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: Дегтярев Денис Андреевич
Группа: М8О-207Б-21
Вариант: 15
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/CHISH08/OCI/tree/main/3lab

Постановка задачи

Цель работы

Научиться создавать потоки, и взаимодействовать с ними.

Задание

Перемножение полиномов. На вход подается N-полиномов, необходимо их перемножить.

Общие сведения о программе

Программа компилируется из файла 3lab.cpp. В нее подается файл input3.txt, в котором содержится количество потоков, количество полиномов, максимальная степень входящих полиномов, а затем сами полиномы.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

Распределяем полиномы на N / 2 или K(если K < N / 2) пар. Каждая пара полиномов перемножается внутри потоков. Это повторяется, пока N != 1, а затем печатается ответ.

Исходный код

#include <bits/stdc++.h>

#include <sys/wait.h>

#include <unistd.h>

#include <fcntl.h>

#include <signal.h>

#include <pthread.h>

#include <chrono>

```
#include <ctime>
using namespace std;

pthread_mutex_t mtx;

typedef struct {
   int N;
   vector<vector<long long>> pol_mtx;
} dat;

dat dt;

void* mult(void *arg)
{
   pthread_mutex_lock(&mtx);
   vector<long long> m1, m2;
   int N = dt.pol_mtx.size();
   m1 = dt.pol_mtx[N - 1];
   m2 = dt.pol_mtx[N - 2];
   dt.pol_mtx.pop_back();
   dt.pol_mtx.pop_back();
   pthread_mutex_unlock(&mtx);

   int K1 = m1.size(), K2 = m2.size();
   vector<long long> m3(K1 + K2 - 1, 0);

   for (int i = 0; i < K1; ++i) {
      for (int j = 0; j < K2; ++j) {
         m3[i + j] += m1[i] * m2[j];
      }
```

```
    }
    pthread_mutex_lock(&mtx) == 0;
    dt.pol_mtx.push_back(m3);
    --dt.N;
    pthread_mutex_unlock(&mtx);
    return NULL;
}

int main()
{
    pthread_mutex_init(&mtx, NULL);
    int file1 = open("./input3.txt", O_RDONLY);
    dup2(file1, STDIN_FILENO);

    int K, N, th_c;

    //cout << "Введите количество потоков: ";
    cin >> th_c;

    pthread_t threads[th_c];

    //cout << "\nВведите количество полиномов: ";
    cin >> N;

    //cout << "\nВведите максимальную степень вводимых полиномов: ";
    cin >> K;

    dt.N = N;

    for (int i = 0; i < N; ++i) {
```

```
        vector<long long> pol(K + 1, 0);
        for (int j = K; j >= 0; --j) {
            // cout << "\nВведите полином " << j << " степени: ";
            cin >> pol[j];
        }
        (dt.pol_mtx).push_back(pol);
    }

        auto start_time = std::chrono::steady_clock::now();
    delete new int(1);

    while(dt.N > 1) {
        int j = 0;
        for (int i = 0; i < th_c && i < dt.N / 2 && dt.N > 1 && dt.pol_mtx.size()
>= 2; ++i) {
            ++j;
            if (pthread_create(&threads[i], NULL, &mult, NULL) != 0) {
                perror("Failed to creating thread!");
            }
        }

        for(int i = 0; i < j; ++i) {
                if (pthread_join(threads[i], NULL) != 0) {
                perror("Failed to joining thread!");
            }
        }
    }

        auto end_time = std::chrono::steady_clock::now();
```

```
    auto elapsed_ns =
std::chrono::duration_cast<std::chrono::nanoseconds>(end_time -
start_time);
    std::cout << elapsed_ns.count() << " ns\n";

    pthread_mutex_destroy(&mtx);

    for (int j = 0; j < dt.pol_mtx.size(); ++j) {
        for (int i = dt.pol_mtx[j].size() - 1; i >= 0; --i) {
            cout << dt.pol_mtx[j][i] << "x^" << i;
            if (i != 0) {
                cout << " + ";
            } else {
                cout << '\n';
            }
        }
    }

    return 0;
}
```

Демонстрация работы программы

input1:

```
1
10
4
1 5 0 4 2
1 0 5 -3 6
1 830 34 42 1
1 4325 563 -4 0
1 0 0 0 5
1 -3 0 0 0
1 0 4 1 19
1 4 2 8 5
1 0 0 9 0
1 23 -6 14 8
```

output1:

```
~/home/Osy/3lab ./a.out
1236939 ns
1x^40 + 5184x^39 + 3739978x^38 + 105435608x^37 + 506750615x^36 + 404263923x^35 + 1891224966x^34 + 3209884228x^33 + -16627382943x^32 + -12894790298x^31
 + -249446074368x^30 + -329324507295x^29 + -1392105928502x^28 + -2615020904942x^27 + -5265383833440x^26 + -12509160414712x^25 + -12989647800453x^24 +
-396954732206 44x^23 + -30955784582612x^22 + -85673830482490x^21 + -574374922837 33x^20 + -153916117820629x^19 + -79032777864332x^18 + -171103221155556x
^17 + -132539324686425x^16 + -100330357324522x^15 + -119627210363718x^14 + -63520565770783x^13 + -40358168149260x^12 + -27914659051472x^11 + -97020616
13514x^10 + -2082219580680x^9 + -402660447300x^8 + -36462373560x^7 + -463821120x^6 + 4924800x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0
```

Input2:

```
5
10
4
1 5 0 4 2
1 0 5 -3 6
1 830 34 42 1
1 4325 563 -4 0
1 0 0 0 5
1 -3 0 0 0
1 0 4 1 19
1 4 2 8 5
1 0 0 9 0
1 23 -6 14 8
```

output2:

```
~/home/Osy/3lab ./a.out
497915 ns
1x^40 + 5184x^39 + 3739978x^38 + 105435608x^37 + 506750615x^36 + 404263923x^35 + 1891224966x^34 + 3209884228x^33 + -16627382943x^32 + -12894790298x^31
 + -249446074368x^30 + -329324507295x^29 + -1392105928502x^28 + -2615020904942x^27 + -5265383833440x^26 + -12509160414712x^25 + -12989647800453x^24 +
-396954732206 44x^23 + -30955784582612x^22 + -85673830482490x^21 + -574374922837 33x^20 + -153916117820629x^19 + -79032777864332x^18 + -171103221155556x
^17 + -132539324686425x^16 + -100330357324522x^15 + -119627210363718x^14 + -63520565770783x^13 + -40358168149260x^12 + -27914659051472x^11 + -97020616
13514x^10 + -2082219580680x^9 + -402660447300x^8 + -36462373560x^7 + -463821120x^6 + 4924800x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0
```

Результат работы strace:

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990,

parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11581]}, 88) = 11581

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11582]}, 88) = 11582

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11583]}, 88) = 11583

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11584]}, 88) = 11584

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11585]}, 88) = 11585

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11586]}, 88) = 11586

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[11587]}, 88) = 11587

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[0]}, 88) = 11588

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7f09da0990, parent_tid=0x7f7f09da0990, exit_signal=0, stack=0x7f7f095a0000, stack_size=0x7fff80, tls=0x7f7f09da06c0} => {parent_tid=[0]}, 88) = 11589

+++ exited with 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.