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: Коваль М.Р.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 24.12.24

Москва, 2024

Постановка задачи

Вариант 10.

Решить систему линейных уравнений методом Гаусса.

Общий метод и алгоритм решения

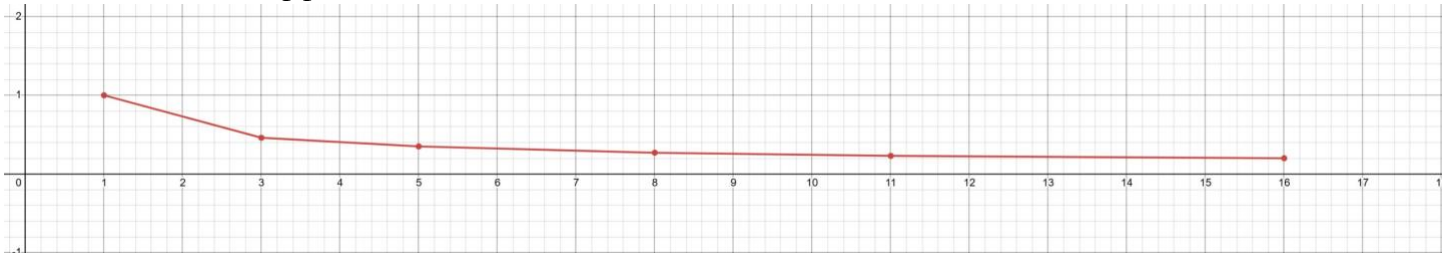
Использованные системные вызовы:

- int pthread_mutex_lock(pthread_mutex_t *mutex) - блокировка мьютекса;
- int pthread_mutex_unlock(pthread_mutex_t *mutex) - разблокировка мьютекса;
- int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void*(*start_routine) (void *), void *arg) - создание потока;
- int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr) - инициализирует мьютекс;
- int pthread_join(pthread_t thread, void ** retval) - ожидание завершения потока;
- void pthread_exit(void *retval) - завершает выполнение текущего потока;
- void exit(int status) - завершение программы с заданным кодом возврата.

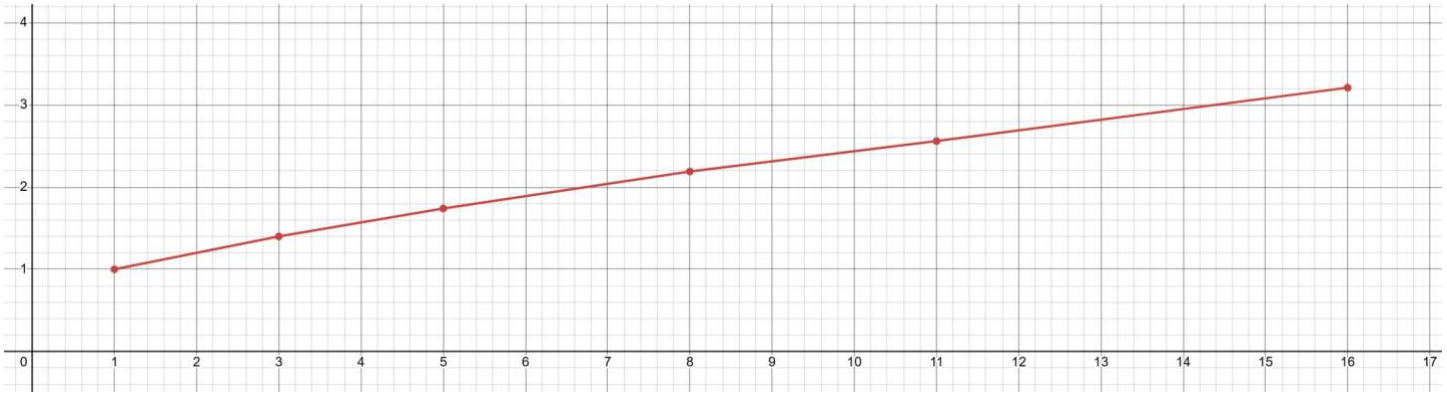
В данной лабораторной работе я написал программу для решения системы линейных уравнений Ax=b методом Гаусса, где сначала генерируется размер матрицы n и максимальное количество потоков, затем выделяется память для матрицы A, вектора b и вектора решения x, которые заполняются случайными значениями (значения одинаковы); матрица A разделяется на части, каждая из которых обрабатывается отдельным потоком; каждый поток выполняет прямой ход метода Гаусса для своей части матрицы; после завершения всех потоков основной поток ожидает их завершения, затем выполняется обратная подстановка для нахождения решения x, которое выводится на экран, и в конце освобождается выделенная память для матрицы A, вектора b и вектора решения x.

Число потоков	Время выполнения (мс)	Ускорение	Эффективность
1	1002	1	1,00
3	712	1,4	0,46
5	575	1,74	0,35
8	456	2,19	0,27
11	391	2,56	0,23
16	312	3,21	0,20

Изменение эффективности:



Изменение ускорения:



Код программы

main.c:

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

#define MAX_THREADS 16


typedef struct {
    int n;
    double *a;
    double *b;
    int start;
} data;


pthread_mutex_t mutex;


void* gauss(void* arg) {
    data* num = (data*)arg;
    double *a = num->a;
    double *b = num->b;
    // int k = num->start;
    int n = num->n;
    for (int k = 0; k < n; k++) {
        double p = a[k * n + k];
        for (int i = k + 1; i < n; i++) {
                pthread_mutex_lock(&mutex);
                double l = a[i * n + k] / p;
                for(int j = k; j < n; j++){
                    a[i * n + j] -= l * a[k * n + j];
                }
                b[i] -= l * b[k];
                pthread_mutex_unlock(&mutex);
            }
    }
    return NULL;
}

void back_way(int n, double *a, double *b, double *x) {
```

```
    for(int k = n - 1; k >= 0; k--){
        x[k] = b[k];
        for(int i = k + 1; i < n; i++){
            x[k] -= a[k * n + i] * x[i];
        }
        x[k] /= a[k * n + k];
        printf("x[%d] = %5.4f\n", k, x[k]);
    }
}

int main(int argc, char *argv[]){
    if (argc < 3) {
        const char msg[] = "error: failed, too many arguments\n";
        write(STDOUT_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    int n = atoi(argv[1]);
    int num_of_threads = atoi(argv[2]);
    if (pthread_mutex_init(&mutex, NULL) != 0) {
        perror("Mutex initialization error\n");
        exit(EXIT_FAILURE);
    }

    double *a = malloc(sizeof(*a) * n * n);
    double *b = malloc(sizeof(*b) * n);
    double *x = malloc(sizeof(*x) * n);

    // создание матрицы
    for (int i = 0; i < n; i++) {
        srand(i * (10 + 1));
        for (int j = 0; j < n; j++) {
            a[i * n + j] = rand() % 10 + 1;
        }
        b[i] = rand() % 10 + 1;
    }

    // отображение матрицы
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            printf("%5.2f ", a[i * n + j]);
        }printf(" | %5.2f\n", b[i]);
    }

    pthread_t threads[MAX_THREADS];
```

```
    data thread_data[MAX_THREADS];

    int chunks = n / num_of_threads;
    for(int i = 0; i < num_of_threads; i++){
        thread_data[i].n = n;
        thread_data[i].a = a;
        thread_data[i].b = b;
        thread_data[i].start = i * chunks;
        pthread_create(&threads[i], NULL, gauss, &thread_data[i]);
    }

    for (int i = 0; i < num_of_threads; i++) {
        pthread_join(threads[i], NULL);
    }

    back_way(n, a, b, x);

    free(a);
    free(b);
    free(x);

    pthread_mutex_destroy(&mutex);
    return 0;
}
```

Протокол работы программы

Тестирование

```
mk@MacBook-Air-Mete src % ./a.out 4 3
  1.00  2.00  6.00  3.00  |  2.00
  8.00  3.00  8.00  4.00  |  6.00
  5.00  5.00  5.00  7.00  |  4.00
  2.00 10.00  2.00 10.00  |  2.00
x[3] = 0.0556
x[2] = 0.2222
x[1] = -0.0000
x[0] = 0.5000
```

Dtruss du -h

SYSCALL(args) = return

44K .

munmap(0x1051E0000, 0x8C000) = 0 0

munmap(0x10526C000, 0x8000) = 0 0

munmap(0x105274000, 0x4000) = 0 0

munmap(0x105278000, 0x4000) = 0 0

munmap(0x10527C000, 0x50000) = 0 0

crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45

fsgetpath(0x16AFCF188, 0x400, 0x16AFCF168) = 12 0

fsgetpath(0x16AFCF198, 0x400, 0x16AFCF178) = 14 0

csrctl(0x0, 0x16AFCF59C, 0x4) = -1 Err#1

__mac_syscall(0x1926B9ACF, 0x2, 0x16AFCF4E0) = 0 0

csrctl(0x0, 0x16AFCF58C, 0x4) = -1 Err#1

__mac_syscall(0x1926B6902, 0x5A, 0x16AFCF520) = 0 0

sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16AFCEAA8, 0x16AFCEAA0, 0x1926B8553, 0xD) = 0 0

sysctl([CTL_KERN, 140, 0, 0, 0, 0] (2), 0x16AFCEB58, 0x16AFCEB50, 0x0, 0x0) = 0 0

open("/\0", 0x20100000, 0x0) = 3 0

openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0

dup(0x4, 0x0, 0x0) = 5 0

fstatat64(0x4, 0x16AFCE631, 0x16AFCE5A0) = 0 0

openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0

fcntl(0x6, 0x32, 0x16AFCE630) = 0 0

dup(0x6, 0x0, 0x0) = 7 0

dup(0x5, 0x0, 0x0) = 8 0

close(0x3) = 0 0

close(0x5) = 0 0

close(0x4) = 0 0

close(0x6) = 0 0

__mac_syscall(0x1926B9ACF, 0x2, 0x16AFCF020) = 0 0

shared_region_check_np(0x16AFCEC40, 0x0, 0x0) = 0 0

fsgetpath(0x16AFCF1A0, 0x400, 0x16AFCF0E8) = 82 0

fcntl(0x8, 0x32, 0x16AFCF1A0) = 0 0

close(0x8) = 0 0

close(0x7) = 0 0

getfsstat64(0x0, 0x0, 0x2) = 11 0

getfsstat64(0x10524A020, 0x5D28, 0x2) = 11 0

getattrlist("/\0", 0x16AFCF0D0, 0x16AFCF040) = 0 0

stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64e\0", 0x16AFCF430, 0x0) = 0 0

dtrace: error on enabled probe ID 1690 (ID 845: syscall::stat64:return): invalid address (0x0) in action #11 at DIF offset 12

stat64("/usr/lib/system/libdispatch.dylib\0", 0x16AFCCA30, 0x0) = -1 Err#2

stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16AFCC9E0, 0x0) = -1 Err#2

stat64("/usr/lib/system/libdispatch.dylib\0", 0x16AFCCA30, 0x0) = -1 Err#2

open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

ioctl(0x3, 0x80086804, 0x16AFCD938) = 0 0

close(0x3) = 0 0

stat64("/usr/bin/du\0", 0x16AFCD3B0, 0x0) = 0 0

open("/usr/bin/du\0", 0x0, 0x0) = 3 0

mmap(0x0, 0x21060, 0x1, 0x40002, 0x3, 0x0) = 0x10528C000 0

fcntl(0x3, 0x32, 0x16AFCD4C8) = 0 0

close(0x3) = 0 0

munmap(0x10528C000, 0x21060) = 0 0

mprotect(0x104E34000, 0x4000, 0x1) = -1 Err#13

shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

bsdthread_register(0x25410001929BCD2C, 0xAC7B8001929BCD20, 0x4000) = 1073746399 0

getpid(0x0, 0x0, 0x0) = 4078 0

shm_open(0x192857F51, 0x0, 0x6C2F7273) = 3 0

fstat64(0x3, 0x16AFCDD20, 0x0) = 0 0

mmap(0x0, 0x4000, 0x1, 0x40001, 0x3, 0x0) = 0x105294000 0

close(0x3) = 0 0

ioctl(0x2, 0x4004667A, 0x16AFCDDCC) = 0 0

mprotect(0x1052A0000, 0x4000, 0x0) = 0 0

mprotect(0x1052AC000, 0x4000, 0x0) = 0 0

mprotect(0x1052B0000, 0x4000, 0x0) = 0 0

mprotect(0x1052BC000, 0x4000, 0x0) = 0 0

mprotect(0x1052C0000, 0x4000, 0x0) = 0 0

mprotect(0x1052CC000, 0x4000, 0x0) = 0 0

mprotect(0x105298000, 0xA0, 0x1) = 0 0

mprotect(0x105298000, 0xA0, 0x3) = 0 0

mprotect(0x105298000, 0xA0, 0x1) = 0 0

mprotect(0x1052D0000, 0x4000, 0x1) = 0 0

mprotect(0x1052D4000, 0xA0, 0x1) = 0 0

mprotect(0x1052D4000, 0xA0, 0x3) = 0 0

mprotect(0x1052D4000, 0xA0, 0x1) = 0 0

mprotect(0x105298000, 0xA0, 0x3) = 0 0

mprotect(0x105298000, 0xA0, 0x1) = 0 0

mprotect(0x1052D0000, 0x4000, 0x3) = 0 0

mprotect(0x1052D0000, 0x4000, 0x1) = 0 0

objc_bp_assist_cfg_np(0x1925E9000, 0x80000018001C1048, 0x0) = -1 Err#5

issetugid(0x0, 0x0, 0x0) = 0 0

getentropy(0x16AFCD418, 0x20, 0x0) = 0 0

getattrlist("/usr/bin/du\0", 0x16AFCDCB0, 0x16AFCDCC8) = 0 0

access("/usr/bin\0", 0x4, 0x0) = 0 0

open("/usr/bin\0", 0x0, 0x0) = 3 0

fstat64(0x3, 0x158E045B0, 0x0) = 0 0

csrctl(0x0, 0x16AFCDEDC, 0x4) = 0 0

fcntl(0x3, 0x32, 0x16AFCDB98) = 0 0

close(0x3) = 0 0

open("/usr/bin/Info.plist\0", 0x0, 0x0) = -1 Err#2

proc_info(0x2, 0xFEE, 0xD) = 64 0

csops_audittoken(0xFEE, 0x10, 0x16AFCDF20) = 0 0

sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16AFCE278, 0x16AFCE270, 0x195DEED3D, 0x15) = 0 0

sysctl([CTL_KERN, 138, 0, 0, 0, 0] (2), 0x16AFCE308, 0x16AFCE300, 0x0, 0x0) = 0 0

csops(0xFEE, 0x0, 0x16AFCE3AC) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_COLLATE\0", 0x0, 0x0) = 3 0

fcntl_nocancel(0x3, 0x3, 0x0) = 0 0

getrlimit(0x1008, 0x16AFCEE68, 0x0) = 0 0

fstat64(0x3, 0x16AFCEDF0, 0x0) = 0 0

read_nocancel(0x3, "1.1A\n\0", 0x1000) = 2086 0

close_nocancel(0x3) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_CTYPE\0", 0x0, 0x0) = 3 0

fcntl_nocancel(0x3, 0x3, 0x0) = 0 0

fstat64(0x3, 0x16AFCEF10, 0x0) = 0 0

fstat64(0x3, 0x16AFCED00, 0x0) = 0 0

lseek(0x3, 0x0, 0x1) = 0 0

lseek(0x3, 0x0, 0x0) = 0 0

read_nocancel(0x3, "RuneMagAUTF-8\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "\0", 0x1000) = 4096 0

read_nocancel(0x3, "@\004\211\0", 0xF5D0) = 62928 0

close_nocancel(0x3) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_MONETARY\0", 0x0, 0x0) = 3 0

fstat64(0x3, 0x16AFCEF40, 0x0) = 0 0

read_nocancel(0x3, "USD \n$\n.\n,\n3;3\n\n-\n2\n2\n1\n0\n1\n0\n1\n1\n(\0", 0x22) = 34 0

close_nocancel(0x3) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_NUMERIC\0", 0x0, 0x0) = 3 0

fstat64(0x3, 0x16AFCEF40, 0x0) = 0 0

read_nocancel(0x3, ".\n,\n3;3\n@$\b\0", 0x8) = 8 0

close_nocancel(0x3) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_TIME\0", 0x0, 0x0) = 3 0

fstat64(0x3, 0x16AFCEF50, 0x0) = 0 0

read_nocancel(0x3, "Jan\nFeb\nMar\nApr\nMay\nJun\nJul\nAug\nSep\nOct\nNov\nDec\nJanuary\nFebruary\nMarch\nApril\nMay\nJune\nJuly\nAugust\nSeptember\nOctober\nNovember\nDecember\nSun\nMon\nTue\nWed\nThu\nFri\nSat\nSunday\nMonday\nTuesday\nWednesday\nThursday\nFriday\nSaturday\n%H:%M:%S\n%m/%d/%Y\n%a %b %e %X %Y\nAM\nP", 0x179) = 377 0

close_nocancel(0x3) = 0 0

open_nocancel("/usr/share/locale/en_US.UTF-8/LC_MESSAGES/LC_MESSAGES\0", 0x0, 0x0) = 3 0

fstat64(0x3, 0x16AFCEF50, 0x0) = 0 0

read_nocancel(0x3, "^[yYsS].*\n^[nN].*\n(\0", 0x12) = 18 0

close_nocancel(0x3) = 0 0

sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16AFCF498, 0x16AFCF490, 0x104E33D9B, 0x22) = 0 0

sysctl([CTL_VFS, 100, 102, 0, 0, 0] (3), 0x0, 0x0, 0x16AFCF5E0, 0x4) = 0 0

sigaction(0x1D, 0x16AFCF4C8, 0x16AFCF4F0) = 0 0

fstatat64(0xFFFFFFFFFFFFFFFE, 0x158E04C78, 0x158E04C80) = 0 0

getattrlist(".\0", 0x16AFCF550, 0x16AFCF5F0) = 0 0

open_nocancel(".\0", 0x1100004, 0x0) = 3 0

getattrlistbulk(0x3, 0x16AFCF398, 0x15900B000) = 3 0

getattrlistbulk(0x3, 0x16AFCF398, 0x15900B000) = 0 0

close_nocancel(0x3) = 0 0

fstat64(0x1, 0x16AFCF290, 0x0) = 0 0

ioctl(0x1, 0x4004667A, 0x16AFCF2DC) = 0 0

write_nocancel(0x1, " 44K\t.\n\0", 0x7) = 7 0

Вывод

В ходе выполнения лабораторной работы я освоил процесс распараллеливания программ на языке Си, а также научился синхронизировать потоки с использованием мьютексов. Сложным для меня было понимание, как строится функция rand в си, а также где именно распараллеливать процесс в решение СЛАУ методом Гаусса.