Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа:М8О-215Б-23

Студент: Венгер Ирина Витальевна Преподаватель: Миронов Е.С. Оценка:

Дата:

Москва, 2024

**Содержание**

1. Постановка задачи.
2. Общие сведения о программе.
3. Общий метод и алгоритм решения.
4. Код программы.
5. Демонстрация работы программы.
6. Вывод.

# Постановка задачи

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. Также 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 7) Два человека играют в кости. Правила игры следующие: каждый игрок делает бросок 2-ух костей K раз; побеждает тот, кто выбросил суммарно большее количество очков. Задача программы экспериментально определить шансы на победу каждого из игроков. На вход программе подается K, какой сейчас тур, сколько очков суммарно у каждого из игроков и количество экспериментов, которые должна произвести программа.

# Общие сведения о программе

Данная программа реализует многопоточный подход для симуляции игры в кости между двумя игроками. Программа состоит из одного файла main.c.

Описание структуры файла:

1. Подключенные заголовочные файлы:
   * <pthread.h> — обеспечивает поддержку работы с потоками.
   * <stdbool.h> — используется для работы с логическими типами данных.
   * <stdio.h> — ввод и вывод данных.
   * <stdlib.h> — динамическое выделение памяти и преобразование типов.
   * <string.h> — работа со строками.
   * <time.h> — замеры времени.
2. Основные функции:
   * void \*simulate\_game(void \*arg) — функция, выполняющая симуляцию одной части экспериментов в отдельном потоке.
   * int main(int argc, char \*argv[]) —

основная функция, где задаются параметры

программы, создаются потоки и собираются результаты.

1. Используемые системные вызовы:
   * pthread\_create — создание потока.
   * pthread\_join — ожидание завершения потока.
   * pthread\_mutex\_lock и pthread\_mutex\_unlock —

синхронизация доступа к общим данным.

# Общий метод и алгоритм решения

Программа моделирует игру двух игроков в кости. Для этого реализуется следующий алгоритм:

1. Входные параметры задают количество туров (K), стартовые очки игроков, общее число экспериментов и максимальное количество потоков.
2. Общее число экспериментов равномерно распределяется между потоками.
3. Каждый поток проводит свою часть экспериментов, используя функци simulate game.
4. Итоги каждого потока синхронизируются с помощью мьютексов и суммируются в общие переменные: количество побед первого игрока, второго и ничьи.
5. После завершения всех потоков оставшиеся эксперименты (если их число не делится нацело на количество потоков) выполняются в основном потоке.
6. Результаты обрабатываются и выводятся пользователю.

# Код программы

Код программы представлен в приложении 1.

# Использование утилиты strace

Скриншоты strace представлены в приложении 2.

# Демонстрация работы программы

(base) boopie@MacBook-Air-Irina lab2 % ./main 2 0 0 200 10

Всего игр было сыграно: 200

Игрок 1 выиграл 79 туров, шанс выигрыша: 39.50%

Игрок 2 выиграл 102 туров, шанс выигрыша: 51.00%

Ничья была в 19 турах, шанс ничьи: 9.50%

Время выполнения: 0.000123 секунд

Формула для ускорения (S(p)) и эффективности(E(p)):

𝑇1 = 𝑆(𝑝), где T1 – время выполнения на 1 потоке, а Tp – время выполнения на p потоках.

𝑇𝑝

𝑆(𝑝) = 𝐸(𝑝), где p – количество потоков.

𝑝

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоков | Время исполнения(м  с) | Ускорение | Эффективнос  ть |
| 1 | 1,61 | 1 | 1 |
| 2 | 2,01 | 0,8 | 0,4 |
| 4 | 1,83 | 0,88 | 0,22 |
| 8 | 1,72 | 0,94 | 0,12 |
| 12 | 1,29 | 1,25 | 0,10 |
| 16 | 1,78 | 0,90 | 0,06 |

# Вывод

В ходе лабораторной работы была создана программа на языке C, использующая многопоточный подход для проведения экспериментов. Удалось изучить:

* Основы работы с потоками через POSIX Threads.
* Синхронизацию потоков с использованием мьютексов.

Программа оказалась достаточно интересной, особенно анализ зависимости производительности от количества потоков. Основная сложность заключалась в правильной синхронизации потоков и управлении памятью.

**Приложение 1**

main.c

#include <pthread.h> #include <stdbool.h> #include <stdio.h>

#include <stdlib.h> #include <string.h> #include <time.h>

typedef struct { int K;

int experiments\_per\_thread; int start\_points\_first;

int start\_points\_second; int \*draws;

int \*first\_wins; int \*second\_wins;

pthread\_mutex\_t \*mutex;

} ThreadData;

void \*simulate\_game(void \*arg) {

ThreadData \*input\_data = (ThreadData \*)arg; int first\_wins = 0;

int second\_wins = 0; int draws = 0;

for (int i = 0; i < input\_data->experiments\_per\_thread; ++i) { int sum1 = input\_data->start\_points\_first;

int sum2 = input\_data->start\_points\_second; for (int j = 0; j < input\_data->K; ++j) {

sum1 += rand() % 6 + 1; // Случайное число от 1 до 6 sum2 += rand() % 6 + 1; // Случайное число от 1 до 6

}

if (sum1 > sum2) {

++first\_wins;

} else if (sum2 > sum1) {

++second\_wins;

} else {

++draws;

}

}

pthread\_mutex\_lock(input\_data->mutex);

\*(input\_data->first\_wins) += first\_wins;

\*(input\_data->second\_wins) += second\_wins;

\*(input\_data->draws) += draws;

pthread\_mutex\_unlock(input\_data->mutex);

return NULL;

}

int main(int argc, char \*argv[]) {

srand(time(NULL)); // Инициализация генератора случайных чисел

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC, &start); // Начало измерения

int max\_threads = 4;

int K = 4; // Сколько туров нужно провести

int start\_points\_first = 0; int start\_points\_second = 0;

int experiments = 4; // Кол-во экспериментов

pthread\_mutex\_t mutex;

pthread\_mutex\_init(&mutex, NULL); // Инициализация мьютекса

for (int i = 1; i < argc; ++i) { if (argv[i] == NULL) {

break;

} else if (i == 1) {

if (strcmp(argv[i], "--help") == 0 || strcmp(argv[i], "-h") == 0) {

printf("./game [K] [S1] [S2] [E]\nK - кол-во туров;\nS1, S2 - " "начальные значения очков игроков;\nE - кол-во

экспериментов, "

задать " "

"которые должна программа провести;\n\nFlags:\n\t-tr - "максимальное кол-во потоков (по умолчанию 4);\n\t-show - "показать также результаты игр.\n");

return 0;

}

K = atoi(argv[i]);

} else if (i == 2) {

start\_points\_first = atoi(argv[i]);

} else if (i == 3) {

start\_points\_second = atoi(argv[i]);

} else if (i == 4) {

experiments = atoi(argv[i]);

} else if (strcmp(argv[i], "-tr") == 0 && argv[i + 1]) { max\_threads = atoi(argv[i + 1]);

printf("Кол-во потоков установлено на %d\n", max\_threads);

++i;

}

}

int first\_wins = 0; int second\_wins = 0; int draws = 0;

pthread\_t threads[max\_threads];

ThreadData threads\_data[max\_threads];

int experiments\_per\_thread = experiments / max\_threads; int remaining\_experiments = experiments % max\_threads;

for (int i = 0; i < max\_threads; ++i) { threads\_data[i].K = K;

threads\_data[i].experiments\_per\_thread = experiments\_per\_thread + (i < remaining\_experiments ? 1 : 0);

threads\_data[i].start\_points\_first = start\_points\_first;

threads\_data[i].start\_points\_second = start\_points\_second; threads\_data[i].draws = &draws;

threads\_data[i].first\_wins = &first\_wins;

threads\_data[i].second\_wins = &second\_wins; threads\_data[i].mutex = &mutex;

pthread\_create(&threads[i], NULL, simulate\_game, &threads\_data[i]);

}

for (int i = 0; i < max\_threads; i++) { pthread\_join(threads[i], NULL);

}

pthread\_mutex\_destroy(&mutex);

clock\_gettime(CLOCK\_MONOTONIC, &end); // Конец измерения

double time\_taken = (end.tv\_sec - start.tv\_sec) + (end.tv\_nsec - start.tv\_nsec) / 1e9; // Время в секундах

printf("Всего игр было сыграно: %d\n", experiments);

printf("Игрок 1 выиграл %d туров, шанс выигрыша: %.2f%%\n", first\_wins, (first\_wins \* 100.0) / experiments);

printf("Игрок 2 выиграл %d туров, шанс выигрыша: %.2f%%\n", second\_wins, (second\_wins \* 100.0) / experiments);

printf("Ничья была в %d турах, шанс ничьи: %.2f%%\n", draws, (draws \* 100.0)

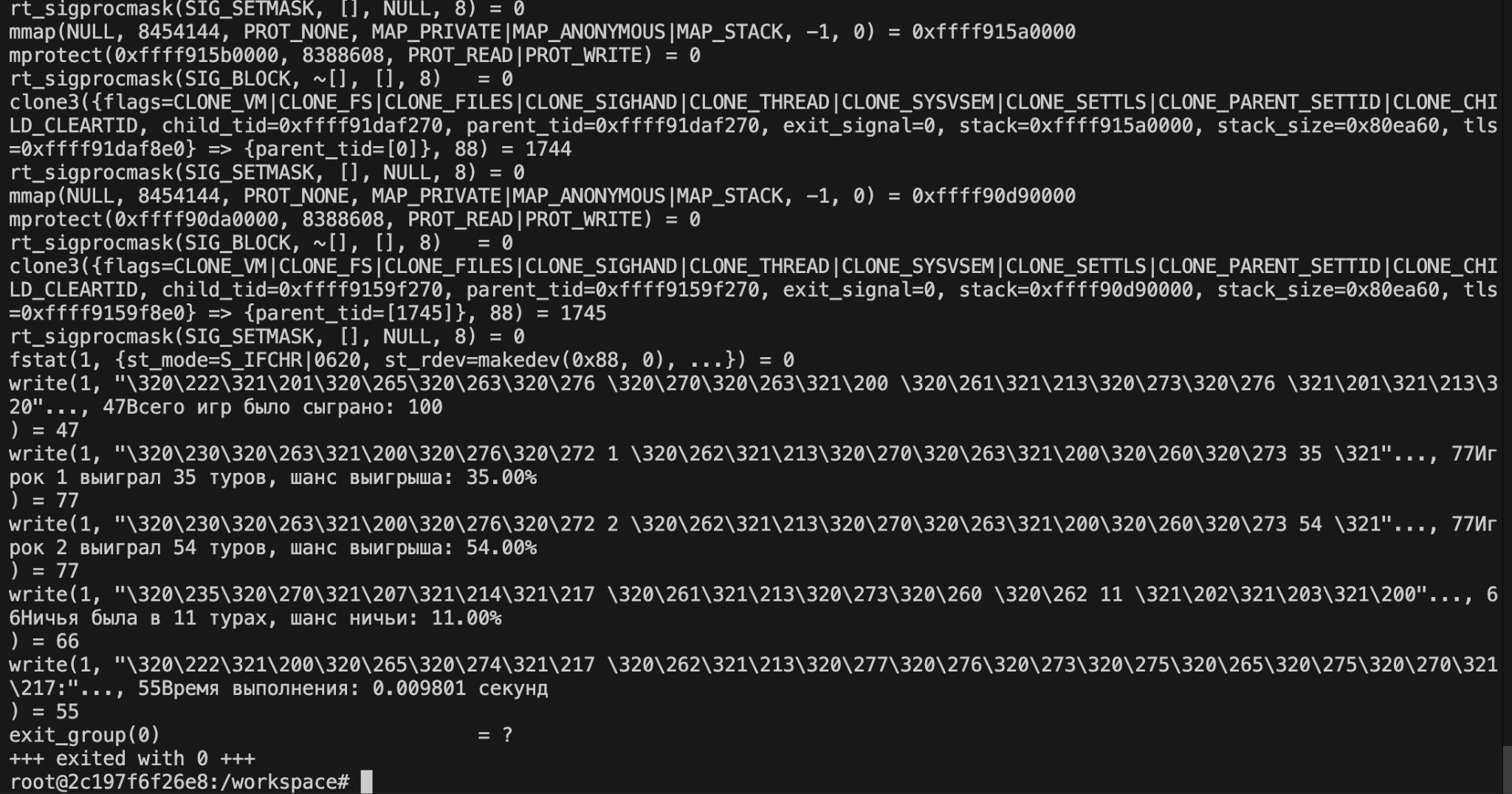
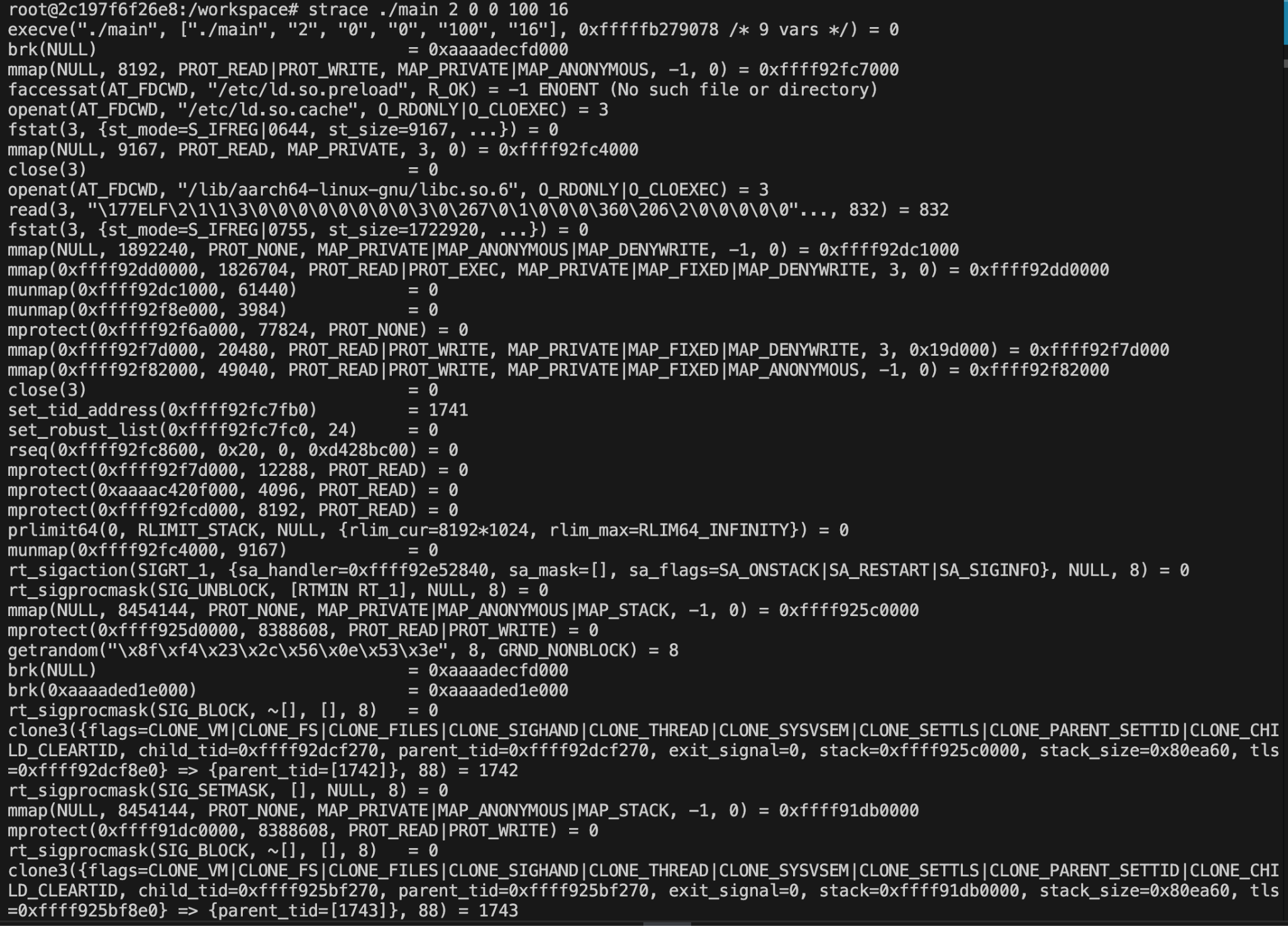
/ experiments);

printf("Время выполнения: %f секунд\n", time\_taken);

return 0;

}

**Приложение 2**

****