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

Студент: Борисов Ян Артурович
Группа: М8О-208Б-20
Вариант: 7
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

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

```c
#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 nega-
tive\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 **) &sta-
tus_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 **) &sta-
tus_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:

```
yannik@DESKTOP-US1A6DR:/mnt/c/Users/boris/CLionProjects/OCs/OS/os_lab3/src$ strace -o file.txt ./a.out 5 2 4 7 3 6
ID of thread: 0
For the 1st player: 5 and 2
For the 2nd player: 2 and 4

ID of thread: 0
For the 1st player: 5 and 3
For the 2nd player: 1 and 1

ID of thread: 0
For the 1st player: 1 and 1
For the 2nd player: 6 and 3

ID of thread: 2
For the 1st player: 3 and 3
For the 2nd player: 1 and 1

ID of thread: 2
For the 1st player: 4 and 1
For the 2nd player: 5 and 4

ID of thread: 2
For the 1st player: 5 and 4
For the 2nd player: 6 and 4

ID of thread: 1
For the 1st player: 3 and 5
For the 2nd player: 2 and 6
```

```
ID of thread: 1
For the 1st player: 3 and 6
For the 2nd player: 2 and 5

ID of thread: 1
For the 1st player: 5 and 2
For the 2nd player: 3 and 2

Chances of the 1st player: 0.33
Chances of the 2nd player: 0.67
```

Содержимое файла file.txt:

```
execve("./a.out", ["./a.out", "5", "2", "4", "7", "3", "6"], 0x7fffe05ee8f0
/* 17 vars */) = 0
brk(NULL)                               = 0x7fffc53e7000
```

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffcc584210) = -1 EINVAL (Invalid argu-
ment)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or direc-
tory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=34820, ...}) = 0
mmap(NULL, 34820, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fed3b947000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= 3
read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832)
= 832
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"...,
68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fed3b980000
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"...,
68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fed3b924000
mmap(0x7fed3b92b000, 69632, PROT_READ|PROT_EXEC, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7fed3b92b000
mmap(0x7fed3b93c000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x18000) = 0x7fed3b93c000
mmap(0x7fed3b941000, 8192, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7fed3b941000
mmap(0x7fed3b943000, 13432, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fed3b943000
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\3\0>\0\1\0\0\0\360q\2\0\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\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848)
= 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X
>\263"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 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
pread64(3,
"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848)
= 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X
>\263"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fed3b730000
mprotect(0x7fed3b755000, 1847296, PROT_NONE) = 0
mmap(0x7fed3b755000, 1540096, PROT_READ|PROT_EXEC, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7fed3b755000
mmap(0x7fed3b8cd000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x19d000) = 0x7fed3b8cd000
```

```
mmap(0x7fed3b918000, 24576, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fed3b918000
mmap(0x7fed3b91e000, 13528, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fed3b91e000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fed3b720000
arch_prctl(ARCH_SET_FS, 0x7fed3b720740) = 0
mprotect(0x7fed3b918000, 12288, PROT_READ) = 0
mprotect(0x7fed3b941000, 4096, PROT_READ) = 0
mprotect(0x7fed3b987000, 4096, PROT_READ) = 0
mprotect(0x7fed3b97d000, 4096, PROT_READ) = 0
munmap(0x7fed3b947000, 34820)           = 0
set_tid_address(0x7fed3b720a10)         = 645
set_robust_list(0x7fed3b720a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7fed3b92bbf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fed3b9393c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fed3b92bc90, sa_mask=[], sa_flags=SA_RE-
STORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fed3b9393c0}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) =
0
time(NULL)                              = 1636615271 (2021-11-
11T10:21:11+0300)
brk(NULL)                               = 0x7fffc53e7000
brk(0x7fffc5408000)                     = 0x7fffc5408000
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7fed3af10000
mprotect(0x7fed3af11000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fed3b70ffb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYS-
VSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[646],
tls=0x7fed3b710700, child_tidptr=0x7fed3b7109d0) = 646
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7fed3a700000
mprotect(0x7fed3a701000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fed3aefffb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYS-
VSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[647],
tls=0x7fed3af00700, child_tidptr=0x7fed3af009d0) = 647
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7fed39ef0000
mprotect(0x7fed39ef1000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fed3a6effb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYS-
VSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[648],
tls=0x7fed3a6f0700, child_tidptr=0x7fed3a6f09d0) = 648
futex(0x7fed3b7109d0, FUTEX_WAIT, 646, NULL) = 0
futex(0x7fed3af009d0, FUTEX_WAIT, 647, NULL) = 0
write(1, "Chances of the 1st player: 0.33\n", 32) = 32
write(1, "Chances of the 2nd player: 0.67\n", 32) = 32
exit_group(0)                           = ?
+++ exited with 0 +++
```

Выводы

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