Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**Тема работы**

**“Потоки”**

Студент: Борисов Ян Артурович

Группа: М8О-208Б-20

Вариант: 7

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/Yannikupy/OS/tree/master/os_lab3>

**Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы. Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

**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**

Программа представляет из себя один файл main.c.

**Общий метод и алгоритм решения**

Проводятся эксперименты с играми в кости. Если количество экспериментов, меньше чем заданное количество потоков, то мы выделяем поток на каждый эксперимент, если же больше чем количество потоков, то мы в цикле выделяем потоки на каждый эксперимент, дожидаемся их завершения и заново делаем то же самое, пока количество оставшихся экспериментов не будет <= количества потоков.

**Исходный код**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <stdbool.h>

struct Arguments {

int id; // ID потока

int num\_of\_throws;

int sum\_of\_points\_1;

int sum\_of\_points\_2;

bool win\_1;

bool win\_2;

};

typedef struct Arguments Args;

void\* thread\_func(void \*args) {

Args \*arg = (Args\*) args;

int id = arg->id;

int num\_of\_throws = arg->num\_of\_throws;

int sum\_of\_points\_1 = arg->sum\_of\_points\_1;

int sum\_of\_points\_2 = arg->sum\_of\_points\_2;

int player\_1\_num\_1;

int player\_1\_num\_2;

int player\_2\_num\_1;

int player\_2\_num\_2;

for (int index = 0; index < num\_of\_throws; index++) {

player\_1\_num\_1 = rand() % 6 + 1;

player\_1\_num\_2 = rand() % 6 + 1;

player\_2\_num\_1 = rand() % 6 + 1;

player\_2\_num\_2 = rand() % 6 + 1;

printf("ID of thread: %d**\n**", id);

printf("For the 1st player: %d and %d**\n**", player\_1\_num\_1, player\_1\_num\_2);

printf("For the 2nd player: %d and %d**\n**", player\_2\_num\_1, player\_2\_num\_2);

sum\_of\_points\_1 += (player\_1\_num\_1 + player\_1\_num\_2);

sum\_of\_points\_2 += (player\_2\_num\_1 + player\_2\_num\_2);

printf("**\n**");

}

if (sum\_of\_points\_1 > sum\_of\_points\_2) {

arg->win\_1 = true;

arg->win\_2 = false;

}

else {

arg->win\_1 = false;

arg->win\_2 = true;

}

return NULL;

}

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

if (argc != 7) {

fprintf(stderr, "Usage: %s, Num\_of\_throws (K), Num\_of\_tour, Points\_1, Points\_2, Num\_of\_experiments Max\_num\_of\_threads**\n**", argv[0]);

exit(EXIT\_FAILURE);

}

if (atoi(argv[1]) < 0 || atoi(argv[2]) < 0 || atoi(argv[3]) < 0 || atoi(argv[4]) < 0 || atoi(argv[5]) < 0 || atoi(argv[6]) < 0) {

fprintf(stderr, "Arguments %d, %d, %d, %d, %d, %d must be non negative**\n**", atoi(argv[1]), atoi(argv[2]), atoi(argv[3]), atoi(argv[4]), atoi(argv[5]), atoi(argv[6]));

exit(EXIT\_FAILURE);

}

int status;

int status\_addr;

int num\_of\_throws = atoi(argv[1]) - atoi(argv[2]);

int sum\_of\_points\_1 = atoi(argv[3]);

int sum\_of\_points\_2 = atoi(argv[4]);

int num\_of\_experiments = atoi(argv[5]);

size\_t num\_of\_threads = (size\_t)atoi(argv[6]);

bool need\_continue = true;

int num\_of\_plays = num\_of\_experiments;

int wins\_1 = 0;

int wins\_2 = 0;

float result\_for\_the\_1st;

float result\_for\_the\_2nd;

srand(time(NULL));

while(need\_continue) {

if (num\_of\_threads < num\_of\_experiments) {

pthread\_t \*threads = (pthread\_t \*) calloc(num\_of\_threads, sizeof(pthread\_t));

if (threads == NULL) {

fprintf(stderr, "in main: Can't allocate memory for threads**\n**");

exit(EXIT\_FAILURE);

}

// вводим значения для аргументов

Args args[num\_of\_threads];

for (int index = 0; index < num\_of\_threads; index++) {

args[index].id = index;

args[index].num\_of\_throws = num\_of\_throws;

args[index].sum\_of\_points\_1 = sum\_of\_points\_1;

args[index].sum\_of\_points\_2 = sum\_of\_points\_2;

}

// создаем новые потоки

for (int index = 0; index < num\_of\_threads; index++) {

status = pthread\_create(&threads[index], NULL, thread\_func, (void \*) &args[index]);

if (status != 0) {

fprintf(stderr, "main error: Can't create thread, status = %d**\n**", status);

exit(EXIT\_FAILURE);

}

}

// ждем завершения работы потоков

for (int index = 0; index < num\_of\_threads; index++) {

status = pthread\_join(threads[index], (void \*\*) &status\_addr);

if (status != 0) {

fprintf(stderr, "main error: Can't join thread, status = %d**\n**", status);

exit(EXIT\_FAILURE);

}

}

// рассчитываем количество побед у каждого игрока

for (int index = 0; index < num\_of\_threads; index++) {

if (args[index].win\_1) {

++wins\_1;

}

else {

++wins\_2;

}

}

num\_of\_experiments -= num\_of\_threads;

free(threads);

}

else {

pthread\_t \*threads = (pthread\_t \*) calloc(num\_of\_experiments, sizeof(pthread\_t));

if (threads == NULL) {

fprintf(stderr, "in main: Can't allocate memory for threads**\n**");

exit(EXIT\_FAILURE);

}

// вводим значения аргументов

Args args[num\_of\_experiments];

for (int index = 0; index < num\_of\_experiments; index++) {

args[index].id = index;

args[index].num\_of\_throws = num\_of\_throws;

args[index].sum\_of\_points\_1 = sum\_of\_points\_1;

args[index].sum\_of\_points\_2 = sum\_of\_points\_2;

}

// создаем новые потоки

for (int index = 0; index < num\_of\_experiments; index++) {

status = pthread\_create(&threads[index], NULL, thread\_func, (void \*) &args[index]);

if (status != 0) {

fprintf(stderr, "main error: Can't create thread, status = %d**\n**", status);

exit(EXIT\_FAILURE);

}

}

// ждем завершения работы потоков

for (int index = 0; index < num\_of\_experiments; index++) {

status = pthread\_join(threads[index], (void \*\*) &status\_addr);

if (status != 0) {

fprintf(stderr, "main error: Can't join thread, status = %d**\n**", status);

exit(EXIT\_FAILURE);

}

}

// рассчитываем количество побед у каждого игрока

for (int index = 0; index < num\_of\_experiments; index++) {

if (args[index].win\_1) {

++wins\_1;

}

else {

++wins\_2;

}

}

free(threads);

need\_continue = false;

}

}

// подсчитываем шансы на победу каждого игрока

result\_for\_the\_1st = (float)wins\_1 / (float)num\_of\_plays;

result\_for\_the\_2nd = 1.0 - result\_for\_the\_1st;

printf("Chances of the 1st player: %.2f**\n**", result\_for\_the\_1st);

printf("Chances of the 2nd player: %.2f**\n**", result\_for\_the\_2nd);

return 0;

}

**Демонстрация работы программы**

Ввод в консоль:

./a.out 5 2 4 7 3 6

ID of thread: 0

For the 1st player: 3 and 5

For the 2nd player: 3 and 4

ID of thread: 1

For the 1st player: 4 and 4

ID of thread: 0

For the 1st player: 4 and 3

For the 2nd player: 5 and 2

ID of thread: 0

For the 1st player: 5 and 3

For the 2nd player: 4 and 5

ID of thread: 2

For the 1st player: 5 and 4

For the 2nd player: 2 and 3

ID of thread: 2

For the 1st player: 6 and 5

For the 2nd player: 5 and 5

ID of thread: 2

For the 1st player: 3 and 6

For the 2nd player: 5 and 5

ID of thread: 1

For the 1st player: 5 and 1

For the 2nd player: 6 and 6

ID of thread: 1

For the 1st player: 5 and 6

For the 2nd player: 1 and 3

Chances of the 1st player: 0.33

Chances of the 2nd player: 0.67

**Выводы**

Проделав лабораторную работу, я приобрёл практические навыки в управлении потоками в ОС и обеспечил синхронизацию между ними, воспользовался утилитой strace.