Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**УПРАВЛЕНИЕ ПОТОКАМИ В ОС**

Студент: Черненко Илья Денисович

Группа: М8О–206Б–18

Вариант: 3

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Примеры работы
6. Вывод

**Постановка задачи**

Составить программу, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы. 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.

Отсортировать массив строк при помощи параллельной сортировки слиянием.

**Общие сведения о программе**

Программа компилируется из файла main.cpp. Также используется заголовочные файлы: iostream, string, pthread.h, math.h. В программе используются следующие системные вызовы:

1. **pthread\_create** – создает новый поток
2. **pthread\_join** – ожидает завершения переданного потока, получает его выходное значение
3. **pthread\_mutex\_init** – инициализация mutex.
4. **pthread\_mutex\_destroy –** уничтожение mutex.
5. **pthread\_mutex\_lock –** блокировка части кода определенным потоком.
6. **pthread\_mutex\_unlock –** разблокировка части кода определенным потоком.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Ввести размер массива строк
2. Аргументом исполняемого файла задать кол-во потоков(которое приведется к степени двойки(кроме 1).
3. Используя системный вызов fork создать дочерний процесс.
4. Исходный массив сортируется параллельной сортировкой слиянием.
5. Отсортированный массив выводится на стандартный поток вывода.

**Основные файлы программы**

**main.cpp:**

#include <iostream>

#include <pthread.h>

#include <string>

#include <cmath>

using namespace std;

int Length;

int NumberOfThreads;

pthread\_mutex\_t mutex;

struct arguments {

string\* orig;

int left;

int right;

string\* mod;

};

int char\_to\_int(char c) {

if (c >= '0' && c <= '9') {

return (c-'0');

} else if (c >= 'a' && c <= 'z') {

c -= 'W';

} else if (c >= 'A' && c <= 'Z') {

c = tolower(c);

c -= 'W';

} else {

return 0;

}

}

bool a\_lower\_or\_eq\_b(string a, string b) {

int v1 = 0;

int v2 = 0;

int i = a.size();

int k = 1;

while (i > 0) {

i--;

v1 += char\_to\_int(a[i])\*k;

k\*=10;

}

i = b.size();

k = 1;

while (i > 0) {

i--;

v2 += char\_to\_int(b[i])\*k;

k\*=10;

}

if (v1 <= v2) {

return true;

}

return false;

}

void merge(string \*orig, int left, int mid, int right, string \*modif) {

int l = left;

int r = mid;

for (int i = left; i < right; i++)

if (l < mid && (r >= right || a\_lower\_or\_eq\_b(orig[l], orig[r])))

modif[i] = orig[l++];

else

modif[i] = orig[r++];

for (int i = left; i < right; i++)

orig[i] = modif[i];

}

void\* split(void\* param) {

struct arguments\* temp\_args = new arguments;

temp\_args = (arguments\*)param;

if (temp\_args->right - temp\_args->left < 2) return NULL;

int tmp\_right = temp\_args->right;

int tmp\_left = temp\_args->left;

temp\_args->right = (tmp\_left + tmp\_right) / 2;

split((void\*)temp\_args);

temp\_args->right = tmp\_right;

temp\_args->left = (tmp\_left + tmp\_right) / 2;

split((void\*)temp\_args);

temp\_args->left = tmp\_left;

merge(temp\_args->orig, temp\_args->left, (temp\_args->left + temp\_args->right) / 2, temp\_args->right, temp\_args->mod);

}

void merge\_sort(string \*in) {

struct arguments\* a = new arguments;

string temp[Length];

pthread\_t threads[NumberOfThreads];

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

int new\_left = i \* Length / NumberOfThreads;

int new\_right = (i + 1) \* Length / NumberOfThreads;

a->mod = temp;

a->orig = in;

a->left = new\_left;

a->right = new\_right;

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&threads[i], NULL, split, (void\*)a);

pthread\_join(threads[i], NULL);

pthread\_mutex\_destroy(&mutex);

}

for (int i = NumberOfThreads/2; i > 0; i = i >> 1) //divide by 2

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

int left = (j)\*Length / i;

int right = (j + 1)\*Length / i;

merge(in, left, (left + right) / 2, right, temp);

}

}

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

cin >> Length;

string array[Length];

for (int i = 0; i < Length; i++)

cin >> array[i];

NumberOfThreads = atoi(argv[1]);

int power = 0;

while (NumberOfThreads > 0) {

NumberOfThreads = NumberOfThreads >> 1;

power++;

}

power--;

if (power == 0) {

power = 1;

}

NumberOfThreads = (int)pow(2.0, (double)(power));

merge\_sort(array);

cout << "Sorted array:\n";

for (int i = 0; i < Length; i++)

cout << array[i] << "\n";

return 0;

}

**Примеры работы**

**test1:**

zebr@zebr-VirtualBox:~/os\_lab\_3/src/cmake-build-debug$ strace -c ./os\_lab3 10000

10

0

0

999

aaa

bbb

aaa

fff

7

4

5

Sorted array:

0

0

4

5

7

999

aaa

aaa

bbb

fff

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

87.19 0.904331 55 16384 clone

12.71 0.131806 36 3673 3282 futex

0.06 0.000582 36 16 read

0.03 0.000341 31 11 write

0.01 0.000064 4 18 mmap

0.00 0.000028 4 8 fstat

0.00 0.000024 24 1 1 lseek

0.00 0.000011 1 13 mprotect

0.00 0.000000 0 6 close

0.00 0.000000 0 1 munmap

0.00 0.000000 0 3 brk

0.00 0.000000 0 2 rt\_sigaction

0.00 0.000000 0 1 rt\_sigprocmask

0.00 0.000000 0 7 7 access

0.00 0.000000 0 1 execve

0.00 0.000000 0 1 arch\_prctl

0.00 0.000000 0 1 set\_tid\_address

0.00 0.000000 0 6 openat

0.00 0.000000 0 1 set\_robust\_list

0.00 0.000000 0 1 prlimit64

------ ----------- ----------- --------- --------- ----------------

100.00 1.037187 20155 3290 total

**test2:**

zebr@zebr-VirtualBox:~/os\_lab\_3/src/cmake-build-debug$ strace -c ./os\_lab3 4

4

и

b

a

5

Sorted array:

5

a

b

и

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

24.94 0.000817 102 8 clone

13.06 0.000428 43 10 read

12.30 0.000403 31 13 mprotect

8.39 0.000275 55 5 write

7.30 0.000239 80 3 futex

6.50 0.000213 12 18 mmap

4.55 0.000149 50 3 brk

4.18 0.000137 17 8 fstat

3.51 0.000115 58 2 rt\_sigaction

2.20 0.000072 72 1 prlimit64

1.98 0.000065 65 1 munmap

1.98 0.000065 9 7 7 access

1.83 0.000060 60 1 set\_tid\_address

1.77 0.000058 58 1 1 lseek

1.74 0.000057 57 1 rt\_sigprocmask

1.74 0.000057 57 1 set\_robust\_list

1.01 0.000033 6 6 openat

0.92 0.000030 5 6 close

0.09 0.000003 3 1 arch\_prctl

0.00 0.000000 0 1 execve

------ ----------- ----------- --------- --------- ----------------

100.00 0.003276 97 8 total

**test3:**

zebr@zebr-VirtualBox:~/os\_lab\_3/src/cmake-build-debug$ strace -c ./os\_lab3 10000

10

0

0

999

aaa

bbb

aaa

fff

7

4

5

Sorted array:

0

0

4

5

7

999

aaa

aaa

bbb

fff

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

85.89 0.870777 53 16384 clone

14.07 0.142641 39 3615 3079 futex

0.03 0.000314 29 11 write

0.00 0.000024 3 8 fstat

0.00 0.000019 19 1 1 lseek

0.00 0.000000 0 16 read

0.00 0.000000 0 6 close

0.00 0.000000 0 18 mmap

0.00 0.000000 0 13 mprotect

0.00 0.000000 0 1 munmap

0.00 0.000000 0 3 brk

0.00 0.000000 0 2 rt\_sigaction

0.00 0.000000 0 1 rt\_sigprocmask

0.00 0.000000 0 7 7 access

0.00 0.000000 0 1 execve

0.00 0.000000 0 1 arch\_prctl

0.00 0.000000 0 1 set\_tid\_address

0.00 0.000000 0 6 openat

0.00 0.000000 0 1 set\_robust\_list

0.00 0.000000 0 1 prlimit64

------ ----------- ----------- --------- --------- ----------------

100.00 1.013775 20097 3087 total

**test4:**

zebr@zebr-VirtualBox:~/os\_lab\_3-master/src/cmake-build-debug$ strace -c ./os\_lab\_3 1

4

9

5

4

87

Sorted array:

4

5

9

87

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

17.98 0.000747 57 13 mprotect

13.14 0.000546 182 3 futex

11.79 0.000490 27 18 mmap

11.34 0.000471 47 10 read

8.21 0.000341 43 8 fstat

5.58 0.000232 232 1 clone

5.17 0.000215 72 3 brk

5.01 0.000208 104 2 rt\_sigaction

4.74 0.000197 39 5 write

2.48 0.000103 103 1 prlimit64

2.45 0.000102 102 1 rt\_sigprocmask

2.43 0.000101 101 1 munmap

2.29 0.000095 95 1 arch\_prctl

2.26 0.000094 94 1 set\_tid\_address

2.21 0.000092 92 1 set\_robust\_list

1.78 0.000074 12 6 close

1.13 0.000047 47 1 1 lseek

0.00 0.000000 0 7 7 access

0.00 0.000000 0 1 execve

0.00 0.000000 0 6 openat

------ ----------- ----------- --------- --------- ----------------

100.00 0.004155 90 8 total

**test5:**

zebr@zebr-VirtualBox:~/os\_lab\_3-master/src/cmake-build-debug$ strace -c ./os\_lab\_3 2

11

bfa

aaa

0

1

2

3

4

bbb

aaa

kkk

llll

Sorted array:

0

1

2

3

4

aaa

aaa

bbb

bfa

kkk

llll

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

60.77 0.001078 63 17 read

13.19 0.000234 59 4 futex

11.89 0.000211 18 12 write

7.50 0.000133 67 2 clone

2.82 0.000050 6 8 fstat

2.03 0.000036 3 13 mprotect

1.41 0.000025 1 18 mmap

0.39 0.000007 7 1 1 lseek

0.00 0.000000 0 6 close

0.00 0.000000 0 1 munmap

0.00 0.000000 0 3 brk

0.00 0.000000 0 2 rt\_sigaction

0.00 0.000000 0 1 rt\_sigprocmask

0.00 0.000000 0 7 7 access

0.00 0.000000 0 1 execve

0.00 0.000000 0 1 arch\_prctl

0.00 0.000000 0 1 set\_tid\_address

0.00 0.000000 0 6 openat

0.00 0.000000 0 1 set\_robust\_list

0.00 0.000000 0 1 prlimit64

------ ----------- ----------- --------- --------- ----------------

100.00 0.001774 106 8 total

**Вывод**

Обрел навыки работы с многопоточностью, использовал mutex для синхронизации потоков. Многопоточность является одним из важнейших механизмов в программировании, позволяющий синхронизировать данные программы и налаживать её корректную работу.