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**

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**

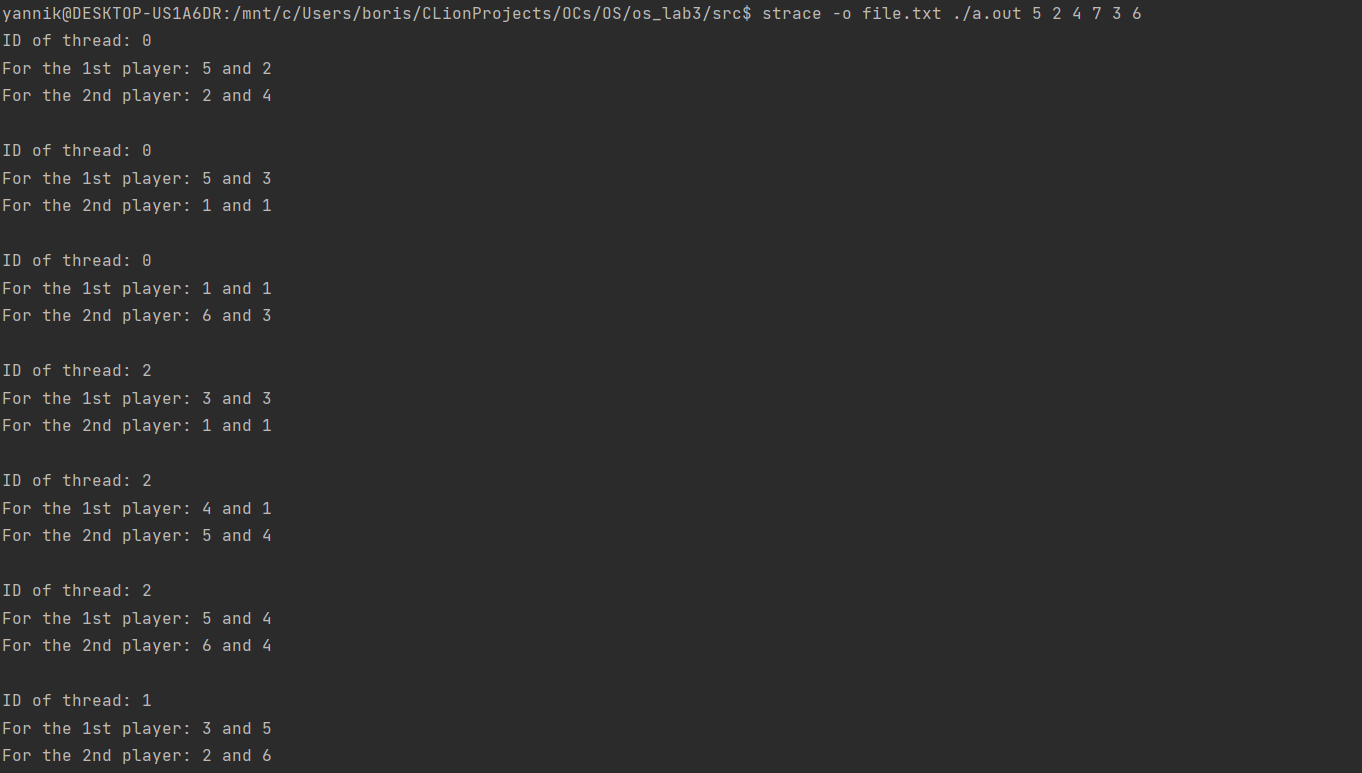
Проводятся эксперименты с играми в кости. Если количество экспериментов, меньше чем заданное количество потоков, то мы выделяем поток на каждый эксперимент, если же больше чем количество потоков, то мы в цикле выделяем потоки на каждый эксперимент, дожидаемся их завершения и заново делаем то же самое,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

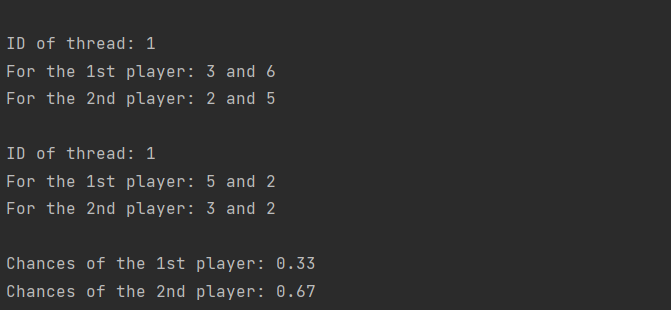
**Исходный код**

#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:





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 argument)  
access("/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=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\_PRIVATE|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\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7fed3b941000  
mmap(0x7fed3b943000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|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\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\_PRIVATE|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\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7fed3b918000  
mmap(0x7fed3b91e000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|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\_RESTORER|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\_SYSVSEM|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\_SYSVSEM|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\_SYSVSEM|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.