Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

УПРАВЛЕНИЕ ПОТОКАМИ

Студент: Недосекин Александр Александрович
Группа: М8О–209Б–22
Вариант: 7
Преподаватель: Гапонов Н.А.
Оценка:_____
Дата:_____
Подпись: _____

Москва, 2023.

Постановка задачи

Цель работы
Целью является приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание
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. Так же 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

Программа компилируется при помощи утилиты g++ main.cpp -pthread и запускается путем запуска ./a.out <NUMBER OF THREAD>. Также используется заголовочные файлы: iostream, thread, chrono. В программе используются следующие системные вызовы:

1. **pthread_self** – Функция возвращает дескриптор потока, в которой эта функция была вызвана: pthread_t pthread_self(void); Ошибки для функции pthread_self() не определены.
2. **pthread_create** – функция создает новый поток.Функция получает в качестве аргументов указатель на поток, переменную типа pthread_t, в которую, в случае удачного завершения сохраняет id потока. pthread_attr_t – атрибуты потока.
3. **pthread_join** – Функция позволяет потоку дождаться завершения

другого потока. В более сложной ситуации, когда требуется дождаться завершения нескольких потоков, можно воспользоваться переменными условия. Функция pthread_join блокирует вызывающий поток до завершения указанного потока.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы pthread, pthread_self, pthread_create, pthread_join.
2. Написать программу, которая будет удовлетворять заданию варианта.
3. Использовать pthread_create для распараллеливания выполнения программы.
4. При помощи функции threadFunc реализовать вычисление бросков кубиков игроками за поток.
5. Создать таймер при помощи библиотеки chrono и замерить время выполнения программы.
6. Скомпилировать обе программы при помощи g++ main.cpp -pthread и запустить ./a.out <NUMBER OF THREADS>.

Основные файлы программы

main.cpp:

```
#include <iostream>
#include <thread>
#include <chrono>

typedef struct _thread_data {
    int numberOfPointsFirst;
    int
    numberOfPointsSecond;
    int plays;
    int tour;
    int experements;
    int
    winChanceFirst;
    int winChanceSecond;
} threadData;

void* threadFunc(void* arg)
{
    threadData* tData =
    (threadData*)arg; unsigned int seed;
    seed = pthread_self();

    int firstP, secondP;
    for (int i = 0; i < tData->experements; ++i) {
        firstP = tData->numberOfPointsFirst;
        secondP = tData->numberOfPointsSecond;
        for (int j = 0; j < tData->plays - tData->tour + 1; ++j) { // + 1, тк индексация с 0
            firstP += rand_r(&seed) % 6 + 1;
            firstP += rand_r(&seed) % 6 + 1;
            secondP += rand_r(&seed) % 6 + 1;
            secondP += rand_r(&seed) % 6 + 1;
        }
```

```
if (firstP > secondP) {
```

```
            tData->winChanceFirst++;
        }

        if (secondP > firstP) {
            tData->winChanceSecond++;
        }
    }

    return 0;
}

int main(int argc, char* argv[])
{
    if (argc != 2) {
        std::cerr << "\tError! Program must have only 1 key\n EXAMPLE:\n \t"
                << argv[0] << " <NUMBER_OF_THREADS>\n";
        exit(EXIT_FAILURE);
    }

    clock_t startTime, endTime;
    float timer;
    startTime = clock();

    int numberOfThreads = atoi(argv[1]);

    int numberOfPointsFirst, numberOfPointsSecond,
    plays, tour, experements;
    float percentWinsFirst = 0, percentWinsSecond = 0;

    std::cout << "1) Enter number of points first player: ";
    std::cin >> numberOfPointsFirst;
    std::cout << "2) Enter number of points second player: ";
    std::cin >> numberOfPointsSecond;
    std::cout << "3) Enter the number of this tour: ";
    std::cin >> tour;
    std::cout << "4) Enter number of throws (K): ";
    std::cin >> plays;
    std::cout << "5) Enter number of experements: ";
    std::cin >> experements;

    int numbOfExperemForOneThread = experements / numberOfThreads;

    threadData tData[numberOfThreads];

    for (int i = 0; i < numberOfThreads; ++i) {
        tData[i].numberOfPointsFirst = numberOfPointsFirst;
        tData[i].numberOfPointsSecond =
        numberOfPointsSecond; tData[i].plays = plays;
        tData[i].tour = tour;

        if (i == numberOfThreads - 1) {
            tData[i].experements = numbOfExperemForOneThread + experements % numberOfThreads;
        } else {
            tData[i].experements = numbOfExperemForOneThread;
        }

        tData[i].winChanceFirst = 0;
        tData[i].winChanceSecond = 0;
    }
```

```
    pthread_t th[numberOfThreads];

    for (int i = 0; i < numberOfThreads; ++i) {
        if (pthread_create(&th[i], NULL, threadFunc, &tData[i]) != 0) {
            std::cerr << "\tError! Can'not create thread # " << i << std::endl;
            exit(EXIT_FAILURE);
        }
    }

    for (int i = 0; i < numberOfThreads; ++i) {
        if (pthread_join(th[i], NULL) != 0) {
            std::cerr << "Error! Can'not join thread # " << i << std::endl;
            break;
        }

        threadData* res = &tData[i];
        percentWinsFirst += res->winChanceFirst;
        percentWinsSecond += res->winChanceSecond;
    }

    std::cout << "Probability of the first player to win: "
              << percentWinsFirst / experements << std::endl;
    std::cout << "Probability of the second player to win: "
              << percentWinsSecond / experements << std::endl;

    endTime = clock();
    timer = endTime - startTime;
    std::cout << "Time: " << timer / CLOCKS_PER_SEC << std::endl;

    return 0;
}
```

Пример работы

```
aleksandr@dots:~/labsOC/lab2var7$ ./main 1

1) Enter number of points first player: 0

2) Enter number of points second player: 0

3) Enter the number of this tour: 1

4) Enter number of throws (K): 1000

5) Enter number of experements: 10000

Probability of the first player to win:
0.5013

Probability of the second player to win:
0.4945
```

```
Time: 0.200807
aleksandr@dots:~/labsOC/lab2var7$ ./main 100
1) Enter number of points first player: 0
2) Enter number of points second player: 0
3) Enter the number of this tour: 1
4) Enter number of throws (K): 1000
5) Enter number of experements: 10000
Probability of the first player to win:
0.4958
Probability of the second player to win:
0.5012
Time: 0.271232
aleksandr@dots:~/labsOC/lab2var7$ ./main
10000
1) Enter number of points first player: 0
2) Enter number of points second player: 0
3) Enter the number of this tour: 1
4) Enter number of throws (K): 1000
5) Enter number of experements: 10000
Probability of the first player to win:
0.4939
Probability of the second player to win:
0.502
Time: 1.04665
```

STRACE

```
aleksandr@dots:~/labsOC/lab2var7$
strace ./main 10
execve("./main", ["./main", "10"],
0x7ffc5f2075b8 /* 45 vars */) = 0
brk(NULL)
= 0x557b0bafc000
arch_prctl(0x3001 /* ARCH_??? */,
0x7ffec3f6cb30) = -1 EINVAL
```

```
(Недопустимый аргумент)

mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f4adfe13000

access("/etc/ld.so.preload", R_OK)
= -1 ENOENT (Нет такого файла или
каталога)

openat(AT_FDCWD, "/etc/ld.so.cache",
O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "",
{st_mode=S_IFREG|0644, st_size=68703,
...}, AT_EMPTY_PATH) = 0

mmap(NULL, 68703, PROT_READ,
MAP_PRIVATE, 3, 0) = 0x7f4adfe02000

close(3)
= 0

openat(AT_FDCWD,
"/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0"..., 832) =
832

newfstatat(3, "",
{st_mode=S_IFREG|0644, st_size=2260296,
...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2275520, PROT_READ,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f4adfa00000

mprotect(0x7f4adfa9a000, 1576960,
PROT_NONE) = 0

mmap(0x7f4adfa9a000, 1118208,
PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x9a000) = 0x7f4adfa9a000
```

```
mmap(0x7f4adfbab000, 454656, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1ab000) = 0x7f4adfbab000

mmap(0x7f4adfc1b000, 57344,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x21a000) = 0x7f4adfc1b000

mmap(0x7f4adfc29000, 10432,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f4adfc29000

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\0P\237\2\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
\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3
\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\37
4\204(\337f#\315I\214\234\f\256\271\32"
..., 68, 896) = 68

newfstatat(3, "",
{st_mode=S_IFREG|0755, st_size=2216304,
...}, AT_EMPTY_PATH) = 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

mmap(NULL, 2260560, PROT_READ,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f4adf600000

mmap(0x7f4adf628000, 1658880,
PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7f4adf628000

mmap(0x7f4adf7bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f4adf7bd000

mmap(0x7f4adf815000, 24576,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x214000) = 0x7f4adf815000

mmap(0x7f4adf81b000, 52816,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f4adf81b000

close(3)
= 0

openat(AT_FDCWD,
"/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0"..., 832) =
832

newfstatat(3, "",
{st_mode=S_IFREG|0644, st_size=940560,
...}, AT_EMPTY_PATH) = 0

mmap(NULL, 942344, PROT_READ,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f4adfd1b000

mmap(0x7f4adfd29000, 507904,
```

```
PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe000) = 0x7f4adfd29000

mmap(0x7f4adfda5000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8a000) = 0x7f4adfda5000

mmap(0x7f4adfe00000, 8192,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe4000) = 0x7f4adfe00000

close(3)
= 0

openat(AT_FDCWD,
"/lib/x86_64-linux-gnu/libgcc_s.so.1",
O_RDONLY|O_CLOEXEC) = 3

read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0
\1\0\0\0\0\0\0\0\0\0\0"..., 832) =
832

newfstatat(3, "",
{st_mode=S_IFREG|0644, st_size=125488,
...}, AT_EMPTY_PATH) = 0

mmap(NULL, 127720, PROT_READ,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f4adfcfb000

mmap(0x7f4adfcfe000, 94208,
PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f4adfcfe000

mmap(0x7f4adfd15000, 16384, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1a000) = 0x7f4adfd15000

mmap(0x7f4adfd19000, 8192,
PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1d000) = 0x7f4adfd19000
```

```
close(3)
= 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f4adfcf9000

arch_prctl(ARCH_SET_FS, 0x7f4adfcfa3c0)
= 0

set_tid_address(0x7f4adfcfa690)
= 4718

set_robust_list(0x7f4adfcfa6a0, 24)
= 0

rseq(0x7f4adfcfad60, 0x20, 0,
0x53053053) = 0

mprotect(0x7f4adf815000, 16384,
PROT_READ) = 0

mprotect(0x7f4adfd19000, 4096,
PROT_READ) = 0

mprotect(0x7f4adfe00000, 4096,
PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f4adfcf7000

mprotect(0x7f4adfc1b000, 45056,
PROT_READ) = 0

mprotect(0x557b0b610000, 4096,
PROT_READ) = 0

mprotect(0x7f4adfe4d000, 8192,
PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL,
{rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f4adfe02000, 68703)
= 0

getrandom("\x14\x00\x4d\x4a\xce\x1c\x0b
\x31", 8, GRND_NONBLOCK) = 8
```

```
brk(NULL)
= 0x557b0bafc000

brk(0x557b0bb1d000)
= 0x557b0bb1d000

futex(0x7f4adfc2977c,
FUTEX_WAKE_PRIVATE, 2147483647) = 0

clock_gettime(CLOCK_PROCESS_CPUTIME_ID,
{tv_sec=0, tv_nsec=2135287}) = 0

newfstatat(1, "",
{st_mode=S_IFCHR|0620,
st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

write(1, "1) Enter number of points
first "..., 401) Enter number of points
first player: ) = 40

newfstatat(0, "",
{st_mode=S_IFCHR|0620,
st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

read(0, 0

"0\n", 1024)                    = 2

write(1, "2) Enter number of points
second"..., 412) Enter number of points
second player: ) = 41

read(0, 0

"0\n", 1024)                    = 2

write(1, "3) Enter the number of this
tour"..., 343) Enter the number of this
tour: ) = 34

read(0, 1

"1\n", 1024)                    = 2

write(1, "4) Enter number of throws
(K): ", 314) Enter number of throws
(K): ) = 31
```

```
read(0, 1000
"1000\n", 1024)                 = 5
write(1, "5) Enter number of
experements: ", 325) Enter number of
experements: ) = 32
read(0, 10000
"10000\n", 1024)                = 6
rt_sigaction(SIGRT_1,
{sa_handler=0x7f4adf691870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_REST
ART|SA_SIGINFO,
sa_restorer=0x7f4adf642520}, NULL, 8) =
0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN
RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adedff000
mprotect(0x7f4adee00000, 8388608,
PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adf5ff910,
parent_tid=0x7f4adf5ff910,
exit_signal=0, stack=0x7f4adedff000,
stack_size=0x7fff00,
tls=0x7f4adf5ff640} =>
{parent_tid=[4720]}, 88) = 4720
rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0
mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
```

```
-1, 0) = 0x7f4ade5fe000

mprotect(0x7f4ade5ff000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adedfe910,
parent_tid=0x7f4adedfe910,
exit_signal=0, stack=0x7f4ade5fe000,
stack_size=0x7fff00,
tls=0x7f4adedfe640} =>
{parent_tid=[4721]}, 88) = 4721

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adddfd000

mprotect(0x7f4adddfe000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4ade5fd910,
parent_tid=0x7f4ade5fd910,
exit_signal=0, stack=0x7f4adddfd000,
stack_size=0x7fff00,
tls=0x7f4ade5fd640} =>
{parent_tid=[4722]}, 88) = 4722

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0
```

```
mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4add5fc000

mprotect(0x7f4add5fd000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adddfc910,
parent_tid=0x7f4adddfc910,
exit_signal=0, stack=0x7f4add5fc000,
stack_size=0x7fff00,
tls=0x7f4adddfc640} =>
{parent_tid=[4723]}, 88) = 4723

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adcdfb000

mprotect(0x7f4adcdfc000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4add5fb910,
parent_tid=0x7f4add5fb910,
exit_signal=0, stack=0x7f4adcdfb000,
stack_size=0x7fff00,
tls=0x7f4add5fb640} =>
{parent_tid=[4724]}, 88) = 4724

rt_sigprocmask(SIG_SETMASK, [], NULL,
```

```
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adc5fa000

mprotect(0x7f4adc5fb000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adcdfa910,
parent_tid=0x7f4adcdfa910,
exit_signal=0, stack=0x7f4adc5fa000,
stack_size=0x7fff00,
tls=0x7f4adcdfa640} =>
{parent_tid=[4725]}, 88) = 4725

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adbdf9000

mprotect(0x7f4adbdfa000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adc5f9910,
parent_tid=0x7f4adc5f9910,
exit_signal=0, stack=0x7f4adbdf9000,
stack_size=0x7fff00,
tls=0x7f4adc5f9640} =>
{parent_tid=[4726]}, 88) = 4726
```

```
rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adb5f8000

mprotect(0x7f4adb5f9000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adbdf8910,
parent_tid=0x7f4adbdf8910,
exit_signal=0, stack=0x7f4adb5f8000,
stack_size=0x7fff00,
tls=0x7f4adbdf8640} =>
{parent_tid=[4727]}, 88) = 4727

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4adadf7000

mprotect(0x7f4adadf8000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adb5f7910,
parent_tid=0x7f4adb5f7910,
exit_signal=0, stack=0x7f4adadf7000,
stack_size=0x7fff00,
tls=0x7f4adb5f7640} =>
```

```
{parent_tid=[4728]}, 88) = 4728

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f4ada5f6000

mprotect(0x7f4ada5f7000, 8388608,
PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)
= 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_F
ILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID
|CLONE_CHILD_CLEARTID,
child_tid=0x7f4adadf6910,
parent_tid=0x7f4adadf6910,
exit_signal=0, stack=0x7f4ada5f6000,
stack_size=0x7fff00,
tls=0x7f4adadf6640} =>
{parent_tid=[4729]}, 88) = 4729

rt_sigprocmask(SIG_SETMASK, [], NULL,
8) = 0

futex(0x7f4adf5ff910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4720, NULL, FUTEX_BITSET_MATCH_ANY) = 0

futex(0x7f4adedfe910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4721, NULL, FUTEX_BITSET_MATCH_ANY) = 0

futex(0x7f4adddfc910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4723, NULL, FUTEX_BITSET_MATCH_ANY) = 0

futex(0x7f4add5fb910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4724, NULL, FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f4adedff000, 8392704)
= 0
```

```
munmap(0x7f4ade5fe000, 8392704)
= 0

futex(0x7f4adc5f9910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4726, NULL, FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f4adddfd000, 8392704)
= 0

munmap(0x7f4add5fc000, 8392704)
= 0

munmap(0x7f4adcdfb000, 8392704)
= 0

futex(0x7f4adadf6910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4729, NULL, FUTEX_BITSET_MATCH_ANY) =
-1 EAGAIN (Ресурс временно недоступен)

munmap(0x7f4adc5fa000, 8392704)
= 0

write(1, "Probability of the first
player "..., 47Probability of the first
player to win: 0.4993

) = 47

write(1, "Probability of the second
player"..., 48Probability of the second
player to win: 0.4963

) = 48

clock_gettime(CLOCK_PROCESS_CPUTIME_ID,
{tv_sec=0, tv_nsec=255137302}) = 0

write(1, "Time: 0.253002\n", 15Time:
0.253002

)        = 15

lseek(0, -1, SEEK_CUR)
= -1 ESPIPE (Недопустимая операция
смещения)

exit_group(0)
= ?
```

```
+++ exited with 0 +++
```

Вывод

Во второй лабораторной работе я научился работать с потоками операционной системы. Изучив принципы работы потоков на низкоуровневом языке я реализовал вариант работы и сделал так, чтобы можно было выполнять вычисления, как на одном потоке, так и на нескольких. Используя системные вызовы я смог распараллелить программу и подсчитать, при помощи chrono, время выполнения программы. Как можно заметить время выполнения на одном потоке большого количества вычисления производится медленнее, чем на 100 потоках. Умение работать с потоками позволит в будущем более фундаментально понимать принципы работы много поточных программ.