МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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»

Студент: Попов А. Д.
Группа: М8О-208Б-23
Преподаватель: Живалев Е. А.
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2024

Тема: Управление потоками и синхронизация в ОС

Цель работы: Целью работы является приобретение практических навыков в:

- Управлении потоками в операционной системе.
- Организации синхронизации между потоками для эффективного использования многопоточности.

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.

Задачи:

1. Разработать программу на языке Си, реализующую многопоточную симуляцию игры в кости для определения шансов на победу каждого игрока.
2. Ограничить максимальное количество одновременно работающих потоков с использованием заданного параметра.
3. Обеспечить корректную синхронизацию потоков с помощью стандартных средств операционной системы.
4. Провести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков.

Описание решения: Программное решение представляет собой многопоточную реализацию сортировки массива методом слияния. Основные компоненты программы:

- **Управление потоками:** Для создания потоков используется библиотека `pthread`. Максимальное количество одновременно работающих потоков задается пользователем в виде параметра запуска программы.
- **Синхронизация потоков:** Для синхронизации доступа к общим данным используется мьютекс. Результаты экспериментов (количество побед и ничьих) аккумулируются в глобальных переменных, защищенных мьютексом.
- **Алгоритм симуляции:** Каждый поток выполняет часть экспериментов, бросая кости для каждого игрока K раз. После завершения всех экспериментов вычисляются шансы на победу каждого игрока.

Программа функционирует следующим образом:

1. Пользователь вводит параметры игры: K (количество бросков костей), текущий тур, суммарные очки игроков, количество экспериментов и максимальное количество потоков.
2. Основной поток создает указанное количество рабочих потоков, каждый из которых выполняет часть экспериментов.
3. После завершения всех потоков выводятся результаты: количество побед каждого игрока, ничьих, а также шансы на победу в процентах.

Репозиторий: https://github.com/aldpopov/OS_labs/tree/master/LW2

Исходный код: Программное обеспечение состоит из следующих файлов:

1. **main.c:** Инициализация программы, ввод данных и запуск симуляции.
2. **lab_2_utils.c:** Реализация функций для симуляции игры и управления потоками.
3. **lab2.h:** Заголовочный файл с объявлениями функций и структур данных.

Пример кода:

```
// Функция симуляции игры в кости
void* simulate_games(void* arg) {
    GameData* data = (GameData*)arg;
    int local_player1_wins = 0;
    int local_player2_wins = 0;
    int base_experiments = data->experiments / data->thread_count;
    int extra_experiments = data->experiments % data->thread_count;
    int experiments_per_thread = base_experiments + (int)(pthread_self() <
extra_experiments);
    for (int i = 0; i < experiments_per_thread; ++i) {
        int player1_total = data->player1_score;
        int player2_total = data->player2_score;
        for (int j = 0; j < data->k; j++) {
            player1_total += roll_dice();
            player2_total += roll_dice();
        }
        if (player1_total > player2_total) {
            local_player1_wins++;
        } else if (player2_total > player1_total) {
            local_player2_wins++;
        } else {
            data->draws++;
        }
    }
```

```
    pthread_mutex_lock(&data->mutex);
    data->player1_wins += local_player1_wins;
    data->player2_wins += local_player2_wins;
    pthread_mutex_unlock(&data->mutex);
    return NULL;
}
```

Пример работы:

```
make run

Enter K, ROUND, P1SCORE, P2SCORE, EXPERIMENTS, MAXTHREADS (separated by spaces):

3 1 0 0 1000 4

Игрок 1 победил в 450 эксперимент(ов)

Игрок 2 победил в 400 эксперимент(ов)

Ничья в 150 эксперимент(ов)

Шанс на победу игрока 1: 45.00%

Шанс на победу игрока 2: 40.00%
```

Вывод: В ходе выполнения лабораторной работы была реализована многопоточная симуляция игры в кости. Программа корректно ограничивает количество одновременно работающих потоков и обеспечивает синхронизацию между ними. Результаты исследования показали, что увеличение числа потоков до определенного предела ускоряет выполнение программы, но при чрезмерном увеличении числа потоков эффективность снижается из-за накладных расходов на управление потоками. Полученные результаты соответствуют теоретическим ожиданиям.