**Отчет**

**по лабораторной работе №3.**

**3 семестр**

|  |  |
| --- | --- |
| **Студент:** | **Фирфаров А.С.** |
| **Группа:** | **8O-208Б** |
| **Преподаватель** | **Миронов Е.С.** |

## Цель работы

Целью является приобретение практических навыков в:

* Управлении потоками в ОС
* Обеспечении синхронизации между потоками

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке 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.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

## Вариант задания

19. Есть набор 128 битных чисел записанных в шестнадцатеричном представлении, хранящихся в файле. Необходимо посчитать их среднее арифметическое. Округлить результат до целых. Количество используемой оперативной памяти должно задаваться "ключом".

**Описание алгоритма**

Программа принимает на вход 2 ключа: количество используемой оперативной памяти и максимальное количество потоков. Данные проверяются на корректность и сохраняются в переменные. Программа запрашивает ввести имя существующего файла с 128 битными числами, записанными в шестнадцатеричном представлении. Каждое число расположено на отдельной строчке. Файл заканчивается пустой строкой. Далее программа вычисляет количество чисел в файле и инициализирует потоки начальными данными. Информация для каждого потока хранится в виде структуры. Далее программа создает потоки и передает им необходимые данные для подсчета локального среднего арифметического с помощью функции pthread\_create. После окончания работы всех потоков, локальные средние арифметические складываются. В результате получается среднее арифметическое всех чисел файла. Результат выводится на экран в виде десятичного числа.

**Текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <ctype.h>

#include <math.h>

#define SIZE\_NUM 32

typedef unsigned \_\_int128 int128\_t;

int128\_t Atonum(char\* bufer);

void \*ThreadFunction(void \*tmpPtr);

void PrintResult(int128\_t result);

typedef struct {

long long countNumbers;

int128\_t localResult;

\_\_off\_t startPos;

long long localCountNumbers;

} ThreadParam;

long long threadCount;

long long memoryCount;

char filename[256];

int main(int argc, char \*\*argv)

{

int fn;

unsigned int localCountNumber;

unsigned int localCountNumbersLast;

if (argc != 3) {

printf("Недопустимое количество параметров\n");

exit(-1);

}

memoryCount = atoi(argv[1]);

if (memoryCount <= 0) {

printf("Количество памяти должно быть больше 0\n");

exit(-1);

}

threadCount = atoi(argv[2]);

if (threadCount <= 0) {

printf("Количество потоков должно быть больше 0\n");

exit(-1);

}

if ((threadCount \* (sizeof(ThreadParam) + sizeof(pthread\_t))) > memoryCount) {

printf("Недостаточно памяти для заданного числа потоков\n");

exit(-1);

}

ThreadParam \*threadParam = (ThreadParam\*)malloc(sizeof(ThreadParam) \* threadCount);

pthread\_t \*threads = (pthread\_t\*)malloc(sizeof(pthread\_t) \* threadCount);

printf("Введите имя файла с числами\n ");

scanf("%s",filename);

if ((fn = open(filename, O\_RDONLY)) == -1) {

printf("Не удалось открыть файл\n");

exit(-1);

}

long long size = lseek(fn, 0, SEEK\_END);

close(fn);

if (size == 0) {

printf("Файл пуст\n");

exit(-1);

}

long long numCount = size % (SIZE\_NUM + 1);

if (numCount != 0 ) {

printf("Неверный формат файла\n");

exit(-1);

}

numCount = size / (SIZE\_NUM + 1);

if (threadCount > numCount) {

threadCount = numCount;

}

localCountNumbersLast = numCount % threadCount;

if (localCountNumbersLast != 0) {

localCountNumber = numCount/ threadCount;

localCountNumbersLast = numCount - localCountNumber \* (threadCount - 1);

}

else {

localCountNumber = numCount / threadCount;

localCountNumbersLast = localCountNumber;

}

threadParam[0].countNumbers = numCount;

threadParam[0].localResult = 0;

threadParam[0].startPos = 0;

threadParam[0].localCountNumbers = localCountNumber;

for (int i = 1; i < threadCount; ++i) {

threadParam[i].countNumbers = numCount;

threadParam[i].localResult = 0;

threadParam[i].startPos = i \* (threadParam[i - 1].localCountNumbers \* (SIZE\_NUM + 1));

threadParam[i].localCountNumbers = localCountNumber;

}

threadParam[threadCount - 1].localCountNumbers = localCountNumbersLast;

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

pthread\_create(&threads[i], NULL, ThreadFunction, (void \*) &threadParam[i]);

}

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

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

}

int128\_t result = 0;

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

result += threadParam[i].localResult;

}

PrintResult(result);

return 0;

}

int128\_t Atonum(char\* bufer) {

int128\_t num = 0;

while(\*bufer) {

if (isalpha(\*bufer)) {

num = num \* 16 + (\*bufer - '7');

}

if (isdigit(\*bufer)) {

num = num \* 16 + (\*bufer - '0');

}

++bufer;

}

return num;

}

void \*ThreadFunction(void \*tmpPtr)

{

ThreadParam \*threadParam = (ThreadParam \*)tmpPtr;

char bufer[SIZE\_NUM + 1];

char ch;

int fn = open(filename, O\_RDONLY);

lseek(fn, threadParam->startPos, SEEK\_SET);

for (int i = 0; i < threadParam->localCountNumbers; ++i)

{

read(fn,bufer,SIZE\_NUM);

bufer[SIZE\_NUM] = '\0';

int128\_t number;

number = Atonum(bufer);

number /= threadParam->countNumbers;

threadParam->localResult += number;

read(fn, &ch, 1);

if ((ch != '\n') && (ch != '\0')) {

printf("Неверный формат файла\n");

exit(-1);

}

}

close(fn);

return 0;

}

void PrintResult(int128\_t result) {

double length = 128 \* log10(2);

int len = (int)(length) + 2;

int i = 0, k = len - 1;

int num[len];

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

num[j] = 0;

}

while (result != 0) {

int temp = result % 10;

num[i] = temp;

result = result / 10;

++i;

}

while (num[k] == 0 && k != 0) {

--k;

}

for (int j = k; j > 0; --j) {

printf("%d",num[j]);

}

printf("%d\n",num[0]);

}

**Тестирование**

user@lubuntu:~/OC/lab3$ ./a.out

Недопустимое количество параметров

user@lubuntu:~/OC/lab3$ ./a.out 1000 20 1

Недопустимое количество параметров

user@lubuntu:~/OC/lab3$ ./a.out -1000 20

Количество памяти должно быть больше 0

user@lubuntu:~/OC/lab3$ ./a.out 1000 -20

Количество потоков должно быть больше 0

user@lubuntu:~/OC/lab3$ ./a.out 1000 20

Недостаточно памяти для заданного числа потоков

user@lubuntu:~/OC/lab3$ cat test1.dat

user@lubuntu:~/OC/lab3$ ./a.out 1000 10

Введите имя файла с числами

test1.dat

Файл пуст

user@lubuntu:~/OC/lab3$ cat test2.dat

ADC899CE9156CDE23BDEBC88A8292064

AFBC607CF9EE944902B7B2752B8DBD8C

31B260CF4B441CA9951EE5519626A584

DF1DB9F50C680CDA42F4E4ECA15AEA2A

3E79EDAEDD12D9B95B61895DDD3F4DD4user@lubuntu:~/OC/lab3$

user@lubuntu:~/OC/lab3$ ./a.out 100000 10

Введите имя файла с числами

test2.dat

Неверный формат файла

user@lubuntu:~/OC/lab3$ cat test3.dat

ADC899CE9156CDE23BDEBC88A8292064

AFBC607CF9EE944902B7B2752B8DBD8C

31B260CF4B441CA9951EE5519626A584

DF1DB9F50C680CDA42F4E4ECA15AEA2A

3E79EDAEDD12D9B95B61895DDD3F4DD4

user@lubuntu:~/OC/lab3$ ./a.out 10000 10

Введите имя файла с числами

test3.dat

182053362048839137574709114752585534433

user@lubuntu:~/OC/lab3$ cat test4.dat

4E542A733CF3F053DF7B5A9EFC1385E4

70BB0D9B2E722D92EAEAAFC3B3EB742F

4E27D4C28B3668B9FC3D0A90CDBD6DD9

ADC899CE9156CDE23BDEBC88A8292064

AFBC607CF9EE944902B7B2752B8DBD8C

31B260CF4B441CA9951EE5519626A584

DF1DB9F50C680CDA42F4E4ECA15AEA2A

3E79EDAEDD12D9B95B61895DDD3F4DD4

4C9CC6FE40F3D99AC5BFFBCDD8C4AFEC

1C1E8F0B51EC3EC4E7DFADABD0B55CE6

4FD6FCAAB4E9FEB5C4D86A4039E58FAF

5F0F6AD338CD2E0AEC9AEADA725AA838

28FECCE020F11E78EAEC8D7A33D75EAC

2E261EFF93546D4256B99B21105AC72E

C8EFEAFB9A0D34A68158B4BF91D877BD

F82585D946D985B56E14CAAE60CF88AE

06AA2239EC0E32F9A0D7AC9BC357AE31

C2C2F3F0A80F8CB4AE9D53F8E356EABB

ADAB3F5ECFED0FC8455A4FBCEE6CB4CC

7D6A3F4DEDB6615B96DE4EBEBE70BA9F

1B8DFC3358C8F61A83D29200C00E36BB

D4054C7AB478C8FE6F123CA0F92D4DB2

BF581DD007F7EFFC0CBF2BD2EAD3D616

A0DCABDDEDF52AE8397A1B888DD0A414

2EF55ED5DD7BFF3CDF90F2F85B422ADD

02A80FDC3AB5E5F5AA3CEFDF6FED9181

7A4D47B7CF04CD58521C25EF06417844

E3628B050A4ED4C1F5EB43B1D6F9BEBC

6F43D08DBFC8D68D40BA36FFF8C2E378

3D6A55110D7F8ABF86982962A9BFA9F6

F715E1892ECD710FECBAB3AC6776ABBA

C0D80E6028E6E4F6D067FD52AA9FFE69

ADF67A7FC6FF9EEC0BABF09CF83E51DD

BEA6E6C4E0E8FA3E43EE278D8E65FB8C

EA75B9C0028DE4834EACAE36A994D5AF

2B5FA16BFB5D355460B24A6BAD044844

430A47597686E9ADBB4EDF6ED59BB893

54DA281C7A7A940FBB0AFB0DFC326FC6

3E3F6965BF1B49CDDD823AB8DDA9E05C

2EABA609B4FCB4D113F6E362D52DCED1

C8DCD661C50C816F9B20268A8B6CBD02

50600059C57F58CB12A9EDC184A2E2B7

4F31EA3A814FCEFCAAEC88688CFCC699

AA98A54C775F9E6D1E938A6ED7B9ED4C

C94A6858CC4D86302C62678DEB484644

840902BACA72FFBAB0FCF8891B4CAC99

7A23C18F483346EFFDDF52BF32C30F6C

AD20A306A4AD8DCC4AB65AC28FEFA492

6BDF7FB53BE01610FED77FF573C1CAC0

AAB599260BFA0134F28AB5CFDC04510B

user@lubuntu:~/OC/lab3$ ./a.out 100 1

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 400 1

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 400 7

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 10000 7

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 100000000 1

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 100000000 10000

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ ./a.out 1000000000 100000

Введите имя файла с числами

test4.dat

165237650543654303602955194562535741301

user@lubuntu:~/OC/lab3$ cat test5.dat

2E261EFF93546D4256B99B21105AC72E

2E261EFF93546D4256B99B21105AC72E

2E261EFF93546D4256B99B21105AC72E

2E261EFF93546D4256B99B21105AC72E

2E261EFF93546D4256B99B21105AC72E

user@lubuntu:~/OC/lab3$ ./a.out 10000 57

Введите имя файла с числами

test5.dat

61342423807796366465244380538292848430

user@lubuntu:~/OC/lab3$ bc

bc 1.06.95

Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.

This is free software with ABSOLUTELY NO WARRANTY.

For details type `warranty'.

ibase=16

2E261EFF93546D4256B99B21105AC72E

61342423807796366465244380538292848430

quit

user@lubuntu:~/OC/lab3$

**Выводы**

В ходе данной лабораторной работы я понял, что использование нескольких потоков в рамках одного процесса позволяет разделить задачу и одновременно решать несколько подзадач. При умелом использовании это способно сильно увеличить быстродействие программы. В моем варианте лабораторной работы это позволяет одновременно вычислять среднее арифметическое для нескольких чисел. Между тем может возникнуть проблема синхронизации потоков. Если потоки одновременно будут менять данные в одном участке памяти, программа может работать неправильно. Также следует дождаться завершения работы всех потоков, прежде чем извлекать из них результирующие данные, так как некоторые потоки могли еще не успеть получить результат. Я считаю, что навыки по работе с потоками очень важны, так как потоки широко используются в программировании и помогают решать множество важных задач.