Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу
«Операционные системы»

ПРОЦЕССЫ И ПОТОКИ. МЕЖПРОЦЕССОРНОЕ ВЗАИМОДЕЙСТВИЕ

Студент: Злобина Валерия Вадимовна
Группа: М8О–208Б–21
Вариант: 20
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022.

Цель работы
Целью является приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

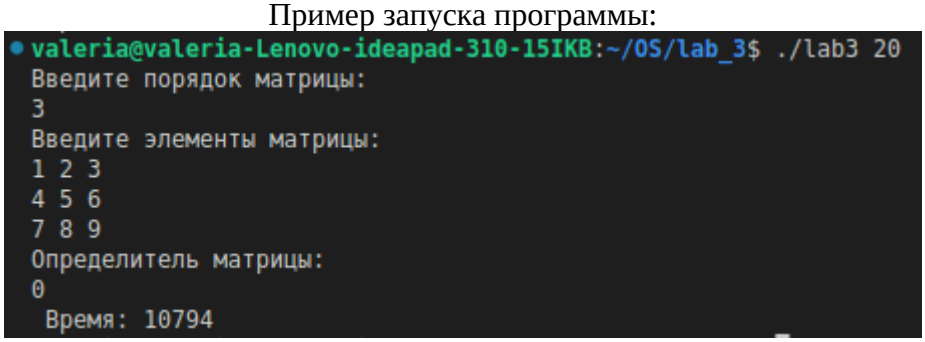
Задание
Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.
Так же необходимо уметь продемонстрировать 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 9
Рассчитать детерминант матрицы

Общие сведения о программе

Программа компилируется из файла lab3.cpp. Количество потоков задаётся ключём программы. Матрицы для тестирования программы генерируются программой matrix_generator.c.

Пример запуска программы:



Исходный код:

lab3.cpp

```
#include<pthread.h>
#include<iostream>
#include<ctime>
#include<vector>
```

```
#include<fstream>
#include<chrono>
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

using namespace std;

int flag = 0;

typedef struct arguments {
  int num_of_thread;
  int total;
  int size;
  vector<int> *minors;
  vector<vector<int>> *matrix;
} arg;

int det(int n, vector<vector<int>> * matrix) {
  int determinant = 0;
  vector<vector<int>> minor(n, vector<int> (n) );
  if (n == 1) {
    return (*matrix)[0][0];
  }
  if (n == 2)
    return (((*matrix)[0][0] * (*matrix)[1][1]) - ((*matrix)[1][0] * (*matrix)[0][1]));
  else {
    for (int x = 0; x < n; x++) {
      int subi = 0;
      for (int i = 1; i < n; i++) {
        int subj = 0;
        for (int j = 0; j < n; j++) {
        if (j == x)
        continue;
        minor[subi][subj] = (*matrix)[i][j];
        subj++;
        }
      subi++;
    }
    determinant = determinant + (pow(-1, x) * (*matrix)[0][x] * det(n - 1, &minor));

    }
  }
return determinant;
}

int count_minor(int n, int x, vector<vector<int>> * matrix) {
  vector<vector<int>> minor(n - 1, vector<int> (n - 1));
  for (int i = 1; i < n; ++i) {
    for (int j = 0; j < x; ++j) {
    minor[i - 1][j] = (*matrix)[i][j];
    }
    for (int j = x + 1; j < n; ++j) {
    minor[i - 1][j - 1] = (*matrix)[i][j];
    }
  }
  return det(n - 1, &minor);
}

void *thread_function(void *arguments) {
  arg args = * (arg *)arguments;
  int num_of_thread = args.num_of_thread;
```

```
  int total = args.total;
  flag = 1;
  vector<int> *minors = args.minors;
  vector<vector<int>> *matrix = args.matrix;
  int size = args.size;

  for (int i = num_of_thread; i < size; i += total) {
    int result = count_minor(size, i, matrix);
    (*minors)[i] = result;
  }
}

int main(int argc, const char **argv) {
int n, i, j;
//cout << "Введите количество потоков: ";
int count_threads;
count_threads = atoi(argv[1]);
//cin >> count_threads;
cout << "Введите порядок матрицы:\n";
cin >> n;
vector<vector<int>> matrix(n, vector <int> (n) );
cout << "Введите элементы матрицы:\n";
for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
cin >> matrix[i][j];

chrono::steady_clock::time_point begin = chrono::steady_clock::now();
vector<int> minors(n, 0);
pthread_t threads[count_threads];
for (int i = 0; i < count_threads; ++i) {
  arg args = {i, count_threads, n, &minors, &matrix};
  pthread_create(&threads[i], NULL, thread_function, &args);
  while (flag == 0) {
  }
  flag = 0;
}
for (int i = 0; i < count_threads; ++i) {
  pthread_join(threads[i], NULL);
}
int result = 0;
for (int i = 0; i < n; ++i) {
  if (i % 2 == 0)
  result += minors[i] * matrix[0][i];
  else
  result -= minors[i] * matrix[0][i];
}

chrono::steady_clock::time_point end = chrono::steady_clock::now();

cout<<"Определитель матрицы:\n"<< result << "\n ";

cout<<"Время: " << chrono::duration_cast<chrono::microseconds>(end-begin).count();
cout<<"\n";
return 0;
}
```

matrix_generator.c

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <unistd.h>

char *itoa(int x) {
int n = 1;
int a = abs(x);
while (a >= 10) {
a /= 10;
n++;
}
char *result;
if (x < 0) {
result = calloc(sizeof(char), n + 1);
a = abs(x);
for (int i = n; i > 0; --i) {
result[i] = a % 10 + 48;
a /= 10;
}
result[0] = '-';
}
else {
result = calloc(sizeof(char), n);
a = x;
for (int i = n - 1; i >= 0; --i) {
result[i] = a % 10 + 48;
a /= 10;
}
}
return result;
}

int main(int argc, const char **argv) {
if (argc != 3) {
char *err = "Error: invalid arguments\n";
write(2, err, strlen(err));
exit(EXIT_FAILURE);
}
```

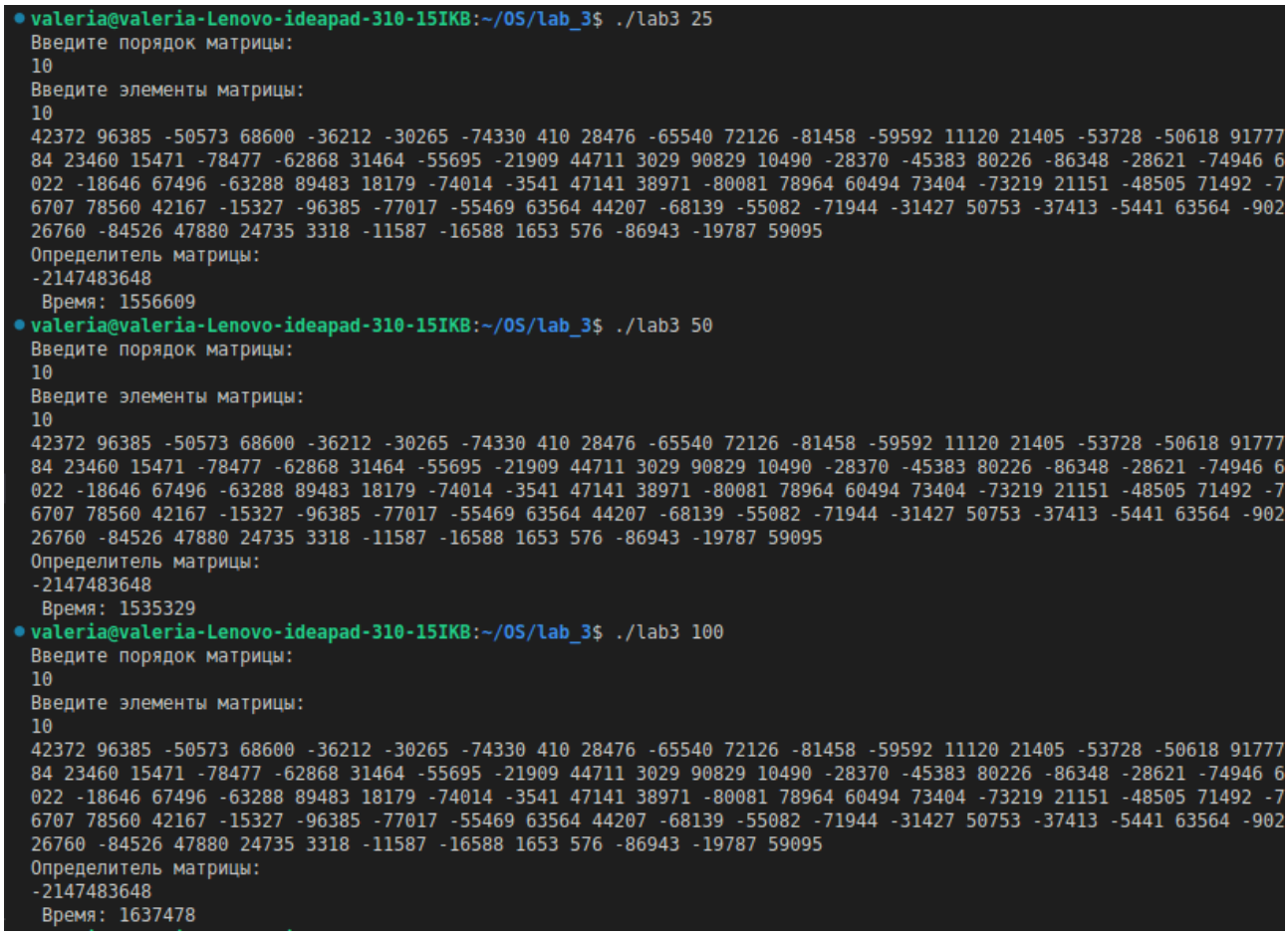
```
int filedes;
if ((filedes = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC, S_IRWXU)) < 0) {
perror(argv[2]);
exit(EXIT_FAILURE);
}
srand(clock());
int x;
const char *numb = argv[1];
write(filedes, numb, strlen(numb));
char nl = '\n';
write(filedes, &nl, 1);
int num = atoi(argv[1]);
for (int i = 0; i < num * num; ++i) {
x = rand() % 200000 - 100000;
char *numb = itoa(x);
write(filedes, numb, strlen(numb));
char space = ' ';
write(filedes, &space, 1);
}
close(filedes);
}
```

Исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков.

Возьмём матрицу порядка 10 и будем при запуске программы увеличивать количество потоков

```
valeria@valeria-Lenovo-ideapad-310-15IKB:~/OS/lab_3$ ./lab3 1
Введите порядок матрицы:
10
Введите элементы матрицы:
10
42372 96385 -50573 68600 -36212 -30265 -74330 410 28476 -65540 72126 -81458 -59592 11120 21405 -53728 -50618 91777
84 23460 15471 -78477 -62868 31464 -55695 -21909 44711 3029 90829 10490 -28370 -45383 80226 -86348 -28621 -74946 6
022 -18646 67496 -63288 89483 18179 -74014 -3541 47141 38971 -80081 78964 60494 73404 -73219 21151 -48505 71492 -7
6707 78560 42167 -15327 -96385 -77017 -55469 63564 44207 -68139 -55082 -71944 -31427 50753 -37413 -5441 63564 -902
26760 -84526 47880 24735 3318 -11587 -16588 1653 576 -86943 -19787 59095
Определитель матрицы:
-2147483648
 Время: 3429804
valeria@valeria-Lenovo-ideapad-310-15IKB:~/OS/lab_3$ ./lab3 10
Введите порядок матрицы:
10
Введите элементы матрицы:
10
42372 96385 -50573 68600 -36212 -30265 -74330 410 28476 -65540 72126 -81458 -59592 11120 21405 -53728 -50618 91777
84 23460 15471 -78477 -62868 31464 -55695 -21909 44711 3029 90829 10490 -28370 -45383 80226 -86348 -28621 -74946 6
022 -18646 67496 -63288 89483 18179 -74014 -3541 47141 38971 -80081 78964 60494 73404 -73219 21151 -48505 71492 -7
6707 78560 42167 -15327 -96385 -77017 -55469 63564 44207 -68139 -55082 -71944 -31427 50753 -37413 -5441 63564 -902
26760 -84526 47880 24735 3318 -11587 -16588 1653 576 -86943 -19787 59095
Определитель матрицы:
-2147483648
 Время: 1422949
```

Вначале мы наблюдаем рост эффективности с увеличением колчества потоков, но потом по мере увеличения потоков время работы программы начинает постепенно увеличиваться.



Так как для создания потоков необходимо время использование слишком большого их количества является не эффективным.