Московский авиационный институт (национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование» Дисциплина «Операционные системы»

Лабораторная работа №3

Тема: Управление потоками в ОС

Студент: Будникова В.П.

Группа: М8О-207Б-19

Преподаватель: Миронов Е. С.

Дата:

Оценка:

Москва, 2020

Цель работы: приобретение практических навыков в управлении потоками в ОС и обеспечение синхронизации между потоками.

Задача: Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Вариант(22): Рассчитать детерминант матрицы

Листинг программы

main.c

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

typedef struct HelpThead {
   int left, right;
   int str;
} HelpThead;

long double * a; //указатель на матрицу
int n; //длина строки/слотбца

void* ThreadFunc(void* args) {
   HelpThead help = *((HelpThead*)args);
   long double temp;
   for (int i = help.left * n; i < help.right * n; ++i) {
      if (i % n == 0 && i + help.str < n * n){
         if (a[n * help.str + help.str] != 0) {
            temp = a[i + help.str] / a[n * help.str + help.str];
         } else {
            temp = 0;
         }

      }
      a[i] -= a[i % n + help.str * n] * temp;
   }
   return 0;
}

int Min(int x, int y) {
   return x < y ? x : y;
}
```

```
signed main(int argc, char * argv[]) {
    int threadCount = 0;
    if (argc == 2){
        for(int i = 0; i < strlen(argv[1]); ++i) {
            threadCount = threadCount * 10 + (int)(argv[1][i] -'0');
        }
    }
    scanf("%d", &n);                    //количество строк(столбцов) матрицы
    a = (long double*)malloc(sizeof(long double) * n * n);
    long double * st = (long double*)malloc(sizeof(long double) * n);
    for (int i = 0; i < n * n; ++i) {
        scanf("%Lf", &a[i]);
    }

    int temp = threadCount;
    long double time_begin, time_end;
    time_begin = clock();
    pthread_t* threads = (pthread_t*)malloc(sizeof(pthread_t) * threadCount);
    HelpThead* HelpTheads = (HelpThead*)malloc(sizeof(HelpThead*) * threadCount);
    int k = 0;
    long double Ans = 1;
    while (k < n) {
        int step = (n + threadCount - 1) / threadCount;
        for (int i = 0; i < threadCount; ++i) {
            HelpTheads[i].left = i * step + 1 + k;
            HelpTheads[i].right = Min(n, (i + 1) * step) + 1 + k;
            HelpTheads[i].str = k;
            int ind = 1;
            while (a[n * k + k] == 0 && ind < n - k) {
                for (int i = 0; i < n; ++i){
                    st[i] = a[i * n + k];
                    a[i * n + k] = a[i * n + k + ind];
                    a[i * n + k + ind] = st[i];
                }
                ++ind;
            }
            ind = 1;
            if (Min(n, (i + 1) * step) + 1 + k > n) { HelpTheads[i].right = n; }
            if (pthread_create(&threads[i], NULL, ThreadFunc, (void*)&HelpTheads[i])) {
                printf("Error creating thread!\n");
                return -1;
            }
        }
        for (int i = 0; i < threadCount; ++i) {
            if (pthread_join(threads[i], NULL)) {
                printf("Error joining thread!\n");
                return -1;
            }
        }
        Ans *= a[n * k + k];
        ++k;
```

```
    }
    time_end = clock();

    printf("Count:  %d\n", threadCount);
    printf("time: %Lf ms\n", time_end - time_begin);
    printf("\nAnswer:: %Lf\n", Ans);
    free(threads);
    free(HelpTheads);
}
```

makefile

```
all:
	gcc -std=c11 lab3.c -o lab3 -pthread
run:
	./lab3 2 < 1.txt
```

Тесты и протокол исполнения

```
Lera:lab3 valeriabudnikova$ cat 1.txt
5
 70  77  67  83  35
 85  25  76  5  73
 50  97  29  39  5
 88  86  77  9  16
 12  8  65  59  4

Lera:lab3 valeriabudnikova$ ./lab3 1 < 1.txt
Count:  1
time: 246.000000 ms
Answer:: -437777488.000000
Lera:lab3 valeriabudnikova$ ./lab3 2 < 1.txt
Count:  2
time: 547.000000 ms
Answer:: -437777488.000000
Lera:lab3 valeriabudnikova$ cat 1.txt
10
 36  96  32  68  82  28  60  4  38  30
 36  79  57  46  90  6  37  75  4  17
 65  29  69  36  14  53  57  91  76  62
 35  5  90  62  9  38  7  27  47  9
 95  18  9  9  21  21  28  83  38  29
 89  2  1  57  26  90  53  94  76  23
 95  96  23  84  89  76  59  72  76  90
 81  88  32  16  2  53  81  99  16  58
 49  57  68  53  77  18  24  90  60  90
 4  43  67  13  40  53  94  89  74  17

Lera:lab3 valeriabudnikova$ ./lab3 1 < 1.txt
Count:  1
time: 771.000000 ms

Answer:: 267515724862276026.953125
Lera:lab3 valeriabudnikova$ ./lab3 2 < 1.txt
Count:  2
time: 1060.000000 ms
```

Вывод strace

strace ./lab3 4 < 1.txt

execve("./lab3", ["./lab3", "4"], 0x7ffc7cbf57c8 /* 67 vars */) = 0

brk(NULL) = 0x55f2348b2000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe7d12b3b0) = -1 EINVAL (Недопустимый аргумент)

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

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=118993, ...}) = 0

mmap(NULL, 118993, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf85741000

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\220\201\0\0\0\0\0\0"..., 832) = 832

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68, 824) = 68

fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbf8573f000

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68, 824) = 68

mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fbf8571c000

mmap(0x7fbf85723000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7fbf85723000

mmap(0x7fbf85734000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7fbf85734000

mmap(0x7fbf85739000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7fbf85739000

mmap(0x7fbf8573b000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbf8573b000

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\360q\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

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68

fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 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

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?
\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68

mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fbf8552a000

mprotect(0x7fbf8554f000, 1847296, PROT_NONE) = 0

mmap(0x7fbf8554f000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x25000) = 0x7fbf8554f000

mmap(0x7fbf856c7000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x19d000) = 0x7fbf856c7000

mmap(0x7fbf85712000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1e7000) = 0x7fbf85712000

mmap(0x7fbf85718000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7fbf85718000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fbf85527000

arch_prctl(ARCH_SET_FS, 0x7fbf85527740) = 0

mprotect(0x7fbf85712000, 12288, PROT_READ) = 0

mprotect(0x7fbf85739000, 4096, PROT_READ) = 0

mprotect(0x55f23327e000, 4096, PROT_READ) = 0

mprotect(0x7fbf8578c000, 4096, PROT_READ) = 0

munmap(0x7fbf85741000, 118993) = 0

set_tid_address(0x7fbf85527a10) = 52910

set_robust_list(0x7fbf85527a20, 24) = 0

rt_sigaction(SIGRTMIN, {sa_handler=0x7fbf85723bf0, sa_mask=[], sa_flags=SA_RESTORER|
SA_SIGINFO, sa_restorer=0x7fbf857313c0}, NULL, 8) = 0

rt_sigaction(SIGRT_1, {sa_handler=0x7fbf85723c90, sa_mask=[], sa_flags=SA_RESTORER|
SA_RESTART|SA_SIGINFO, sa_restorer=0x7fbf857313c0}, 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

fstat(0, {st_mode=S_IFREG|0664, st_size=105, ...}) = 0

brk(NULL) = 0x55f2348b2000

brk(0x55f2348d3000) = 0x55f2348d3000

read(0, "5\n31 \n 4 86 57 71 21 \n 65 1"..., 4096) = 105

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=1502741}) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1,
0) = 0x7fbf84d26000

mprotect(0x7fbf84d27000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7fbf85525fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52911],
tls=0x7fbf85526700, child_tidptr=0x7fbf855269d0) = 52911

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1,
0) = 0x7fbf84525000

mprotect(0x7fbf84526000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7fbf84d24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52912],
tls=0x7fbf84d25700, child_tidptr=0x7fbf84d259d0) = 52912

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1,
0) = 0x7fbf83d24000

mprotect(0x7fbf83d25000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7fbf84523fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52913],
tls=0x7fbf84524700, child_tidptr=0x7fbf845249d0) = 52913

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1,
0) = 0x7fbf83523000

mprotect(0x7fbf83524000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7fbf83d22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52914],
tls=0x7fbf83d23700, child_tidptr=0x7fbf83d239d0) = 52914

clone(child_stack=0x7fbf83d22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52915],
tls=0x7fbf83d23700, child_tidptr=0x7fbf83d239d0) = 52915

clone(child_stack=0x7fbf84523fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52916],
tls=0x7fbf84524700, child_tidptr=0x7fbf845249d0) = 52916

clone(child_stack=0x7fbf84d24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52917],
tls=0x7fbf84d25700, child_tidptr=0x7fbf84d259d0) = 52917

clone(child_stack=0x7fbf85525fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52918],
tls=0x7fbf85526700, child_tidptr=0x7fbf855269d0) = 52918

futex(0x7fbf84d259d0, FUTEX_WAIT, 52917, NULL) = 0

clone(child_stack=0x7fbf85525fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|

CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52919], tls=0x7fbf85526700, child_tidptr=0x7fbf855269d0) = 52919

clone(child_stack=0x7fbf84d24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52920], tls=0x7fbf84d25700, child_tidptr=0x7fbf84d259d0) = 52920

clone(child_stack=0x7fbf84523fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52921], tls=0x7fbf84524700, child_tidptr=0x7fbf845249d0) = 52921

clone(child_stack=0x7fbf83d22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52922], tls=0x7fbf83d23700, child_tidptr=0x7fbf83d239d0) = 52922

clone(child_stack=0x7fbf83d22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52923], tls=0x7fbf83d23700, child_tidptr=0x7fbf83d239d0) = 52923

clone(child_stack=0x7fbf84523fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52924], tls=0x7fbf84524700, child_tidptr=0x7fbf845249d0) = 52924

clone(child_stack=0x7fbf84d24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52925], tls=0x7fbf84d25700, child_tidptr=0x7fbf84d259d0) = 52925

clone(child_stack=0x7fbf85525fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52926], tls=0x7fbf85526700, child_tidptr=0x7fbf855269d0) = 52926

clone(child_stack=0x7fbf85525fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52927], tls=0x7fbf85526700, child_tidptr=0x7fbf855269d0) = 52927

clone(child_stack=0x7fbf84d24fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52928], tls=0x7fbf84d25700, child_tidptr=0x7fbf84d259d0) = 52928

clone(child_stack=0x7fbf84523fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52929], tls=0x7fbf84524700, child_tidptr=0x7fbf845249d0) = 52929

clone(child_stack=0x7fbf83d22fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|

```
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[52930],
tls=0x7fbf83d23700, child_tidptr=0x7fbf83d239d0) = 52930
futex(0x7fbf845249d0, FUTEX_WAIT, 52929, NULL) = -1 EAGAIN (Ресурс временно
недоступен)
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=2542089}) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Count:  4\n", 10Count:  4
)            = 10
write(1, "time: 1040.000000 ms\n", 21time: 1040.000000 ms
)  = 21
write(1, "\n", 1
)                    = 1
write(1, "Answer:: -1142674134.000000\n", 28Answer:: -1142674134.000000
) = 28
lseek(0, -2, SEEK_CUR)                = 103
exit_group(0)                         = ?
+++ exited with 0 +++
```

Strace:

clone - системный вызов для создания нового процесса. Обеспечивает разделяемые адресные пространства, родительский процесс(откуда происходит вызов) не может выполняться в том же стеке, что и дочерний. родительский процесс передает указатель на пространство памяти для дочернего.

write - системный вызов, для записи в файл. Первый аргумент - файловый дескриптор. Второй аргумент - адрес, с которого начинается буфер. Третий аргумент - количество байтов, которое нужно записать в файл, на который ссылается файловый дескриптор из буфера. При удачном завершении возвращается количество байтов, которые были записаны(в случае ошибки - "-1").

clock_gettime - системный возов, который получает и устанавливает время.

Выводы

Так как при написании программы я использовала для нахождения определителя матрицы Алгоритм Гаусса, при распараллеливании программы, которое в моем случае будет происходить только в тот момент, когда мы вычитаем строчку из строчки(чтобы обнулить элемент). На маленьких матрицах время при увеличении количества потоков увеличивается гораздо быстрее, чем при больших. Алгоритм Гаусса нельзя

распараллелить по-другому, кроме как таким способом, что не показывает ускорении программы, при большем количестве потоков, так как происходит распараллеливание, при котором параллельные процессы выполняют очень маленькое количество действий, а время на создание потоков будет сильно влиять на общее время работы программы. В программе также присутствуют глобальные переменные, чтобы каждый поток мог обращаться к этим переменным. При данной реализации данный использование примитивов синхронизации использовать не потребовалось, так как происходит один раз чтение, а дальше каждый поток меняет данные матрицы строго не "пересекающиеся" с данными матрицы, которые меняет другой поток, для этого используется специальная структура, которая хранит индексы элементов, для каждого потока. В данной лабораторной работе я реализовала программу на си и отладила ее.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб.: Издательский дом «Питер», 2018. — С. 111 - 123
2. Поисковик Google [электронный ресурс] URL: https://google.com/ (дата обращения: 22.09.2020)