Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа № 3 по курсу
«Операционные системы»

Студент: Тулин Иван Денисовчи
Группа: М8О-201Б-21
Вариант: 17
Преподаватель Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/YusayuSharingan/Opsys-labs/tree/main/lab3

Постановка задачи

Цель работы

Научиться создавать потоки, и взаимодействовать с ними.

Задание

17)Найти в большом целочисленном массиве минимальный элемент

Общие сведения о программе

Программа компилируется из файла lab3.cpp. Скомпилированная программа при запуске принимает аргумент — количество потоков. В программе используются следующие системные вызовы:

1. pthread_mutex_init() - создает mutex
2. pthread_mutex_lock() – блокирует все остальные потоки, до вызова pthread_mutex_unlock()
3. pthread_mutex_unlock() – говорит о конце блокировки после pthread_mutex_lock()
4. pthread_mutex_destroy() -- уничтожает mutex
5. pthread_create() – создает поток.
6. pthread_join() – закрываает поток.

Общий метод и алгоритм решения

Входной массив параллельно считывается разными потоками, но переменная с которой сравнивается и в которую записывается минимальный элемент одна общая для всех потоков.

Таким образом массив обрабатывается разными потоками.

Исходный код

lab3.cpp

```
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>
#include <pthread.h>

pthread_mutex_t mutex;
const int STOP_SIG = -1;
```

```
void* ThreadFunc(void* arg){
    int num, *minV=(int*)arg;
    while(scanf("%d", &num)!=STOP_SIG){
        pthread_mutex_lock(&mutex);
        if(num < *minV){
            *minV = num;
        }
        pthread_mutex_unlock(&mutex);

    }
    return NULL;
}


int main_routine(int threadNum){
    int minValue = INT_MAX;
    pthread_t *ths = (pthread_t*)malloc(sizeof(pthread_t)*threadNum);
    if (ths == NULL){
        printf("ERROR: couldn't allocate memmmory\n");
        return EXIT_FAILURE;
    }
    pthread_mutex_init(&mutex, NULL);
    for (int i=0; i<threadNum; i++){
        if(pthread_create(ths + i, NULL, ThreadFunc, &minValue)){
            printf("ERROR: couldn't create thread\n");
            return EXIT_FAILURE;
        }
    }
    for(int i=0; i<threadNum; i++){
        pthread_join(ths[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    return minValue;
}
```

Демонстрация работы программы

```
yorokobeshounen@YS:~/Рабочий стол/OpSys/lab3/build$ cat input.txt
8991 993 34 345609 43185 4 5643132165 1
yorokobeshounen@YS:~/Рабочий стол/OpSys/lab3/build$ ./lab3 5 < input.txt
Min value is 1
Time of executing: 0
```

Выводы

Чтобы проверить, реально ли ускоряют потоки программы я специально сделал два одинаковых input, отличающихся только количеством потоков. Из фотографий результата видно, что программа, работающая на 5 потоках значительно быстрее работает, чем программа работающая на 1 потоке... Также хочется отметить, что с потоками намного проще и интреснее работать, тк для их взаимодействий не нужно создавать pipe. Да и в целом потоки очень полезная, ускоряющая программу, штука.