|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №3 по курсу Операционные системы ­  студента группы М8О-207Б-18 Хренова Геннадия Николаевича , № по списку 28  Адреса www, e-mail, jabber, skype [khrenov.gena@yandex.ru](mailto:khrenov.gena@yandex.ru)  Работа выполнена: “30“ октября 2019 г.  Преподаватель: Миронов Е. С.  Входной контроль знаний с оценкой  Отчёт сдан “ “ 20 г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Потоки в ОС
2. **Цель работы**: Приобретение практических навыков в: управлении потоками в ОС, обеспечении синхронизации между потоками
3. **Задание** (*вариант №* 20): Есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность того, что сверху лежат две одинаковых карты. Количество раундов подается с ключом.
4. **Оборудование** (*лабораторное*):

ЭВМ Pentium , процессор 2.7 ГГц , имя узла сети с ОП 4096 МБ

НМД 2 ГБ. Терминал адрес . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор intel core i3 , ОП 3072 , НМД 50 ГБ. Монитор

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Unix , наименование версия

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Ubuntu версия 18.04LTS

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Для выполнения работы необходимы следующие команды:

pthread\_create — создание потока

pthread\_exit — завершение потока

pthread\_join — ожидание выполнения потока

pthread\_mutex\_init — создание мьютекса

pthread\_mutex\_destroy — удаление мьютекса

pthread\_mutex\_lock — закрытие мьютекса потоком

pthread\_mutex\_unlock — открытие мьютекса потоком

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

Метод Монте-Карло предполагает экспериментальный подсчет вероятности. Этот метод не отличается большой точностью, однако чем больше сыграно, тем вероятность более приближается к теоретической вероятности, равной 1/17(шанс того, что первая карта может быть из пары = 1, шанс того, что вторая карта окажется из такой пары — 3/51 = 1/17). Программа получает при вызове 1 аргумент — количество раундов. Создаётся число потоков, равное числу раундов, если оно не превышает максимальное число потоков, в противном случае создаётся максимальное число потоков, происходит их выполнение и удаление, и после этого создается ещё нужное количество потоков. В функции потока рандомным образом выбираются 2 неодинаковых числа от 0 до 51, далее их целочисленное деление на 4 определяет из старшинство(например если div 4 = 1 это тройки, 10 — вальты). Если div совпадает и это пара, то процесс закрывает mutex, прибавляет 1 к глобальному счётчику и открывает mutex, после чего завершается. Также в функции потока есть ещё переменная, увеличивающаяся на 1. Она нужна для улучшения рандома. Затем мы ждём, пока все процессы завершаться, и печатаем ответ.

Тесты:

1) 10

2) 100

3) 1000

4) 100000

(точность должна слегка увеличиваться)

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <sys/types.h>

#include <unistd.h>

#include <pthread.h>

const int MAX\_THREAD = 6800;

double count = 0;

int r = 0;

pthread\_mutex\_t mutex;

void\* threadFunc() {

srand(pthread\_self() + r);

int a = rand() % 52;

int b = rand() % 52;

while(a == b) {

b = rand() % 52;

}

if (a / 4 == b / 4) {

if(pthread\_mutex\_lock(&mutex) != 0){

fprintf(stderr, "error lock mutex\n");

}

count++;

if(pthread\_mutex\_unlock(&mutex) !=0){

fprintf(stderr, "error unlock mutex\n");

}

}

r++;

pthread\_exit(NULL);

}

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

if (argc != 2) {

fprintf(stderr, "invalid input\n");

exit(2);

}

if(pthread\_mutex\_init(&mutex, NULL) != 0){

fprintf(stderr, "error mutexin\n");

}

pthread\_t\* threads = (pthread\_t\*) malloc(MAX\_THREAD \* sizeof(pthread\_t));

double x = 0;

double n = 0;

while (n < atoi(argv[1])){

for(int i = 0; i < MAX\_THREAD; i++) {

if (pthread\_create(&(threads[i]), NULL, threadFunc, NULL) != 0){

fprintf(stderr, "error pthread creating\n");

}

x++;

if(n + x == atoi(argv[1])){

break;

}

}

n += x;

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

if(pthread\_join(threads[i], NULL) != 0){

fprintf(stderr, "error join\n");

}

}

x = 0;

}

printf("%f\n", count/atoi(argv[1]));

if(pthread\_mutex\_destroy(&mutex) != 0){

fprintf(stderr, "error mutex destroy\n");

}

free(threads);

return 0;

}

результат работы strace:

execve("./a.out", ["./a.out", "5"], 0x7ffcf44a03d8 /\* 57 vars \*/) = 0

brk(NULL) = 0x5589db177000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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=92646, ...}) = 0

mmap(NULL, 92646, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f59cb9a6000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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\0000b\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f59cb9a4000

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f59cb577000

mprotect(0x7f59cb591000, 2093056, PROT\_NONE) = 0

mmap(0x7f59cb790000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f59cb790000

mmap(0x7f59cb792000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f59cb792000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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\34\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f59cb186000

mprotect(0x7f59cb36d000, 2097152, PROT\_NONE) = 0

mmap(0x7f59cb56d000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f59cb56d000

mmap(0x7f59cb573000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f59cb573000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f59cb9a1000

arch\_prctl(ARCH\_SET\_FS, 0x7f59cb9a1740) = 0

mprotect(0x7f59cb56d000, 16384, PROT\_READ) = 0

mprotect(0x7f59cb790000, 4096, PROT\_READ) = 0

mprotect(0x5589da1ee000, 4096, PROT\_READ) = 0

mprotect(0x7f59cb9bd000, 4096, PROT\_READ) = 0

munmap(0x7f59cb9a6000, 92646) = 0

set\_tid\_address(0x7f59cb9a1a10) = 2354

set\_robust\_list(0x7f59cb9a1a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f59cb57ccb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f59cb589890}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f59cb57cd50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f59cb589890}, 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) = 0x5589db177000

brk(0x5589db198000) = 0x5589db198000

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f59ca985000

mprotect(0x7f59ca986000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f59cb184fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f59cb1859d0, tls=0x7f59cb185700, child\_tidptr=0x7f59cb1859d0) = 2355

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f59ca184000

mprotect(0x7f59ca185000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f59ca983fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f59ca9849d0, tls=0x7f59ca984700, child\_tidptr=0x7f59ca9849d0) = 2356

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f59c976b000

mprotect(0x7f59c976c000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f59c9f6afb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f59c9f6b9d0, tls=0x7f59c9f6b700, child\_tidptr=0x7f59c9f6b9d0) = 2357

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f59c8f6a000

mprotect(0x7f59c8f6b000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f59c9769fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f59c976a9d0, tls=0x7f59c976a700, child\_tidptr=0x7f59c976a9d0) = 2358

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f59c8769000

mprotect(0x7f59c876a000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f59c8f68fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f59c8f699d0, tls=0x7f59c8f69700, child\_tidptr=0x7f59c8f699d0) = 2359

munmap(0x7f59ca985000, 8392704) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

write(1, "0.200000\n", 90.200000

) = 9

exit\_group(0) = ?

+++ exited with 0 +++

Демонстрация работы:

gennadii@lenovo-b560:~/workdir/os/laba3$ ./a.out 100

0.070000

gennadii@lenovo-b560:~/workdir/os/laba3$ ./a.out 1000

0.071000

gennadii@lenovo-b560:~/workdir/os/laba3$ ./a.out 10000

0.059500

gennadii@lenovo-b560:~/workdir/os/laba3$ ./a.out 100000

0.058390

**9.Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. Замечание автора по существу работы
2. Выводы:Потоки имеют общее адресное пространство, в отличие от процессов. Также потоки более легкие(быстрые) в плане их создания и удаления по сравнению с более тяжелыми процессами. Во многих системах создание потоков осуществляется в 10–100 раз быстрее, чем создание процессов. Потоки нужны, когда одновременно происходит несколько действий, и некоторые из них блокируются. Тогда работа с несколькими потоками, которые параллельно выполняют действия, ускоряет программу. Чтобы огранить доступ нежелательным потокам к данным, используются мьютекс, семафор или условные переменные.

