

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

Кафедра вычислительной техники

Отчет по лабораторной работе №3  
по дисциплине «Операционные системы»

Тема: Процессы и потоки

Студент гр. 9308

Преподаватель

Дубенков С.А

Тимофеев А.В.

Санкт-Петербург

2021

## Цель работы

Исследовать механизмы создания и управления процессами и потоками в ОС Windows.

Код обоих заданий представлен на Github:

<https://github.com/sedub01/eltech/tree/main/OS/3>

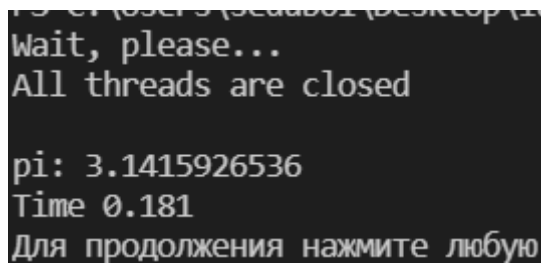
## Задание 3.1

Создайте приложение, которое вычисляет число  $\pi$  с точностью N знаков после запятой по следующей формуле

$$\pi = \left( \frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \dots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, i = \overline{0, N-1},$$

где  $N=1000000000$ .

Был произведен замер времени по кол-ву потоков (1, 2, 4, 8, 12, 16), меняя значение MAX\_THREADS.



```
wait, please...
All threads are closed

pi: 3.1415926536
Time 0.181
Для продолжения нажмите любую
```

Рисунок 1. Пример работы программы на 16 потоках

График представлен ниже.

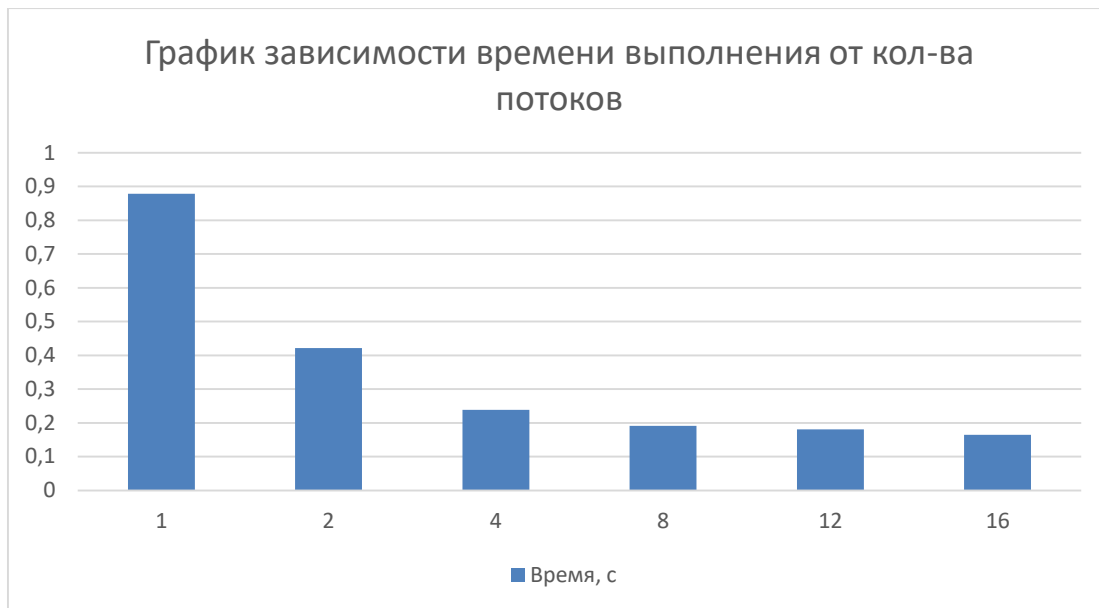


Рисунок 2. График зависимости времени выполнения от кол-ва потоков

Вывод: с увеличением кол-ва потоков уменьшается время выполнения программы, как следствие, повышается ее эффективность. Это происходит потому, что в однопоточной среде вся нагрузка приходилась на одно ядро процессора – теперь же, благодаря распараллеливанию, части программы выполняются одновременно на нескольких ядрах, однако в таком случае необходимо следить за выполнением потоков: за созданием, течением и их удалением.

Сначала время выполнения уменьшалось в два раза, затем темпы уменьшения времени начали снижаться. Оптимальное количество потоков для моей программы – 16 (из 16 доступных)

В современном процессоре одновременно запущены несколько тысяч процессов и потоков, так что написанная программа не входила с ними в конфликт

### Задание 3.2

Создайте приложение, которое вычисляет число  $\pi$  с точностью

N знаков после запятой по следующей формуле

$$\pi = \left( \frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \dots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, i = \overline{0, N-1},$$

где  $N=1000000000$  с помощью OpenMP-стандарта.

Программа была написана на Си, так как C++ компилятор не поддерживал OpenMP функции.

Был произведен замер времени по кол-ву потоков (1, 2, 4, 8, 12, 16), меняя значение MAX\_THREADS.

```
Wait please...  
Time 0.230000  
pi = 3.1415926536
```

Рисунок 3. Пример выполнения программы на 8 потоках

График приведен ниже



Рисунок 4. График зависимости времени выполнения от кол-ва потоков

Вывод: с использованием стандарта OpenMP прослеживается та же зависимость времени выполнения от кол-ва потоков: чем больше потоков, тем меньше время исполнения. Отклонения от результатов предыдущей работы минимальны.

Благодаря использованию этого стандарта программист может не заботиться о создании, протекании и удалении потоков: стандарт OpenMP сделает все за него, однако необходимо хорошо знать этот стандарт, так как отсутствующее или неправильное ключевое слово могут привести к краху программы при ее компиляции.