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-209БВ-24

Студент: Корепанов И. А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 19.11.25

Москва, 2025

Постановка задачи

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- pthread_create() - создание потоков

- pthread_join() - ожидание завершения потоков

- pthread_mutex_lock()/unlock() - синхронизация доступа

- clock_gettime() - измерение времени выполнения

- pthread_mutex_init()/destroy() - управление мьютексом

Алгоритм работы программы:

Сначала программа инициализирует количество потоков и раундов, распределяет нагрузку между потоками для более эффективного выполнения программы. Затем каждый из потоков выполняет программу. После получения результатов потоки поочерёдно используют мьтекс, после чего выводится результат и время за которое программа выполнена.

Код программы

lab2.c

```
#define _POSIX_C_SOURCE 200112L
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>
#include <getopt.h>

typedef unsigned long long ull;

pthread_mutex_t mutex;
ull global_success = 0;
```

```
void* worker(void* arg) {
    ull rounds = *(ull*)arg;
    unsigned int seed = (unsigned int)time(NULL) ^ (unsigned int)pthread_self();
    ull local_success = 0;

    for (ull i = 0; i < rounds; i++) {
        int a = rand_r(&seed) % 52;
        int b = rand_r(&seed) % 51;
        if (b >= a) b++;

        if ((a % 13) == (b % 13))
            local_success++;
    }

    pthread_mutex_lock(&mutex);
    global_success += local_success;
    pthread_mutex_unlock(&mutex);

    return NULL;
}

int main(int argc, char* argv[]) {
    int threads = -1;
    ull rounds = 0;
    int opt;
    while ((opt = getopt(argc, argv, "t:n:")) != -1) {
        switch (opt) {
            case 't': threads = atoi(optarg); break;
            case 'n': rounds = strtoull(optarg, NULL, 10); break;
        }
    }
```

```
if (threads <= 0 || rounds == 0) {
    fprintf(stderr, "Использование: %s -t <потоки> -n <раунды>\n", argv[0]);
    return 1;
}

pthread_t* ths = malloc(sizeof(pthread_t) * threads);
ull* args = malloc(sizeof(ull) * threads);
ull base = rounds / threads;
ull rem = rounds % threads;

pthread_mutex_init(&mutex, NULL);

struct timespec t0, t1;
clock_gettime(CLOCK_MONOTONIC, &t0);

for (int i = 0; i < threads; i++) {
    args[i] = base + (i < (int)rem ? 1 : 0);
    pthread_create(&ths[i], NULL, worker, &args[i]);
}

for (int i = 0; i < threads; i++)
    pthread_join(ths[i], NULL);

clock_gettime(CLOCK_MONOTONIC, &t1);
double elapsed = (t1.tv_sec - t0.tv_sec) + (t1.tv_nsec - t0.tv_nsec) / 1e9;

double probability = (double)global_success / (double)rounds;

printf("Потоков: %d\n", threads);
printf("Раундов: %llu\n", rounds);
printf("Вероятность совпадения: %.6f\n", probability);
printf("Время выполнения: %.6f сек\n", elapsed);
```

```
    pthread_mutex_destroy(&mutex);
    free(ths);
    free(args);
    return 0;
}
```

Протокол работы программы

Тестирование 1:

vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 1 -n 1000

Потоков: 1

Раундов: 1000

Вероятность совпадения: 0.043000

Время выполнения: 0.000376 сек

vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 2 -n 1000

Потоков: 2

Раундов: 1000

Вероятность совпадения: 0.057000

Время выполнения: 0.000366 сек

vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 4 -n 1000

Потоков: 4

Раундов: 1000

Вероятность совпадения: 0.056000

Время выполнения: 0.000609 сек

vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 8 -n 1000

Потоков: 8

Раундов: 1000

Вероятность совпадения: 0.068000

Время выполнения: 0.001173 сек

Число потоков	**Время выполнения (с)**	**Ускорение**	**Эффективность**
1	0.000376	1	1
2	0.000366	1,03	0,515
4	0.000609	0,62	0,155
8	0.001173	0,32	0,04

Тестирование 2:

```
vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 1 -n 999999999
```

```
Потоков: 1
Раундов: 999999999
Вероятность совпадения: 0.058791
Время выполнения: 18.655831 сек
vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 2 -n 999999999
Потоков: 2
Раундов: 999999999
Вероятность совпадения: 0.058791
Время выполнения: 9.880005 сек
vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 4 -n 999999999
Потоков: 4
Раундов: 999999999
Вероятность совпадения: 0.058794
Время выполнения: 5.174774 сек
vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ ./lab2 -t 8 -n 999999999
Потоков: 8
Раундов: 999999999
Вероятность совпадения: 0.058791
Время выполнения: 2.799594 сек
```

Число потоков	Время выполнения (с)	Ускорение	Эффективность
1	18.655831	1	1
2	9.880005	1,89	0,945
4	5.174774	3,61	0,9025
8	2.799594	6,66	0,8325

Strace:

vscode ➜ /workspaces/OS_LABS/lab2/src (main) $ strace -f ./lab2 -t 8 -n 999999999

execve("./lab2", ["./lab2", "-t", "8", "-n", "999999999"], 0x7ffdb1efb978 /* 29 vars */) = 0

brk(NULL) = 0x5f6f4cb6b000

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=21784, ...}) = 0

mmap(NULL, 21784, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75ec3b882000

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 l\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=149520, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x75ec3b880000

mmap(NULL, 136304, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x75ec3b85e000

mmap(0x75ec3b864000, 65536, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x75ec3b864000

mmap(0x75ec3b874000, 24576, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x75ec3b874000

mmap(0x75ec3b87a000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x75ec3b87a000

mmap(0x75ec3b87c000, 13424, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75ec3b87c000

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\260>\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1901536, ...}) = 0

mmap(NULL, 1914496, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x75ec3b68a000

mmap(0x75ec3b6ac000, 1413120, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x75ec3b6ac000

mmap(0x75ec3b805000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17b000) = 0x75ec3b805000

mmap(0x75ec3b854000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c9000) = 0x75ec3b854000

mmap(0x75ec3b85a000, 13952, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75ec3b85a000

close(3)                        = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x75ec3b687000

arch_prctl(ARCH_SET_FS, 0x75ec3b687740) = 0

mprotect(0x75ec3b854000, 16384, PROT_READ) = 0

mprotect(0x75ec3b87a000, 4096, PROT_READ) = 0

mprotect(0x5f6f13e46000, 4096, PROT_READ) = 0

mprotect(0x75ec3b8b2000, 4096, PROT_READ) = 0

munmap(0x75ec3b882000, 21784)           = 0
```

set_tid_address(0x75ec3b687a10) = 12693

set_robust_list(0x75ec3b687a20, 24) = 0

rt_sigaction(SIGRTMIN, {sa_handler=0x75ec3b864690, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x75ec3b871140}, NULL, 8) = 0

rt_sigaction(SIGRT_1, {sa_handler=0x75ec3b864730, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x75ec3b871140}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

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

brk(NULL) = 0x5f6f4cb6b000

brk(0x5f6f4cb8c000) = 0x5f6f4cb8c000

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75ec3ae86000

mprotect(0x75ec3ae87000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x75ec3b685fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12694 attached

, parent_tid=[12694], tls=0x75ec3b686700, child_tidptr=0x75ec3b6869d0) = 12694

[pid 12694] set_robust_list(0x75ec3b6869e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12694] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec3a685000

[pid 12693] mprotect(0x75ec3a686000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec3ae84fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12695 attached

, parent_tid=[12695], tls=0x75ec3ae85700, child_tidptr=0x75ec3ae859d0) = 12695

[pid 12695] set_robust_list(0x75ec3ae859e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12695] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec39e84000

[pid 12693] mprotect(0x75ec39e85000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec3a683fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE

_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12696 attached

, parent_tid=[12696], tls=0x75ec3a684700, child_tidptr=0x75ec3a6849d0) = 12696

[pid 12696] set_robust_list(0x75ec3a6849e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12696] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec39683000

[pid 12693] mprotect(0x75ec39684000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec39e82fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12697 attached

, parent_tid=[12697], tls=0x75ec39e83700, child_tidptr=0x75ec39e839d0) = 12697

[pid 12697] set_robust_list(0x75ec39e839e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12697] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec38e82000

[pid 12693] mprotect(0x75ec38e83000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec39681fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12698 attached

, parent_tid=[12698], tls=0x75ec39682700, child_tidptr=0x75ec396829d0) = 12698

[pid 12698] set_robust_list(0x75ec396829e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12698] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec38681000

[pid 12693] mprotect(0x75ec38682000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec38e80fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12699 attached

, parent_tid=[12699], tls=0x75ec38e81700, child_tidptr=0x75ec38e819d0) = 12699

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12699] set_robust_list(0x75ec38e819e0, 24 <unfinished ...>

[pid 12693] <... mmap resumed>) = 0x75ec37e80000

[pid 12699] <... set_robust_list resumed>) = 0

[pid 12693] mprotect(0x75ec37e81000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec3867ffb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12700 attached

, parent_tid=[12700], tls=0x75ec38680700, child_tidptr=0x75ec386809d0) = 12700

[pid 12700] set_robust_list(0x75ec386809e0, 24 <unfinished ...>

[pid 12693] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 12700] <... set_robust_list resumed>) = 0

[pid 12693] <... mmap resumed>) = 0x75ec3767f000

[pid 12693] mprotect(0x75ec37680000, 8388608, PROT_READ|PROT_WRITE) = 0

[pid 12693] clone(child_stack=0x75ec37e7efb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDstrace: Process 12701 attached

, parent_tid=[12701], tls=0x75ec37e7f700, child_tidptr=0x75ec37e7f9d0) = 12701

[pid 12701] set_robust_list(0x75ec37e7f9e0, 24 <unfinished ...>

[pid 12693] futex(0x75ec3b6869d0, FUTEX_WAIT, 12694, NULL <unfinished ...>

[pid 12701] <... set_robust_list resumed>) = 0

[pid 12698] madvise(0x75ec38e82000, 8368128, MADV_DONTNEED) = 0

[pid 12698] exit(0) = ?

[pid 12698] +++ exited with 0 +++

[pid 12700] madvise(0x75ec37e80000, 8368128, MADV_DONTNEED) = 0

[pid 12700] exit(0) = ?

[pid 12700] +++ exited with 0 +++

[pid 12694] madvise(0x75ec3ae86000, 8368128, MADV_DONTNEED) = 0

[pid 12694] exit(0) = ?

[pid 12693] <... futex resumed>) = 0

[pid 12694] +++ exited with 0 +++

[pid 12693] futex(0x75ec3ae859d0, FUTEX_WAIT, 12695, NULL <unfinished ...>

[pid 12695] madvise(0x75ec3a685000, 8368128, MADV_DONTNEED) = 0

```
[pid 12695] exit(0)                     = ?
[pid 12693] <... futex resumed>)        = 0
[pid 12695] +++ exited with 0 +++
[pid 12693] futex(0x75ec3a6849d0, FUTEX_WAIT, 12696, NULL <unfinished ...>
[pid 12697] madvise(0x75ec39683000, 8368128, MADV_DONTNEED) = 0
[pid 12697] exit(0)                     = ?
[pid 12697] +++ exited with 0 +++
[pid 12696] madvise(0x75ec39e84000, 8368128, MADV_DONTNEED) = 0
[pid 12696] exit(0)                     = ?
[pid 12693] <... futex resumed>)        = 0
[pid 12696] +++ exited with 0 +++
[pid 12693] munmap(0x75ec3ae86000, 8392704) = 0
[pid 12693] futex(0x75ec38e819d0, FUTEX_WAIT, 12699, NULL <unfinished ...>
[pid 12699] madvise(0x75ec38681000, 8368128, MADV_DONTNEED) = 0
[pid 12699] exit(0)                     = ?
[pid 12693] <... futex resumed>)        = 0
[pid 12699] +++ exited with 0 +++
[pid 12693] munmap(0x75ec3a685000, 8392704) = 0
[pid 12693] munmap(0x75ec39e84000, 8392704) = 0
[pid 12693] futex(0x75ec37e7f9d0, FUTEX_WAIT, 12701, NULL <unfinished ...>
[pid 12701] madvise(0x75ec3767f000, 8368128, MADV_DONTNEED) = 0
[pid 12701] exit(0)                     = ?
[pid 12693] <... futex resumed>)        = 0
[pid 12701] +++ exited with 0 +++
munmap(0x75ec39683000, 8392704)         = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0
write(1, "\320\237\320\276\321\202\320\276\320\272\320\276\320\262: 8\n", 18Потоков: 8
) = 18
write(1, "\320\240\320\260\321\203\320\275\320\264\320\276\320\262: 999999999\n",
26Раундов: 999999999
) = 26
write(1,
"\320\222\320\265\321\200\320\276\321\217\321\202\320\275\320\276\321\201\321\202\321\214
\321\201\320\276\320\262\320\277\320"..., 54Вероятность совпадения: 0.058809
```

) = 54

write(1, "\320\222\321\200\320\265\320\274\321\217 \320\262\321\213\320\277\320\276\320\273\320\275\320\265\320\275\320\270\321\217:"..., 49Время выполнения: 2.852213 сек

) = 49

exit_group(0) = ?

+++ exited with 0 +++

Вывод

В ходе лабораторной работы была реализована многопоточная программа для вычисления вероятности того, что сверху лежат две одинаковых карты.

При выполнении работы для синхронизации потоков я использовал мьютексы. После написания программы я провёл тесты для анализа эффективности многопоточности.

Было проведено два тестирования. Одно с маленьким количеством данных, а другое с большим. Первое тестирование показало, что при маленьких данных большое количество потоков замедляет программу. Это связано с тем, что на создание потоков требуется время и ресурсы компьютера. Однако второе тестирование показало, что при большом количестве данных многопоточность ускоряет программу. Таким образом можно сказать, что многопоточность эффективна при большом объеме данных.