Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу
«Операционные системы»

Тема работы
“Потоки”

Студент: Тутаев Владимир Владимирович
Группа: М8О-201Б-23
Вариант: 15

Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Репозиторий

https://github.com/Volan4ik/MAI_OS.git

Постановка задачи

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.

Вариант 15: Есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность того, что сверху лежат две одинаковых карты. Количество раундов задаётся ключом программы

Общие сведения о программе

Этот код представляет собой многопоточную реализацию симуляции Монте-Карло. Программа использует несколько потоков для выполнения симуляций параллельно, что ускоряет процесс. Основная идея заключается в том, что каждый поток выполняет определенное количество раундов симуляции, где в каждом раунде колода из 52 карт заполняется значениями от 0 до 12 (представляющими карты), перемешивается с помощью алгоритма Фишера-Йетса, а затем проверяется, совпадают ли две верхние карты. После завершения всех потоков программа суммирует результаты и вычисляет вероятность совпадения двух верхних карт. Входные данные программы — количество потоков и общее количество раундов симуляции, которые делятся между потоками.

Исходный код в Приложении 1

Strace в Приложении 2

Зависимости ускорения и эффективности алгоритма

Число потоков	Время исполнения(мс)	Ускорение	Эффективность
1	4575	1	1
2	2491	1,83	0,915
3	1860	2,46	0,820
4	1613	2,84	0,710
5	1668	2,74	0,548
6	1750	2,61	0,435

Выводы

В процессе выполнения данной работы я изучил многопоточность в C, научился использовать функции `pthread_create` и `pthread_join` для создания и управления потоками. Также я познакомился с методом Монте-Карло и его применением для оценки вероятности событий.

Приложение 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>

#define DECK_SIZE 52

typedef struct {
    int rounds;
    int thread_id;
    int matches;
} ThreadData;

void shuffle_deck(int *deck) { // Fisher-Yates algorithm
    for (int i = DECK_SIZE - 1; i > 0; i--) {
        int j = rand() % (i + 1);
        int temp = deck[i];
        deck[i] = deck[j];
        deck[j] = temp;
    }
}

void* monte_carlo_simulation(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    int deck[DECK_SIZE];

    for (int round = 0; round < data->rounds; ++round) {
        for (int i = 0; i < DECK_SIZE; ++i) {
            deck[i] = i % 13;
        }

        shuffle_deck(deck);

        if (deck[0] == deck[1]) {
            data->matches++;
        }
    }

    return NULL;
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        printf("Usage: %s <number_of_threads> <number_of_rounds>\n", argv[0]);
        return 1;
    }

    int num_threads = atoi(argv[1]);
    int total_rounds = atoi(argv[2]);
```

```
    if (num_threads <= 0 || total_rounds <= 0) {
        printf("Number of threads and rounds must be positive.\n");
        return 1;
    }

    pthread_t threads[num_threads];
    ThreadData thread_data[num_threads];

    int rounds_per_thread = total_rounds / num_threads;
    srand(time(NULL));

    for (int i = 0; i < num_threads; ++i) {
        thread_data[i].rounds = rounds_per_thread;
        thread_data[i].thread_id = i;
        thread_data[i].matches = 0;

        if (pthread_create(&threads[i], NULL, monte_carlo_simulation, &thread_data[i]) != 0) {
            perror("Failed to create thread");
            return 1;
        }
    }

    int total_matches = 0;
    for (int i = 0; i < num_threads; ++i) {
        pthread_join(threads[i], NULL);
        total_matches += thread_data[i].matches;
    }

    double probability = (double)total_matches / total_rounds;

    printf("Probability of matching top two cards: %.6f)\n", probability);

    return 0;
}
```

Приложение 2

```
execve("./monte_carlo_simulation", ["./monte_carlo_simulation", "4", "100000"], 0x7ffd7e61d8c0 /* 75 vars */) = 0
brk(NULL)                               = 0x577ac3cb3000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x799e07a23000
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=87915, ...}) = 0
mmap(NULL, 87915, PROT_READ, MAP_PRIVATE, 3, 0) = 0x799e07a0d000
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) = 0x799e07600000
mmap(0x799e07628000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x799e07628000
mmap(0x799e077b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x799e077b0000
mmap(0x799e077ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x799e077ff000
mmap(0x799e07805000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x799e07805000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x799e07a0a000
arch_prctl(ARCH_SET_FS, 0x799e07a0a740) = 0
set_tid_address(0x799e07a0aa10)         = 919710
set_robust_list(0x799e07a0aa20, 24)     = 0
rseq(0x799e07a0b060, 0x20, 0, 0x53053053) = 0
mprotect(0x799e077ff000, 16384, PROT_READ) = 0
mprotect(0x577ac20bb000, 4096, PROT_READ) = 0
mprotect(0x799e07a5b000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x799e07a0d000, 87915)           = 0
rt_sigaction(SIGRT_1, {sa_handler=0x799e07699520, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x799e07645320}, 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) =
0x799e06c00000
mprotect(0x799e06c01000, 8388608, PROT_READ|PROT_WRITE) = 0
getrandom("\xad\xaf\x10\x0d\xdf\x14\xd2\x4e", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x577ac3cb3000
brk(0x577ac3cd4000)                     = 0x577ac3cd4000
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSV
SEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x799e07400990,
parent_tid=0x799e07400990, exit_signal=0, stack=0x799e06c00000, stack_size=0x7fff80, tls=0x799e074006c0}
=> {parent_tid=[919711]}, 88) = 919711
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x799e06200000
mprotect(0x799e06201000, 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_SYSV
SEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x799e06a00990,
parent_tid=0x799e06a00990, exit_signal=0, stack=0x799e06200000, stack_size=0x7fff80, tls=0x799e06a006c0}
=> {parent_tid=[919712]}, 88) = 919712
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x799e05800000
mprotect(0x799e05801000, 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_SYSV
SEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x799e06000990,
parent_tid=0x799e06000990, exit_signal=0, stack=0x799e05800000, stack_size=0x7fff80, tls=0x799e060006c0}
=> {parent_tid=[919713]}, 88) = 919713
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x799e04e00000
mprotect(0x799e04e01000, 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_SYSV
SEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x799e05600990,
parent_tid=0x799e05600990, exit_signal=0, stack=0x799e04e00000, stack_size=0x7fff80, tls=0x799e056006c0}
=> {parent_tid=[919714]}, 88) = 919714
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
futex(0x799e07400990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 919711, NULL,
FUTEX_BITSET_MATCH_ANY) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Probability of matching top two "..., 49) = 49
exit_group(0)                           = ?
+++ exited with 0 +++
```