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: Гречников А.В.

Группа: М8О–206Б–20

Вариант: 13

Преподаватель: Соколов Андрей Алексеевич

Дата: 12.01.2022

Оценка: 3

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Постановка задачи**

## Задание

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.

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

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 13:

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**

Алгоритм моей программы не сильно отличает от классического, который описан на первых страницах любого учебника по математике средней школы. Единственное отличие - это многопоточность, но и ее можно избежать, если использовать один поток. Для хранения 128битных чисел я использую редко используемый тип int128. У меня есть две структуры - в первой хранятся пользовательские данные: количество потоков и размер оперативной памяти, во второй - параметры одного потока. Файл с числами у меня перегенерируется при каждом запуске программы, это действие изолировано в функции generate(), в случае чего этого можно избежать. После того, как я узнаю, сколько чисел записано в файле, я инициализирую каждый поток начальными данными в функции init(), там же я распределяю числа по потокам. Далее создаю потоки, в которые зачитываю числа этого потока, и там считаю локальные средние арифметические. Далее объединяю потоки и складываю локальные суммы. Остается только напечатать 128битное число в десятичном представлении.

Из библиотеки pthreads использовались следующие функции:

pthreadcreate() - создание нового потока.

pthreadjoin() - ожидает завершение потока, переданного в аргументах.

Системные вызовы:

void exit(int status) - выходит из процесса с заданным статусом.

int close(int fd) - закрывает файловый дескриптор.

int open(const char \*path, int oflag, ...) - открытие файлового дескриптора: первый аргумент - путь до файла, второй - флаги открытия.

int write(int fd, void \*buffer, int nbyte) - записывает количество байтов в 3 аргументе из буфера в файл с дискриптором fd.

offt lseek(int fd, offt offset, int whence) - устанавливает смещение для файлового дескриптора в значение аргумента offset в соответствии с директивой whence.

**Листинг программы**

**main.c:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <time.h>

#include <inttypes.h>

#include <limits.h>

#include <string.h>

#define DEC\_SIZE 40

#define NUMBER\_SIZE 32

#define FILE\_SIZE 100

typedef unsigned \_\_int128 int128\_t;

void \*thread\_function(void \*);

int counter = 0;

typedef struct \_params {

long long nc;

int128\_t localsum;

unsigned int counter;

off\_t start\_pos;

long long cc;

} Params;

typedef struct \_command {

int threads\_num;

int memory\_set;

} Command;

const char \*file\_name = "test.dat";

void init(Params \*ptr, Command \*command, long long num\_count);

void generate();

void parse\_command\_line(int argc, char \*\*argv, Command \*command);

void print\_int128(int128\_t u128);

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

{

Command command;

parse\_command\_line(argc, argv, &command);

if (command.threads\_num < 0 || command.memory\_set < 0) {

fprintf(stderr, "%s\n", "ThreadsNumber and RAM must be > 0");

exit(EXIT\_FAILURE);

}

if (command.threads\_num \* sizeof(Params) + command.threads\_num \* sizeof(pthread\_t) > command.memory\_set) {

fprintf(stderr, "%s\n", "Too much threads for this amount of memory");

exit(EXIT\_FAILURE);

}

pthread\_t \*thread\_id = (pthread\_t \*) malloc(command.threads\_num \* sizeof(pthread\_t));

Params \*params = (Params \*) malloc(command.threads\_num \* sizeof(Params));

int i, j;

generate();

int fd = open(file\_name, O\_RDWR| O\_CREAT, 0666);

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

close(fd);

long long num\_count = size / (NUMBER\_SIZE + 1);

init(params, &command, num\_count);

for (i = 0; i < command.threads\_num; ++i) {

pthread\_create(&thread\_id[i], NULL, thread\_function, (void \*) &params[i]);

}

for (j = 0; j < command.threads\_num; ++j) {

pthread\_join(thread\_id[j], NULL);

}

int128\_t sum = 0;

for (int i = 0; i < command.threads\_num; ++i) {

sum += params[i].localsum;

}

print\_int128(sum);

return 0;

}

void init(Params \*ptr, Command \*command, long long num\_count)

{

ptr[0].nc = num\_count;

ptr[0].localsum = 0;

ptr[0].counter = 0;

ptr[0].start\_pos = 0;

ptr[0].cc = (num\_count / command->threads\_num);

for (int i = 1; i < command->threads\_num; ++i) {

ptr[i].nc = num\_count;

ptr[i].localsum = 0;

ptr[i].counter = 0;

ptr[i].start\_pos = i \* (ptr[i - 1].cc \* (NUMBER\_SIZE + 1));

ptr[i].cc = ptr[i-1].cc;

}

ptr[command->threads\_num - 1].cc += num\_count % command->threads\_num;

}

void generate()

{

int fd = open(file\_name, O\_RDWR| O\_CREAT, 0666);

char buf[NUMBER\_SIZE];

srand(time(NULL));

for (int i = 0; i < FILE\_SIZE; ++i) {

for (int i = 0; i < NUMBER\_SIZE; ++i) {

if (((int) rand()) % 2 == 0) {

buf[i] = '0' + (((int) rand()) % 10);

} else {

buf[i] = 'A' + (((int) rand()) % 6);

}

}

write(fd, &buf, NUMBER\_SIZE);

write(fd, "\n", 1);

}

close(fd);

}

void print\_int128(int128\_t u128)

{

printf("DEC: \n");

char buf[DEC\_SIZE + 1];

int i;

for (i = 0; i < DEC\_SIZE; ++i) {

buf[i] = '0';

}

buf[DEC\_SIZE] = '\0';

for (i = DEC\_SIZE - 1; u128 > 0; --i) {

buf[i] = (int) (u128 % 10) + '0';

u128 /= 10;

}

if (i == DEC\_SIZE - 1) {

printf("%d\n", 0);

} else {

printf("%s\n", &buf[i + 1]);

}

}

int is\_num(char \*s)

{

return (\*s >= '0' && \*s <= '9');

}

int hex\_to\_dec(char \*s)

{

if(\*s == 'A')

return 10;

if(\*s == 'B')

return 11;

if(\*s == 'C')

return 12;

if(\*s == 'D')

return 13;

if(\*s == 'E')

return 14;

if(\*s == 'F')

return 15;

return 0;

}

int128\_t atobigint(char \*str)

{

int128\_t res = 0;

while (\*str) {

if(is\_num(str))

res = res \* 16 + (\*str - '0');

else {

int kek = hex\_to\_dec(str);

res = res \* 16 + kek;

}

++str;

}

return res;

}

void \*thread\_function(void \*dummyPtr)

{

Params \*ptr = (Params \*)dummyPtr;

char buf[NUMBER\_SIZE + 1];

char c;

int fd = open(file\_name, O\_RDWR| O\_CREAT, 0666);

lseek(fd, ptr->start\_pos, SEEK\_SET);

for (int i = 0; i < ptr->cc; ++i)

{

read(fd,buf,NUMBER\_SIZE);

buf[NUMBER\_SIZE] = '\0';

int128\_t s;

s = atobigint(buf);

s /= ptr->nc;

ptr->localsum += s;

read(fd, &c, 1);

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

fprintf(stderr, "%s %d\n","c = ", c);

fprintf(stderr, "%s %ld\n","start pos = ", ptr->start\_pos);

fprintf(stderr, "%s %lld\n","cc = ", ptr->cc);

fprintf(stderr, "%s\n", "format error");

exit(EXIT\_FAILURE);

}

}

close(fd);

return 0;

}

void parse\_command\_line(int argc, char \*\*argv, Command \*command)

{

if (argc != 3) {

fprintf(stderr, "%s\n","Usage: ThreadsNumber RAM");

exit(EXIT\_FAILURE);

}

command->threads\_num = atoi(argv[1]);

command->memory\_set = atoi(argv[2]);

}

**Makefile:**

CC = gcc

FLAGS = -std=c99 -pthread -w -pipe -O3 -Wextra -Werror -Wall -Wno-sign-compare -pedantic -lm

all: run

run: main.o

$(CC) $(FLAGS) $^ -o $@

clean:

rm -f \*.o run

**Пример работы**

artem@ideapad:~$ ./run

Usage: ThreadsNumber RAM

artem@ideapad:~$ ./run 2

Usage: ThreadsNumber RAM

artem@ideapad:~$ ./run -10 200

ThreadsNumber and RAM must be > 0

artem@ideapad:~$ ./run 10 -200

ThreadsNumber and RAM must be > 0

artem@ideapad:~$ ./run 123 987

Too much threads for this amount of memory

artem@ideapad:~$ cat test.dat

B4DCCDA60B399AFE9D4AEE8006AB1B0B

artem@ideapad:~$ bc

bc 1.07.1

Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.

This is free software with ABSOLUTELY NO WARRANTY.

For details type `warranty'.

ibase = 16

B4DCCDA60B399AFE9D4AEE8006AB1B0B

240407515599480220982277249670086073099

artem@ideapad:~$ ./run 1 1000

DEC:

240407515599480220982277249670086073099

artem@ideapad:~$ make -f Makefile

gcc -c -o main.o main.c

gcc -std=c99 -pthread -w -pipe -O3 -Wextra -Werror -Wall -Wno-sign-compare -pedantic -lm main.o -o run

artem@ideapad:~$ ./run 15 10000

DEC:

173171873828711844299539089646540054151

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как производится работа с потоками в языке Си. Основная идея применения потоков состоит в разбиение большой задачи на множество более мелких задач, которые выполняются параллельно. Но данный подход требует особенной внимательности программиста к совместно используемым данным и тщательного продумывания алгоритма.