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

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.
3. 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)):

$\frac{T1}{Tp} = S(p)$, где T1 – время выполнения на 1 потоке, а Tp – время выполнения на p потоках.

$\frac{S(p)}{p} = E(p)$, где p – количество потоков.

Число потоков	Время исполнения(мс)	Ускорение	Эффективность
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

```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, "
                       "которые должна программа провести;\n\nFlags:\n\t-tr -
задать "
                       "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

```
root@2c197f6f26e8:/workspace# strace ./main 2 0 0 100 16
execve("./main", ["./main", "2", "0", "0", "100", "16"], 0xfffffb279078 /* 9 vars */) = 0
brk(NULL)                               = 0xaaaadecfd000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff92fc7000
faccessat(AT_FDCWD, "/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
fstat(3, {st_mode=S_IFREG|0644, st_size=9167, ...}) = 0
mmap(NULL, 9167, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff92fc4000
close(3)                                = 0
openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0
mmap(NULL, 1892240, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffff92dc1000
mmap(0xffff92dd0000, 1826704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff92dd0000
munmap(0xffff92dc1000, 61440)           = 0
munmap(0xffff92f8e000, 3984)            = 0
mprotect(0xffff92f6a000, 77824, PROT_NONE) = 0
mmap(0xffff92f7d000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffff92f7d000
mmap(0xffff92f82000, 49040, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff92f82000
close(3)                                = 0
set_tid_address(0xffff92fc7fb0)         = 1741
set_robust_list(0xffff92fc7fc0, 24)     = 0
rseq(0xffff92fc8600, 0x20, 0, 0xd428bc00) = 0
mprotect(0xffff92f7d000, 12288, PROT_READ) = 0
mprotect(0xaaaac420f000, 4096, PROT_READ) = 0
mprotect(0xffff92fcd000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0xffff92fc4000, 9167)            = 0
rt_sigaction(SIGRT_1, {sa_handler=0xffff92e52840, sa_mask=[], sa_flags=SA_ONSTACK|SA_RESTART|SA_SIGINFO}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffff925c0000
mprotect(0xffff925d0000, 8388608, PROT_READ|PROT_WRITE) = 0
getrandom("\x8f\xf4\x23\x2c\x56\x0e\x53\x3e", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0xaaaadecfd000
brk(0xaaaaded1e000)                     = 0xaaaaded1e000
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_CHI
LD_CLEARTID, child_tid=0xffff92dcf270, parent_tid=0xffff92dcf270, exit_signal=0, stack=0xffff925c0000, stack_size=0x80ea60, tls
=0xffff92dcf8e0} => {parent_tid=[1742]}, 88) = 1742
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffff91db0000
mprotect(0xffff91dc0000, 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_CHI
LD_CLEARTID, child_tid=0xffff925bf270, parent_tid=0xffff925bf270, exit_signal=0, stack=0xffff91db0000, stack_size=0x80ea60, tls
=0xffff925bf8e0} => {parent_tid=[1743]}, 88) = 1743
```

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffff915a0000
mprotect(0xffff915b0000, 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_CHI
LD_CLEARTID, child_tid=0xffff91daf270, parent_tid=0xffff91daf270, exit_signal=0, stack=0xffff915a0000, stack_size=0x80ea60, tls
=0xffff91daf8e0} => {parent_tid=[0]}, 88) = 1744
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffff90d90000
mprotect(0xffff90da0000, 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_CHI
LD_CLEARTID, child_tid=0xffff9159f270, parent_tid=0xffff9159f270, exit_signal=0, stack=0xffff90d90000, stack_size=0x80ea60, tls
=0xffff9159f8e0} => {parent_tid=[1745]}, 88) = 1745
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "\320\222\321\201\320\265\320\263\320\276 \320\270\320\263\321\200 \320\261\321\213\320\273\320\276 \321\201\321\213\3
20"..., 47Всего игр было сыграно: 100
) = 47
write(1, "\320\230\320\263\321\200\320\276\320\272 1 \320\262\321\213\320\270\320\263\321\200\320\260\320\273 35 \321"..., 77Иг
рок 1 выиграл 35 туров, шанс выигрыша: 35.00%
) = 77
write(1, "\320\230\320\263\321\200\320\276\320\272 2 \320\262\321\213\320\270\320\263\321\200\320\260\320\273 54 \321"..., 77Иг
рок 2 выиграл 54 туров, шанс выигрыша: 54.00%
) = 77
write(1, "\320\235\320\270\321\207\321\214\321\217 \320\261\321\213\320\273\320\260 \320\262 11 \321\202\321\203\321\200"..., 6
6Ничья была в 11 турах, шанс ничьи: 11.00%
) = 66
write(1, "\320\222\321\200\320\265\320\274\321\217 \320\262\321\213\320\277\320\276\320\273\320\275\320\265\320\275\320\270\321
\217:"..., 55Время выполнения: 0.009801 секунд
) = 55
exit_group(0)                           = ?
+++ exited with 0 +++
root@2c197f6f26e8:/workspace#
```