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-211Б-23

Студент: Королев И.А

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.12.24

Москва, 2024

Постановка задачи

Вариант 10.

Решить систему линейных уравнений методом Гаусса.

Общий метод и алгоритм решения

Использованные системные вызовы:

- **ssize_t read(int fd, void *buf, size_t count);** – читает данные из файлового дескриптора fd в буфер buf. В программе используется для чтения пользовательского ввода в родительском процессе и передачи данных через пайпы в дочерние процессы, а также для получения данных, обработанных дочерними процессами.
- **ssize_t write(int fd, const void *buf, size_t count);** – записывает данные из буфера buf в файловый дескриптор fd. Применяется для отправки данных между процессами через пайпы, а также для вывода на экран обработанных данных.
- **int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void *), void *arg);** - Создает новый поток. В main вызывается для создания потоков, выполняющих функцию gaussian_elimination_thread.
- **int pthread_join(pthread_t thread, void **retval);** - Ожидает завершения указанного потока. Используется в main для ожидания завершения всех потоков, выполняющих вычисления.
- **int pthread_barrier_init(pthread_barrier_t *barrier, const pthread_barrierattr_t *attr, unsigned count);** - Инициализирует барьер синхронизации для потоков. В main используется для синхронизации потоков во время вычислений методом Гаусса.
- **int pthread_barrier_wait(pthread_barrier_t *barrier);** - Останавливает выполнение потока до тех пор, пока все потоки, использующие данный барьер, не достигнут его. Вызван в функции gaussian_elimination_thread для синхронизации потоков на каждом этапе алгоритма.
- **int pthread_barrier_destroy(pthread_barrier_t *barrier);** - Уничтожает барьер, освобождая связанные с ним ресурсы. Используется в main после завершения всех вычислений.
- **int snprintf(char *str, size_t size, const char *format, ...);** - Форматирует строку и записывает её в буфер с ограничением по размеру. В программе используется для форматирования строк перед их записью через системный вызов write.

Код программы

parent.c

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

#define BUF_SIZE 4096

typedef struct {
```

```c
    double** matrix;
    int n;
    int current_row;  // Общая текущая строка
    pthread_barrier_t* barrier;
} ThreadArgs;

void read_matrix(const char* filename, double*** matrix, int* n) {
    int fd = open(filename, O_RDONLY);
    if (fd < 0) {
        const char* msg = "Error opening file\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(EXIT_FAILURE);
    }

    char buffer[BUF_SIZE];
    ssize_t bytes = read(fd, buffer, sizeof(buffer));
    if (bytes <= 0) {
        const char* msg = "Error reading file\n";
        write(STDERR_FILENO, msg, strlen(msg));
        close(fd);
        exit(EXIT_FAILURE);
    }
    close(fd);

    char* ptr = buffer;
    *n = strtol(ptr, &ptr, 10);

    *matrix = malloc(*n * sizeof(double*));
    for (int i = 0; i < *n; i++) {
        (*matrix)[i] = malloc((*n + 1) * sizeof(double));
        for (int j = 0; j <= *n; j++) {
            (*matrix)[i][j] = strtod(ptr, &ptr);
        }
    }
}

void print_matrix(double** matrix, int n) {
    char buf[BUF_SIZE];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= n; j++) {
            int len = snprintf(buf, sizeof(buf), "%.2lf ", matrix[i][j]);
            write(STDOUT_FILENO, buf, len);
        }
        write(STDOUT_FILENO, "\n", 1);
    }
    write(STDOUT_FILENO, "\n", 1);
}

void* gaussian_elimination_thread(void* args) {
    ThreadArgs* data = (ThreadArgs*)args;
    double** matrix = data->matrix;
    int n = data->n;

    for (int row = 0; row < n; row++) {
        if (data->current_row == row) {
            int max_row = row;
            for (int i = row + 1; i < n; i++) {
                if (fabs(matrix[i][row]) > fabs(matrix[max_row][row])) {
                    max_row = i;
                }
            }
            if (max_row != row) {
                for (int j = 0; j <= n; j++) {
                    double temp = matrix[row][j];
                    matrix[row][j] = matrix[max_row][j];
                    matrix[max_row][j] = temp;
                }
```

```
            }
            if (fabs(matrix[row][row]) < 1e-9) {
                const char* msg = "Error: Singular matrix. Cannot solve system.\n";
                write(STDERR_FILENO, msg, strlen(msg));
                exit(EXIT_FAILURE);
            }
        }
        pthread_barrier_wait(data->barrier);
        for (int i = row + 1; i < n; i++) {
            double factor = matrix[i][row] / matrix[row][row];
            for (int j = row; j <= n; j++) {
                matrix[i][j] -= factor * matrix[row][j];
            }
        }
        pthread_barrier_wait(data->barrier);
    }
    return NULL;
}

void back_substitution(double** matrix, double* solution, int n) {
    for (int i = n - 1; i >= 0; i--) {
        solution[i] = matrix[i][n];
        for (int j = i + 1; j < n; j++) {
            solution[i] -= matrix[i][j] * solution[j];
        }
        solution[i] /= matrix[i][i];
    }
}

int main(int argc, char* argv[]) {
    clock_t start = clock();
    if (argc != 3) {
        const char* msg = "Usage: ./gauss_solver <matrix_file> <max_threads>\n";
        write(STDERR_FILENO, msg, strlen(msg));
        return EXIT_FAILURE;
    }

    double** matrix;
    int n;
    read_matrix(argv[1], &matrix, &n);

    int max_threads = strtol(argv[2], NULL, 10);
    if (max_threads <= 0) {
        const char* msg = "Error: max_threads must be a positive integer\n";
        write(STDERR_FILENO, msg, strlen(msg));
        return EXIT_FAILURE;
    }

    pthread_t threads[max_threads];
    ThreadArgs thread_args[max_threads];
    pthread_barrier_t barrier;

    pthread_barrier_init(&barrier, NULL, max_threads);

    for (int i = 0; i < max_threads; i++) {
        thread_args[i].matrix = matrix;
        thread_args[i].n = n;
        thread_args[i].current_row = 0;
        thread_args[i].barrier = &barrier;

        pthread_create(&threads[i], NULL, gaussian_elimination_thread,
&thread_args[i]);
    }

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

```
    pthread_barrier_destroy(&barrier);

    print_matrix(matrix, n);

    double* solution = malloc(n * sizeof(double));
    back_substitution(matrix, solution, n);

    char buf[BUF_SIZE];
    for (int i = 0; i < n; i++) {
        int len = snprintf(buf, sizeof(buf), "x%d = %.6lf\n", i + 1, solution[i]);
        write(STDOUT_FILENO, buf, len);
    }

    for (int i = 0; i < n; i++) {
        free(matrix[i]);
    }
    free(matrix);
    free(solution);
    clock_t end = clock();
    double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    int len = snprintf(buf, sizeof(buf), "Time elapsed: %f\n", time_taken);
    write(STDOUT_FILENO, buf, len);
    return 0;
}
```

Протокол работы программы

root@DESKTOP-VOD4IPT:/mnt/d/ClionProjects/OSlabs# cd lab2/src

root@DESKTOP-VOD4IPT:/mnt/d/ClionProjects/OSlabs/lab2/src# gcc lab2.c -o a -lm

root@DESKTOP-VOD4IPT:/mnt/d/ClionProjects/OSlabs/lab2/src# ./a matrix2.txt 2

5.00 2.00 3.00 -1.00 7.00

0.00 -2.80 2.80 2.40 3.20

0.00 0.00 -1.50 3.29 6.71

0.00 0.00 0.00 0.90 -2.24

x1 = 12.105263

x2 = -13.157895

x3 = -9.894737

x4 = -2.473684

Time elapsed: 0.001435

Strace:

root@DESKTOP-VOD4IPT:/mnt/d/ClionProjects/OSlabs/lab2/src# strace ./a matrix2.txt 2

execve("./a", ["./a", "matrix2.txt", "2"], 0x7fffa0d911d0 /* 28 vars */) = 0

brk(NULL) = 0x55e9383f1000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe29499310) = -1 EINVAL (Invalid argument)

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5d40773000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=18647, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 18647, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5d4076e000

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\0P\237\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

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5d40545000

mprotect(0x7f5d4056d000, 2023424, PROT_NONE) = 0

mmap(0x7f5d4056d000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f5d4056d000

mmap(0x7f5d40702000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f5d40702000

mmap(0x7f5d4075b000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f5d4075b000

mmap(0x7f5d40761000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f5d40761000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5d40542000

arch_prctl(ARCH_SET_FS, 0x7f5d40542740) = 0

set_tid_address(0x7f5d40542a10)         = 12475

set_robust_list(0x7f5d40542a20, 24)     = 0

rseq(0x7f5d405430e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f5d4075b000, 16384, PROT_READ) = 0

mprotect(0x55e926e0d000, 4096, PROT_READ) = 0

mprotect(0x7f5d407ad000, 8192, PROT_READ) = 0
```

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

munmap(0x7f5d4076e000, 18647) = 0

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=4707000}) = 0

openat(AT_FDCWD, "matrix2.txt", O_RDONLY) = 3

read(3, "4\r\n3 2 -1 4 10\r\n2 -2 4 2 6\r\n1 0."..., 4096) = 54

close(3) = 0

getrandom("\xc4\x94\x6f\x6b\xf6\x36\x60\x41", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x55e9383f1000

brk(0x55e938412000) = 0x55e938412000

rt_sigaction(SIGRT_1, {sa_handler=0x7f5d405d6870, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f5d40587520}, 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) = 0x7f5d3fd41000

mprotect(0x7f5d3fd42000, 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=0x7f5d40541910, parent_tid=0x7f5d40541910, exit_signal=0, stack=0x7f5d3fd41000, stack_size=0x7fff00, tls=0x7f5d40541640} => {parent_tid=[12476]}, 88) = 12476

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f5d3f540000

mprotect(0x7f5d3f541000, 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=0x7f5d3fd40910, parent_tid=0x7f5d3fd40910, exit_signal=0, stack=0x7f5d3f540000, stack_size=0x7fff00, tls=0x7f5d3fd40640} => {parent_tid=[12477]}, 88) = 12477

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

futex(0x7f5d40541910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 12476, NULL, FUTEX_BITSET_MATCH_ANY) = 0

write(1, "5.00 ", 55.00) = 5

write(1, "2.00 ", 52.00) = 5

write(1, "3.00 ", 53.00) = 5

```
write(1, "-1.00 ", 6-1.00 )                = 6
write(1, "7.00 ", 57.00 )                  = 5
write(1, "\n", 1
)                   = 1
write(1, "0.00 ", 50.00 )                  = 5
write(1, "-2.80 ", 6-2.80 )                = 6
write(1, "2.80 ", 52.80 )                  = 5
write(1, "2.40 ", 52.40 )                  = 5
write(1, "3.20 ", 53.20 )                  = 5
write(1, "\n", 1
)                   = 1
write(1, "0.00 ", 50.00 )                  = 5
write(1, "0.00 ", 50.00 )                  = 5
write(1, "-1.50 ", 6-1.50 )                = 6
write(1, "3.29 ", 53.29 )                  = 5
write(1, "6.71 ", 56.71 )                  = 5
write(1, "\n", 1
)                   = 1
write(1, "0.00 ", 50.00 )                  = 5
write(1, "0.00 ", 50.00 )                  = 5
write(1, "0.00 ", 50.00 )                  = 5
write(1, "0.90 ", 50.90 )                  = 5
write(1, "-2.24 ", 6-2.24 )                = 6
write(1, "\n", 1
)                   = 1
write(1, "\n", 1
)                   = 1
write(1, "x1 = 12.105263\n", 15x1 = 12.105263
)       = 15
write(1, "x2 = -13.157895\n", 16x2 = -13.157895
)       = 16
write(1, "x3 = -9.894737\n", 15x3 = -9.894737
```

) = 15

write(1, "x4 = -2.473684\n", 15x4 = -2.473684

) = 15

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=11636700}) = 0

write(1, "Time elapsed: 0.006929\n", 23Time elapsed: 0.006929

) = 23

exit_group(0) = ?

+++ exited with 0 +++

Таблица зависимости времени от количества потоков:

Число потоков	Время исполнения (мс)	Ускорение	Эффективность
1	934	1	1
2	1072	0,87	0,435
3	1470	0,63	0,21
4	1723	0,54	0,135
5	1943	0,48	0,096
6	2111	0,44	0,048

Ускорение показывает во сколько раз применение параллельного алгоритма уменьшает время решения задачи по сравнению с последовательным алгоритмом.

Вывод

В ходе выполнения лабораторной работы была разработана программа, в которой, в зависимости от введенного числа, создается определенное количество потоков, и программа решает поставленную задачу. В ходе вычисления корней методом Гаусса они выводятся на экран. Также на экран выводится и матрица, приведенная к угловому виду. В ходе выполнения я столкнулся с трудностями реализации решения СЛАУ методом Гаусса, которые были успешно решены.