ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)"

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Лабороторная работа № 3 по курсу

по курсу «Операционные системы»

Группа: М8о-207Б-18

Студент:

Токарев Никита Станиславович

Преподаватель:

Миронов Евгений Сергеевич

Оценка:

Дата:

Москва,2019

Оглавление

1.Постановка задачи

Вариант № 14:

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). При создании необходимо предусмотреть ключи, которые позволяли бы задать максимальное количество потоков, используемое программой. При возможности необходимо использовать максимальное количество возможных потоков. Ограничение потоков может быть задано или ключом запуска вашей программы, или алгоритмом. Рассчитать определитель матрицы.

2.Структура программы

В данной работе в моя программа состоит из двух файлов:

1. main.c

2. Makefile

Также в моей программе используются такие системные вызовы: pthread_create, pthread_join.

3.Описание программы

Создание потока происходит с помощью функции **pthread_create(pthread_t *tid, const pthread_attr_t *attr, void*(*function)(void*), void* arg)**. Функция **pthread_join(pthread_t *tid, const pthread_attr_t *attr)** ожидает завершения потока thread. Второй параметр этой функции — результат, возвращаемый потоком.

Моя программа высчитывает определитель матрицы многопоточно, методом Гаусса(приведение к треугольному виду). Для начала программа запускается с ключом то есть ./a.out 15, где 15 количество используемых потоков, также стало известно, что, максимум потоков на моем компьютере не должен превышать 8949. После запуска матрица стандартно считывается с потока данных, а потом запускается цикл с условием **while(size_of_amount > 1)**, где size_of_amount - количество операций, для привидения матрицы к треугольному виду. После запуска цикла, встречается функция swap(threadData). Данная функция проверяет,является ли наш i-тый столбец 0, если да, то цикл прекращается и детерминант = 0. Если проверка прошла успешно, то дальше встречаются два условия.

1. Если количество возможных используемых потоков, больше количества операций на данном этапе.
2. Если количество возможных используемых потоков, меньше количества операций на данном этапе.

Дальше в зависимости от ситуации запускается системный вызов pthread_create(&threads[amount_thr], NULL, Operation, &new_data[amount_thr]); - где Передается структура new_data которая содержит индекс строки, которую надо обнулить и индекс строки, относительно которой обнуляется и также ссылку на массив. В функции Operation, строка обнуляется. Похожий принцип работы во втором условии,но, есть одно отличие: там запускается возможное количество потоков,затем они завершаются и снова запускаются пока количество операций в данном условии не зайдет за пределы переменной size_of_amount. Также в функции Operation, если будет обрабатываться структура с передаваемой строкой превыщающей размеры массива, то функция Operation вернет NULL. Затем элементы на главной диагонали перемножаются. Полученное произведение является детерминантом матрицы.

4.Листинг программы

//main.cpp

```
#include <pthread.h>
#include <stdlib.h>
#include <stdbool.h>
#include <mcheck.h>
#include <stdio.h>
#define max_thr 8949


typedef struct{
        double** array;
        int number_str;
        int position;
   int size_m;
} pthrData;


bool swap(pthrData *data) {
   if(data->array[data->number_str][data->number_str] == 0) {
      int i = data->number_str;
      while(i < data->size_m) {
         if(data->array[i][data->number_str] != 0) {
            double v;
            for(int index = data->number_str; index < data->size_m; index++) {
               v = data->array[i][index];
               data->array[i][index] = data->array[data->number_str][index];
               data->array[data->number_str][index] = v;
            }
            return true;
         }
      i++;
```

```
        }
    } else {
        return true;
    }
    return false;
}


pthrData get_new_data(pthrData *data,int j) {
    pthrData new_data;
    new_data.array = data->array;
            new_data.number_str = data->number_str;
            new_data.position = j;
    new_data.size_m = data->size_m;
            return new_data;
}


void* Operation(void* thread_data){
            pthrData *data = (pthrData*)thread_data;
    if(data->position >= data->size_m) {
        return NULL;
    }
    double x = -1 * data->array[data->position][data->number_str] / data->array[data->number_str]
[data->number_str];
    for(int index = data->number_str;index < data->size_m; index++) {
        data->array[data->position][index] = data->array[data->position][index] + x * data-
>array[data->number_str][index];
    }
    return NULL;
}


int main(int argc, char **argv){
```

```
	if (argc != 2) {
		printf("Error\n");
		return 0;
	}
	int n = atoi(argv[1]);
	int size;
	scanf("%d", &size);
	if(n > max_thr) {
		printf("max of threads = 8949\n");
		n = max_thr;
}
	double **matrix = (double **)malloc(sizeof(double *) * size);
	for(int i = 0; i < size; i++) {
		matrix[i] = (double *)malloc(sizeof(double) * size);
	}
	for(int i = 0; i < size; i++){
		for(int j = 0; j < size; j++){
			double value;
			scanf("%lf", &value);
			matrix[i][j] = value;
		}
	}
pthread_t* threads = (pthread_t*) malloc(sizeof(pthread_t) * n);
	pthrData* threadData = (pthrData*)malloc(sizeof(pthrData));
	threadData->array = matrix;
	threadData->number_str = 0;
	threadData->position = 0;
threadData->size_m = size;
int size_of_amount = size;
bool ind = true;
int amount_thr = 0;
while(size_of_amount > 1) {
   if(swap(threadData) == false) {
	ind = false;
```

```
    printf("Determinant = 0\n");
    break;
}
if(n >= size_of_amount - 1) {
    pthrData* new_data = (pthrData*)malloc(sizeof(pthrData) * size_of_amount - 1);
    for(int index = threadData->position + 1; index < size; index++) {
        new_data[amount_thr] = get_new_data(threadData,index);
        pthread_create(&threads[amount_thr], NULL, Operation, &new_data[amount_thr]);
        amount_thr++;
    }
    for(int index = 0; index < amount_thr; index++) {
        pthread_join(threads[index], NULL);
    }
    amount_thr = 0;
    free(new_data);
} else if(n < size_of_amount - 1) {
    bool indicator = true;
    int count_n = threadData->position + 1;
    int limit = count_n + n;
    pthrData* new_data = (pthrData*)malloc(sizeof(pthrData) * n);
    while(indicator == true) {
        if(limit >= size) {
            indicator = false;
        }
        for(int index = count_n; index < limit; index++) {
            new_data[amount_thr] = get_new_data(threadData,index);
            pthread_create(&threads[amount_thr], NULL, Operation, &new_data[amount_thr]);
            amount_thr++;
        }
        for(int index = 0; index < amount_thr; index++) {
            pthread_join(threads[index], NULL);
        }
        amount_thr = 0;
        count_n = limit;
```

```
            limit = limit + n;
        }
        free(new_data);
    }
    threadData->position++;
    threadData->number_str++;
    size_of_amount--;
  }
  if(ind == true) {
    double det = 1;
    for(int index = 0; index < threadData->size_m; index++) {
        if(threadData->array[index][index] == 0) {
            det = 0;
            break;
        }
        det = det * threadData->array[index][index];
    }
    printf("Determinant = %.1lf\n", det);
  }
        free(threads);
        for(int i = 0; i < size; i++){
                free(matrix[i]);
        }
        free(matrix);
        free(threadData);
        return 0;
}
//Makefile
CC = gcc

CFLAGS = -g -std=c99 -Wextra  -pedantic -Werror

OBJ = main.o
```

```
main: main.o
	$(CC) $(CCFLAGS) -o det main.o -lpthread

main.o: main.c

clean:
	@rm -r *.o det
```

5.Результат работы

```
nikita@nikita-HP:~/oc/lr3v14$ cat t1.txt
5
2 0 5 4 7
1 2 4 5 0
4 5 22 3 33
22 11 13 45 45
23 12 34 56 78

nikita@nikita-HP:~/oc/lr3v14$ cat t2.txt
4
 3 4 9 4
 2 5 9 2
 2 2 5 1
 2 2 5 1

nikita@nikita-HP:~/oc/lr3v14$ cat t3.txt
3
  4 9 4
  5 9 2
  2 5 1

nikita@nikita-HP:~/oc/lr3v14$ make
gcc -g -std=c99 -Wextra  -pedantic -Werror    -c -o main.o main.c
gcc  -o det main.o -lpthread
```

```
nikita@nikita-HP:~/oc/lr3v14$ ./det 15 < t1.txt
Determinant = -91323.0
nikita@nikita-HP:~/oc/lr3v14$ ./det 2 < t1.txt
Determinant = -91323.0
nikita@nikita-HP:~/oc/lr3v14$ ./det 2 < t2.txt
Determinant = 0.0
nikita@nikita-HP:~/oc/lr3v14$ ./det 2 < t3.txt
Determinant = 15.0
```

6.Вывод

Одно из отличий потоков от процессов: потоки делят между собой одно адресное пространство. Все потоки в программе разделяют некоторую память, они имеют доступ ко всем переменным в области их видимости. Это очень удобно, но также опасно: с момента параллельного запуска потока, переменные или функции могут использоваться одновременно разными потоками. Если операция не является безопасной, это может привести к неопределенному поведению (т. е. это может привести к сбою или повреждению данных). В данной работе детерминант матрицы считается многопоточно: матрица приводится к ступенчатому виду в мнопоточном режиме, что является одним из важных действий для нахождения детерминанта матрицы.

7.Вызов Strace

Данный вызов актуален для файла t2.txt. Также можно заметить , что создается два потока, затем исполняются.

```
nikita@nikita-HP:~/oc/lr3v14$ ./det 2 < t2.txt
Determinant = 0.0
nikita@nikita-HP:~/oc/lr3v14$ strace ./det 2 < t2.txt
execve("./det", ["./det", "2"], 0x7ffdd5048898 /* 55 vars */) = 0
brk(NULL)                               = 0x5618490fc000
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=96869, ...}) = 0
```

```
mmap(NULL, 96869, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd50ae17000
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) = 0x7fd50ae15000
mmap(NULL, 2221184, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_DENYWRITE, 3, 0) = 0x7fd50a9e9000
mprotect(0x7fd50aa03000, 2093056, PROT_NONE) = 0
mmap(0x7fd50ac02000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x19000) = 0x7fd50ac02000
mmap(0x7fd50ac04000, 13440, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd50ac04000
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"..., 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) = 0x7fd50a5f8000
mprotect(0x7fd50a7df000, 2097152, PROT_NONE) = 0
mmap(0x7fd50a9df000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fd50a9df000
mmap(0x7fd50a9e5000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd50a9e5000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7fd50ae12000
arch_prctl(ARCH_SET_FS, 0x7fd50ae12740) = 0
mprotect(0x7fd50a9df000, 16384, PROT_READ) = 0
mprotect(0x7fd50ac02000, 4096, PROT_READ) = 0
mprotect(0x561848c83000, 4096, PROT_READ) = 0
mprotect(0x7fd50ae2f000, 4096, PROT_READ) = 0
munmap(0x7fd50ae17000, 96869)           = 0
set_tid_address(0x7fd50ae12a10)         = 6078
set_robust_list(0x7fd50ae12a20, 24)     = 0
```

```
rt_sigaction(SIGRTMIN, {sa_handler=0x7fd50a9eecb0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fd50a9fb890}, NULL, 8)
= 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fd50a9eed50, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO,
sa_restorer=0x7fd50a9fb890}, 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=39, ...}) = 0
brk(NULL)                               = 0x5618490fc000
brk(0x56184911d000)                     = 0x56184911d000
read(0, "4\n 3 4 9 4\n 2 5 9 2\n 2 2 5 1\n 2 "..., 4096) = 39
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|
MAP_STACK, -1, 0) = 0x7fd509df7000
mprotect(0x7fd509df8000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fd50a5f6fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd50a5f79d0, tls=0x7fd50a5f7700, child_tidptr=0x7fd50a5f79d0)
= 6079
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|
MAP_STACK, -1, 0) = 0x7fd5095f6000
mprotect(0x7fd5095f7000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fd509df5fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd509df69d0, tls=0x7fd509df6700, child_tidptr=0x7fd509df69d0)
= 6080
clone(child_stack=0x7fd509df5fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd509df69d0, tls=0x7fd509df6700, child_tidptr=0x7fd509df69d0)
= 6081
clone(child_stack=0x7fd50a5f6fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd50a5f79d0, tls=0x7fd50a5f7700, child_tidptr=0x7fd50a5f79d0)
= 6082
clone(child_stack=0x7fd50a5f6fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
```

```
parent_tidptr=0x7fd50a5f79d0, tls=0x7fd50a5f7700, child_tidptr=0x7fd50a5f79d0)
= 6083
clone(child_stack=0x7fd509df5fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd509df69d0, tls=0x7fd509df6700, child_tidptr=0x7fd509df69d0)
= 6084
clone(child_stack=0x7fd509df5fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd509df69d0, tls=0x7fd509df6700, child_tidptr=0x7fd509df69d0)
= 6085
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
write(1, "Determinant = 0.0\n", 18Determinant = 0.0
)     = 18
lseek(0, -2, SEEK_CUR)                  = 37
exit_group(0)                           = ?
+++ exited with 0 +++
```