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: Ворошилов Кирилл Сергеевич
Группа: М8О-201Б-21
Вариант: 18
Преподаватель Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/kiviyi/operation-system/tree/lab-3

Постановка задачи

Цель работы

Научиться создавать потоки, и взаимодействовать с ними.

Задание

Найти образец в строке наивным алгоритмом.

Общие сведения о программе

Программа компилируется из файла main.cpp. В нее подается 3 значения: строка, в которой искать, подсрока, которую нужно найти и количество потоков. В программе используются следующие системные вызовы:

1. pthread_mutex_lock() – блокирует все остальные потоки, до вызова pthread_mutex_unlock()
2. pthread_mutex_unlock() – говорит о конце блокировки после pthread_mutex_lock()
3. dup2() – копирует old_file_descriptor в new_file_descriptor.
4. std::chrono::steady_clock::now() – делает замеры по времени(для вычисления скорости работы программы.
5. pthread_create() – создает поток.
6. **pthread_join() – закрываает поток.**

Общий метод и алгоритм решения

разделяем всю строку на подстроки, причем количество подстрок зависит от количества потоков и он проверяет.

Исходный код

```
#include "include/lab.h"
#include <vector>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
```

```
#include <string>
#include <pthread.h>
using namespace std;

int main() {
    string input;

    string p;
    string s;
    int CountTread;
    getline(cin, p);
    getline(cin, s);
    cin >> CountTread;
    string output = NaivSearc(p, s, CountTread);

    for (const auto &res : output){
        cout << res;
    }
    cout << endl;
    return 0;
}
#include "../include/lab.h"
#include <unistd.h>
#include <vector>
#include <stdio.h>
#include <stdlib.h>
#include <thread>
#include <pthread.h>
#include <iostream>
#include <chrono>
using namespace std;

void srav(string str, string sub, int &b){
    int rez = str.find(sub);
    if(rez < str.length() && rez >= 0){
        b += 1;
    }
```

```
}


string NaivSearc(string str, string substr, int CountTread){
    int rez = 0;
    int plu = str.size()/substr.size();
    int actualThr = min(CountTread, plu);
    vector<thread> threads;
    threads.reserve(actualThr);
    auto start = std::chrono::high_resolution_clock::now();
    for(int i=0; i<str.size(); i += plu){
        if(i + plu + substr.size() >= str.size()){
            threads.emplace_back(srav, str.substr(i), substr, ref(rez));
        } else {
            // cout << str.substr(i, i + plu + substr.size()) << endl;
            threads.emplace_back(srav, str.substr(i, i + plu + substr.size()), substr, ref(rez));

        }
    }
    for(auto& thread : threads) {
        thread.join();
    }
    auto end = std::chrono::high_resolution_clock::now();
    auto searchTime = std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count();

    cout << searchTime << endl;

    if(rez > 0){
        return "True";
    } else{
        return "False";
    }
```

```
    // return output;
}
```

Демонстрация работы программы

INPUT:

```
asd;fasdjfasd;f         sd
```

OUTPUT:

```
kivi@LAPTOP-DGOM1IRE:~/os/lab_3/lab3/l3$ ./lab3

asd;fasdjfasd;f

sd

3

0

True
```

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.