Московский Авиационный Институт

(Национальный Исследовательский

Университет)

Институт №8 “Компьютерные науки и прикладная математика” Кафедра №806 “Вычислительная математика и

программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М8О-213Б-23

Студент: Марьин Д. А.

Преподаватель: Бахарев В.Д.

Оценка:

Дата: 06.11.24

Москва, 2024

**Постановка задачи**

**Вариант 7.**

**Два человека играют в кости. Правила игры следующие: каждый игрок делает бросок 2-ух костей К раз; побеждает тот, кто выбросил суммарно большее количество очков. Задача программы экспериментально определить шансы на победу каждого из игроков. На вход программе подается К, какой сейчас тур, сколько очков суммарно у каждого из игроков и количество экспериментов, которые должна произвести программа**

**Общий метод и алгоритм решения**

Использованные системные вызовы:

• write() – записываем число байт из буфера в указанный файловый дескриптор

• pthread\_create() – создаем новый поток с атрибутами

• pthread\_join() – ожидаем завершение потока

Создадим функцию которая будет производить необходимое нам количество экспериментов. Разделим введенное пользователем количество экспериментов между потоками и будем запускать функцию в каждом потоке. Результатом работы функции будет суммарное количество очков за все эксперименты проведенные в потоке. После объединения потоков суммируем счет и считаем вероятности.

| Количество потоков | Таблица (секунд) |
| --- | --- |
| 1 | 2,7 |
| 2 | 1,4 |
| 7 | 0,5 |
| 12 | 0,3 |

**Код программы**

**main.c**

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <pthread.h>

#include <time.h>

#define MAX\_THREADS 12

int validate\_nums(int amount, char\* nums[]) {

for (int j = 0; j <= amount; ++j) {

int len\_str = strlen(nums[j]);

for (int i = 0; i < len\_str; ++i) {

if ('0' > nums[j][i] || nums[j][i] > '9') {

return 0;

}

}

}

return 1;

}

void float\_to\_string(long double value, char \*buffer, int len, int precision) {

for (int i = 0; i < len; ++i) {

buffer[i] = '\0';

}

if (value < 0) {

\*buffer++ = '-';

value = -value;

}

int integerPart = (int)value;

float fractionalPart = value - integerPart;

char \*intPtr = buffer;

if (integerPart == 0) {

\*intPtr++ = '0';

} else {

char temp[20];

int i = 0;

while (integerPart > 0) {

temp[i++] = (integerPart % 10) + '0';

integerPart /= 10;

}

while (i > 0) {

\*intPtr++ = temp[--i];

}

}

\*intPtr++ = '.';

for (int i = 0; i < precision; i++) {

fractionalPart \*= 10;

int fractionalDigit = (int)fractionalPart;

\*intPtr++ = fractionalDigit + '0';

fractionalPart -= fractionalDigit;

}

\*intPtr++ = '%';

\*intPtr = '\n';

}

void\* calc\_score(void\* arg) {

// проблема в том, что использование rand() не позволяет достичь требуемой производительности,

// поэтому используем свой генератор псевдослучайных чисел

unsigned long current = time(NULL);

const unsigned long a = 1664525;

const unsigned long c = 1013904223;

const unsigned long m = 4294967296;

int rounds = ((int\*)arg)[0];

int exp = ((int\*)arg)[1];

int first\_score = ((int\*)arg)[2];

int second\_score = ((int\*)arg)[3];

int\* score = malloc(sizeof(int) \* 2);

long double\* result = malloc(sizeof(long double) \* 3);

for (int j = 0; j < exp; j ++) {

score[0] = (float)first\_score;

score[1] = (float)second\_score;

for (int i = 0; i < rounds; i++) {

current = (current \* a + c) % m;

score[0] += current % 6 + 1;

current = (current \* a + c) % m;

score[0] += current % 6 + 1;

current = (current \* a + c) % m;

score[1] += current % 6 + 1;

current = (current \* a + c) % m;

score[1] += current % 6 + 1;

}

if(score[0] > score[1]) {

result[0] += 1.;

result[2] += 1.;

} else if(score[0] < score[1]) {

result[1] += 1.;

result[2] += 1.;

}

}

free(arg);

free(score);

return result;

}

// ввод (6 цифорок):

// количество бросков двух костей, какой сейчас тур, сколько очков суммарно у каждого из игроков,

// количество экспериментов, которые должна произвести программа, количество потоков

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

if (argc != 7) {

char msg[] = "USAGE: ./a.out <total rounds> <current tour> <first\_score> <second\_score> <experiments> <threads>\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 1;

}

if (validate\_nums(argc, argv)) {

char msg[] = "ERROR: all input numbers must be integer and positive\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 2;

}

int k = atoi(argv[1]);

int tour = atoi(argv[2]);

int first\_score = atoi(argv[3]);

int second\_score = atoi(argv[4]);

int experiments = atoi(argv[5]);

int num\_threads = atoi(argv[6]);

if (k < tour) {

char msg[] = "ERROR: total rounds must be greater than current tour\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 3;

}

if (num\_threads > MAX\_THREADS || num\_threads < 1) {

char msg[] = "ERROR: number of threads must be lesser than 12 and greater than 1\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 4;

}

// если игра уже окончена, то у одного шанс на победу 100%, а у второго 0%

if (k == tour) {

if (first\_score > second\_score) {

char msg1[] = "First player win with 100%% probability\n";

char msg2[] = "Second player win with 0%% probability\n";

write(STDOUT\_FILENO, msg1, sizeof(msg1) - 1);

write(STDOUT\_FILENO, msg2, sizeof(msg2) - 1);

} else if (first\_score < second\_score) {

char msg1[] = "First player win with 0%% probability\n";

char msg2[] = "Second player win with 100%% probability\n";

write(STDOUT\_FILENO, msg1, sizeof(msg1) - 1);

write(STDOUT\_FILENO, msg2, sizeof(msg2) - 1);

} else {

char msg1[] = "First player win with 100%% probability\n";

char msg2[] = "Second player win with 100%% probability\n";

write(STDOUT\_FILENO, msg1, sizeof(msg1) - 1);

write(STDOUT\_FILENO, msg2, sizeof(msg2) - 1);

}

return 0;

}

pthread\_t experiments\_threads[num\_threads];

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

int\* data\_for\_calc = malloc(sizeof(int) \* 4);

data\_for\_calc[0] = k - tour;

data\_for\_calc[1] = experiments / num\_threads + 1;

data\_for\_calc[2] = first\_score;

data\_for\_calc[3] = second\_score;

if (pthread\_create(&experiments\_threads[i], NULL, calc\_score, data\_for\_calc)) {

char msg[] = "ERROR: thread cannot be created\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 5;

}

}

long double first\_prob = 0.;

long double second\_prob = 0.;

long double exp = 0.;

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

long double\* scores;

if (pthread\_join(experiments\_threads[i], (void\*\*)&scores)) {

char msg[] = "ERROR: thread cannot be joined\n";

write(STDERR\_FILENO, msg, sizeof(msg) - 1);

return 6;

}

first\_prob += scores[0];

second\_prob += scores[1];

exp += scores[2];

}

first\_prob = first\_prob / exp \* 100;

second\_prob = second\_prob / exp \* 100;

char num[16];

float\_to\_string(first\_prob, num, 16, 2);

char msg1[] = "Probability of the first player winnig - ";

write(STDOUT\_FILENO, msg1, sizeof(msg1) - 1);

write(STDOUT\_FILENO, num, sizeof(num) - 1);

float\_to\_string(second\_prob, num, 16, 2);

char msg2[] = "Probability of the second player winnig - ";

write(STDOUT\_FILENO, msg2, sizeof(msg2) - 1);

write(STDOUT\_FILENO, num, sizeof(num) - 1);

return 0;

}

**Протокол работы программы**

**Некорректный ввод:**

dmitrij@Katana:~/Документы/MAI/os/MAI\_OS/lab02/src$ ./a.out 10 0 1 1 100000000 13

ERROR: number of threads must be lesser than 12 and greater than 1

**10 раундов, 1 тур, 1 очко у первого, 1 у второго, 10000000 экспериментов, 12 потоков:**

dmitrij@Katana:~/Документы/MAI/os/MAI\_OS/lab02/src$ ./a.out 10 0 1 1 100000000 12

Probability of the first player winnig - 49.98%

Probability of the second player winnig - 50.01%

**Strace:**

strace ./a.out 10 0 1 1 100000000 12

**execve("./a.out", ["./a.out", "10", "0", "1", "1", "100000000", "12"], 0x7ffcb768a4f0 /\* 81 vars \*/) = 0**

**brk(NULL) = 0x58f9f4dad000**

**mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7b1cd4b5c000**

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

**openat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/glibc-hwcaps/x86-64-v3/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)**

**newfstatat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/glibc-hwcaps/x86-64-v3/", 0x7ffebc0cf830, 0) = -1 ENOENT (Нет такого файла или каталога)**

**openat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/glibc-hwcaps/x86-64-v2/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)**

**newfstatat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/glibc-hwcaps/x86-64-v2/", 0x7ffebc0cf830, 0) = -1 ENOENT (Нет такого файла или каталога)**

**openat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)**

**newfstatat(AT\_FDCWD, "/usr/local/cuda-12.6/lib64/", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}, 0) = 0**

**openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3**

**fstat(3, {st\_mode=S\_IFREG|0644, st\_size=78439, ...}) = 0**

**mmap(NULL, 78439, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7b1cd4b48000**

**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\220\243\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**

**fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 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, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7b1cd4800000**

**mmap(0x7b1cd4828000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7b1cd4828000**

**mmap(0x7b1cd49b0000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7b1cd49b0000**

**mmap(0x7b1cd49ff000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7b1cd49ff000**

**mmap(0x7b1cd4a05000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7b1cd4a05000**

**close(3) = 0**

**mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7b1cd4b45000**

**arch\_prctl(ARCH\_SET\_FS, 0x7b1cd4b45740) = 0**

**set\_tid\_address(0x7b1cd4b45a10) = 131763**

**set\_robust\_list(0x7b1cd4b45a20, 24) = 0**

**rseq(0x7b1cd4b46060, 0x20, 0, 0x53053053) = 0**

**mprotect(0x7b1cd49ff000, 16384, PROT\_READ) = 0**

**mprotect(0x58f9f3c05000, 4096, PROT\_READ) = 0**

**mprotect(0x7b1cd4b94000, 8192, PROT\_READ) = 0**

**prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0**

**munmap(0x7b1cd4b48000, 78439) = 0**

**getrandom("\x74\x49\xd7\xb0\xf7\x43\x59\xdc", 8, GRND\_NONBLOCK) = 8**

**brk(NULL) = 0x58f9f4dad000**

**brk(0x58f9f4dce000) = 0x58f9f4dce000**

**rt\_sigaction(SIGRT\_1, {sa\_handler=0x7b1cd4899520, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7b1cd4845320}, 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) = 0x7b1cd3e00000**

**mprotect(0x7b1cd3e01000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd4600990, parent\_tid=0x7b1cd4600990, exit\_signal=0, stack=0x7b1cd3e00000, stack\_size=0x7fff80, tls=0x7b1cd46006c0} => {parent\_tid=[131764]}, 88) = 131764**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1ccb600000**

**mprotect(0x7b1ccb601000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1ccbe00990, parent\_tid=0x7b1ccbe00990, exit\_signal=0, stack=0x7b1ccb600000, stack\_size=0x7fff80, tls=0x7b1ccbe006c0} => {parent\_tid=[131765]}, 88) = 131765**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd3400000**

**mprotect(0x7b1cd3401000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd3c00990, parent\_tid=0x7b1cd3c00990, exit\_signal=0, stack=0x7b1cd3400000, stack\_size=0x7fff80, tls=0x7b1cd3c006c0} => {parent\_tid=[131766]}, 88) = 131766**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd2a00000**

**mprotect(0x7b1cd2a01000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd3200990, parent\_tid=0x7b1cd3200990, exit\_signal=0, stack=0x7b1cd2a00000, stack\_size=0x7fff80, tls=0x7b1cd32006c0} => {parent\_tid=[131767]}, 88) = 131767**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd2000000**

**mprotect(0x7b1cd2001000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd2800990, parent\_tid=0x7b1cd2800990, exit\_signal=0, stack=0x7b1cd2000000, stack\_size=0x7fff80, tls=0x7b1cd28006c0} => {parent\_tid=[131768]}, 88) = 131768**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd1600000**

**mprotect(0x7b1cd1601000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd1e00990, parent\_tid=0x7b1cd1e00990, exit\_signal=0, stack=0x7b1cd1600000, stack\_size=0x7fff80, tls=0x7b1cd1e006c0} => {parent\_tid=[131769]}, 88) = 131769**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd0c00000**

**mprotect(0x7b1cd0c01000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd1400990, parent\_tid=0x7b1cd1400990, exit\_signal=0, stack=0x7b1cd0c00000, stack\_size=0x7fff80, tls=0x7b1cd14006c0} => {parent\_tid=[131770]}, 88) = 131770**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cd0200000**

**mprotect(0x7b1cd0201000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cd0a00990, parent\_tid=0x7b1cd0a00990, exit\_signal=0, stack=0x7b1cd0200000, stack\_size=0x7fff80, tls=0x7b1cd0a006c0} => {parent\_tid=[131771]}, 88) = 131771**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1ccac00000**

**mprotect(0x7b1ccac01000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1ccb400990, parent\_tid=0x7b1ccb400990, exit\_signal=0, stack=0x7b1ccac00000, stack\_size=0x7fff80, tls=0x7b1ccb4006c0} => {parent\_tid=[131772]}, 88) = 131772**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cca200000**

**mprotect(0x7b1cca201000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1ccaa00990, parent\_tid=0x7b1ccaa00990, exit\_signal=0, stack=0x7b1cca200000, stack\_size=0x7fff80, tls=0x7b1ccaa006c0} => {parent\_tid=[131773]}, 88) = 131773**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cc9800000**

**mprotect(0x7b1cc9801000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cca000990, parent\_tid=0x7b1cca000990, exit\_signal=0, stack=0x7b1cc9800000, stack\_size=0x7fff80, tls=0x7b1cca0006c0} => {parent\_tid=[131774]}, 88) = 131774**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7b1cc8e00000**

**mprotect(0x7b1cc8e01000, 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\_CHILD\_CLEARTID, child\_tid=0x7b1cc9600990, parent\_tid=0x7b1cc9600990, exit\_signal=0, stack=0x7b1cc8e00000, stack\_size=0x7fff80, tls=0x7b1cc96006c0} => {parent\_tid=[131775]}, 88) = 131775**

**rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0**

**futex(0x7b1cd4600990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 131764, NULL, FUTEX\_BITSET\_MATCH\_ANY) = 0**

**futex(0x7b1ccbe00990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 131765, NULL, FUTEX\_BITSET\_MATCH\_ANY) = 0**

**munmap(0x7b1cd3e00000, 8392704) = 0**

**munmap(0x7b1ccb600000, 8392704) = 0**

**munmap(0x7b1cd3400000, 8392704) = 0**

**munmap(0x7b1cd2a00000, 8392704) = 0**

**munmap(0x7b1cd2000000, 8392704) = 0**

**munmap(0x7b1cd1600000, 8392704) = 0**

**munmap(0x7b1cd0c00000, 8392704) = 0**

**munmap(0x7b1cd0200000, 8392704) = 0**

**write(1, "Probability of the first player "..., 41Probability of the first player winnig - ) = 41**

**write(1, "49.99%\n\0\0\0\0\0\0\0\0", 1549.99%**

**) = 15**

**write(1, "Probability of the second player"..., 42Probability of the second player winnig - ) = 42**

**write(1, "50.00%\n\0\0\0\0\0\0\0\0", 1550.00%**

**) = 15**

**exit\_group(0) = ?**

**+++ exited with 0 +++**

**Вывод**

Язык Си с поддержкой библиотек позволяет создавать многопоточные приложения, предоставляя инструменты для работы с потоками и механизмы ограничения для обеспечения безопасности. Это делает разработку на Си более разнообразной и увлекательной.