым обозначены системные вызовы, ответственные за создание новых потоков и ожидания завершения новых потоков.

Вывод

В ходе выполнения лабораторной работы была составлена и отлажена программа на языке C++, вычисляющая детерминант матрицы в многопоточном режиме в операционной системе macos. Программа использует стандартные средства создания потоков операционной системы. Написаная функция корректно вычисляет детерминант матриц разного размера. В ходе работы программы утечки памяти не возникают.