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", 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", 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.