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: Чистяков Д. М.

Группа: М8О–206Б–20

Вариант: 18

Преподаватель: Соколов Андрей Алексеевич

Оценка:

Дата: 04.03.2022

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022

**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 18:

Найти образец в строке наивным алгоритмом

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: stdio.h, stdlib.h, string.h, errno.h, unistd.h, pthread.h, sys/time.h. В программе используются следующие системные вызовы:

1. **gettimeofday** ­­–– позволяет получить текущее время. Принимает в качестве аргументов структуру timeval, в которую записывает результат работы, и переменную для коррекции времени. В случае успеха возвращает 0.
2. **pthread\_create ––** (является оберткой над системным вызовом clone) создает новый поток в вызывающем процессе. В качестве аргументов принимает указатель на структуру-идентификатор потока pthread\_t, атрибуты потока, функцию, которая будет запускаться в потоке, список аргументов для функции в виде указателя на void. В случае успеха возвращает 0, иначе возвращает номер ошибки.
3. **pthread\_join ––** используется для ожидания завершения потока. Данная функция блокирует вызывающий поток, пока указанный поток не завершится. В качестве аргументов принимает структуру pthread\_t потока и указатель на переменную, в которую будет записан результат, возвращаемый потоком. В случае успеха возвращает 0, иначе возвращает номер ошибки.
4. **pthread\_exit ––** завершает вызываемый поток. В качестве аргумента принимает значение, которое вернется при завершении потока. Функция всегда завершается успехом.
5. **read ––** предназначена для чтения какого-то числа байт из файла, принимает в качестве аргументов файловый дескриптор, буфер, в который будут записаны данные и число байт. В случае успеха вернет число прочитанных байт, иначе -1.

**Общий метод и алгоритм решения**.

Для реализации, поставленной задачи необходимо:

1. Изучить принципы работы с потоками в C.
2. Написать наивный алгоритм поиска подстроки в строке.
3. С помощью принципа разбиения задачи на маленькие подзадачи, реализовать функцию, которая будет проводить частичную обработку данных, для ее запуска в многопоточном режиме.
4. Реализовать функцию, которая будет создавать потоки, запускать потоковую функцию, управлять потоками.
5. Реализовать обработку системных ошибок согласно заданию

**Основные файлы программы**

**main.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/time.h>

#define MAX\_BUF 65534

typedef struct

{

    int l;

    int r;

}

data;

char\* text;

int len\_text;

char\* str;

int len\_str;

void\* func(void\* arg)

{

    int l = ((data\*)arg)->l;

    int r = ((data\*)arg)->r;

    for(int i = l; i <= r; i++)

    {

        int state = 1;

        for (int j = 0; j < len\_str; j++)

        {

            if (text[i + j] != str[j])

            {

                state = 0;

                break;

            }

        }

        if(state)

        {

            printf("i: %d\n", i);

        }

    }

    pthread\_exit(0);

}

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

{

    if(argc != 2)

    {

        perror("invalid args\n");

        return 1;

    }

    int count = atoi(argv[1]);

    if(count <= 0)

    {

        perror("ERROR\n");

        return 1;

    }

    text = malloc(MAX\_BUF);

    if (read(0, text, MAX\_BUF) < 0)

    {

        perror("ERROR\n");

        return 1;

    }

    len\_text = strlen(text) - 1;

    text[len\_text] = '\0';

    str = malloc(MAX\_BUF);

    if (read(0, str, MAX\_BUF) < 0)

    {

        perror("ERROR\n");

        return 1;

    }

    len\_str = strlen(str) - 1;

    str[len\_str] = '\0';

    int num = len\_text - len\_str + 1;

    count = count > num ? num : count;

    int celoe = num / count;

    int ostatok = num % count;

    pthread\_t\* id = malloc(count \* sizeof(pthread\_t));

    data\* arg = malloc(count \* sizeof(data));

    struct timeval start, end;

    gettimeofday(&start, NULL);

    int l = 0;

    int r = -1;

    for(int i = 0; i < count; i++)

    {

        l = r + 1;

        r += celoe;

        if(ostatok > 0)

        {

            r++;

            ostatok--;

        }

        arg[i].l = l;

        arg[i].r = r;

        if (pthread\_create(&id[i], NULL, func, &arg[i]) != 0)

        {

            perror("Can't create a thread.\n");

        }

    }

    for (int i = 0; i < count; i++)

    {

        if (pthread\_join(id[i], NULL) != 0)

        {

            perror("Can't wait for thread\n");

        }

    }

    gettimeofday(&end, NULL);

    long sec = end.tv\_sec - start.tv\_sec;

    long microsec = end.tv\_usec - start.tv\_usec;

    if (microsec < 0)

    {

        --sec;

        microsec += 1000000;

    }

    long elapsed = sec\*1000000 + microsec;

    printf("time: %ld ms\n", elapsed);

    return 0;

}

**Пример работы**

den@den-VirtualBox:~/Документы/lab3\_18$ ./a.out 1

hello world hello world hello world

world

i: 6

i: 18

i: 30

time: 253 ms

den@den-VirtualBox:~/Документы/lab3\_18$ ./a.out 2

Food is an important substance for human organism. It mainly consists of proteins, fats and carbohydrates. Besides these three nutritive materials, food can also be nodivided into 2 groups – meals of plant origin and meals of animal origin.

of

i: 199

i: 225

i: 70

time: 282 ms

**Вывод**

В процессе выполнения данной лабораторной работы я научился работать с потоками на языке Си для ОС Линукс. Изучил принципы организации потоков в многопоточных программах, а также способы разбиения алгоритмов на множество более мелких задач, выполняющихся параллельно.