Минобрнауки России  
Федеральное государственное автономное образовательное  
Учреждение высшего образования  
«Санкт-Петербургский государственный электротехнический  
Университет им. В.И. Ульянова (Ленина)»  
(СПГЭТУ «ЛЭТИ»)  
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники  
  
  
  
**Отчет по лабораторной работе №3**

**на тему: «Процессы и потоки»**

**по дисциплине «Операционные системы»**

Выполнил студент группы 9308: Паникаровская Д. А.

Принял: к.т.н., доцент Тимофеев А. В.

Санкт-Петербург  
2021 г.

Содержание

[Цель работы 3](#_Toc89199665)

[Реализация многопоточного приложения с использованием функций Win32 API 3](#_Toc89199666)

[Указания к выполнению 3](#_Toc89199667)

[Результаты выполнения программы 4](#_Toc89199668)

[Вывод по заданию 8](#_Toc89199669)

[Реализация многопоточного приложения с использованием технологии OpenMP 9](#_Toc89199670)

[Указания к выполнению 9](#_Toc89199671)

[Результаты выполнения программы 10](#_Toc89199672)

[Вывод по заданию 11](#_Toc89199673)

[Вывод 12](#_Toc89199674)

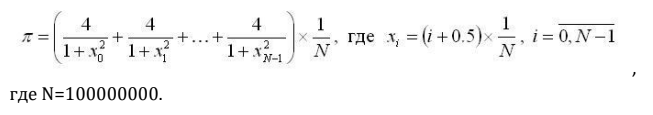
# Цель работы

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

## Реализация многопоточного приложения с использованием функций Win32 API

## Указания к выполнению

1. Создайте приложение, которое вычисляет число pi с точностью N знаков после запятой по следующей формуле:



* Используйте распределение итераций блоками (размер блока = 10 \* N студбилета) по потокам. Сначала каждый поток по очереди получает свой блок итераций, затем тот поток, который заканчивает выполнение своего блока, получает следующий свободный блок итераций. Освободившиеся потоки получают новые блоки итераций до тех пор, пока все блоки не будут исчерпаны;
* Создание потоков выполняйте с помощью функции Win32 API **CreateThread**;
* Для реализации механизма распределения блоков итераций необходимо сразу в начале программы создать необходимое количество потоков в приостановленном состоянии, для освобождения потока из приостановленного состояния используйте функцию Win32 API **ResumeThread**;
* По окончании обработки текущего блока итераций поток не должен завершаться, а должен быть, например, приостановлен с помощью функции Win32 API SuspendThread. Затем потоку должен быть предоставлен следующий свободный блок итераций, и поток должен быть освобожден, например, с помощью функции Win32 API **ResumeThread**.

2. Произведите замеры времени выполнения приложения для разного числа потоков (1, 2, 4, 8, 12, 16). По результатам измерений постройте график и определите число потоков, при котором достигается наибольшая скорость выполнения. Запротоколируйте результаты в отчет.

## Результаты выполнения программы

Написанное многопоточное консольное приложение отображает вычисление числа Pi и время этого вычисления для количества 1, 2, 4, 8 и 16 потоков. Для достоверных значений времени было выполнено 5 замеров, которые представлены на рисунках 1, 2, 3, 4 и 5 соответственно. Округление времени производилось именно до 3-х знаков после запятой, так как дальше стояли незначащие нули.

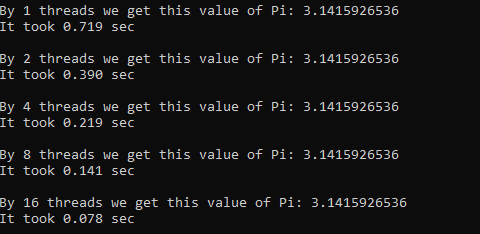
****

Рисунок 1. Замер 1

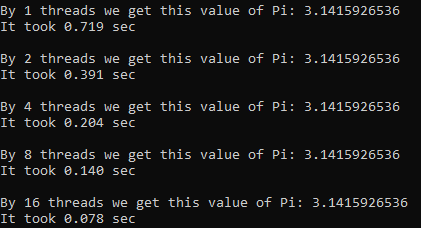
****

Рисунок 2. Замер 2

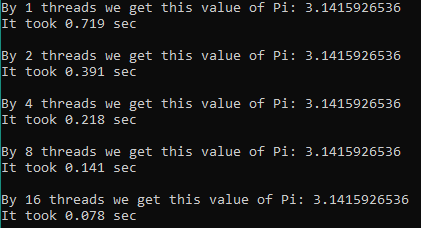
****

Рисунок 3. Замер 3

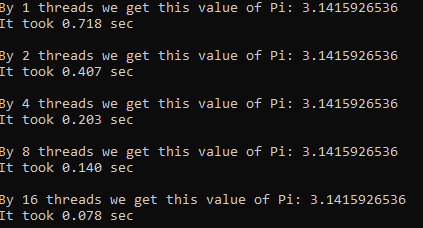
****

Рисунок 4. Замер 4

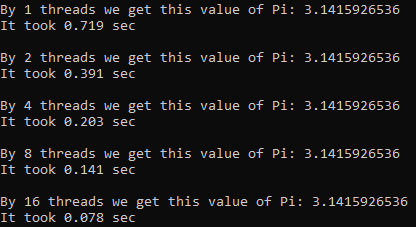
****

Рисунок 5. Замер 5

Ниже, на рисунке 6 представлен график зависимости времени вычисления от кол-ва потоков, построенный исходя из значений, полученных в ходе вышеописанных замеров. Для значения времени было взято среднее арифметическое для каждого количества потоков.

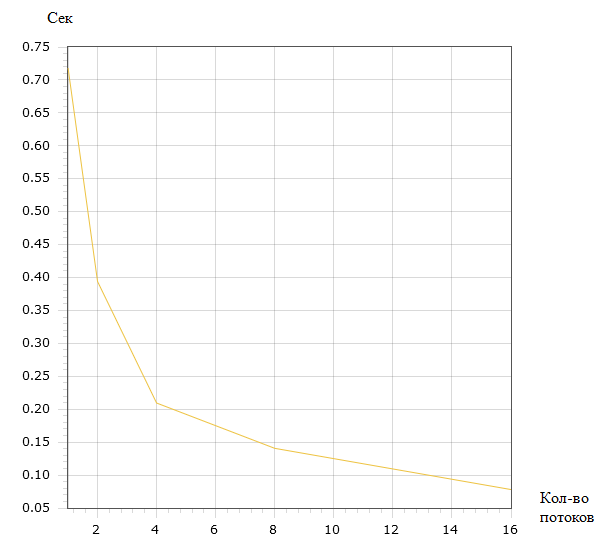


Рисунок 6. График зависимости времени вычисления от кол-ва потоков

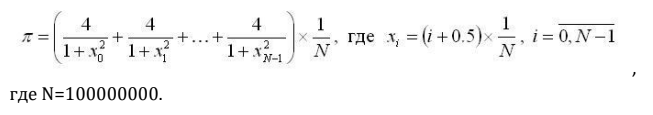
## Вывод по заданию

В ходе задания было изучено создание и управление процессами и потоками в ОС Windows. Как видно по графику, оптимальное число потоков в данном случае находится между 8 и 16. Это обусловлено тем, что замеры проводились на компьютере с 12 логическими процессорами, и увеличение количества потоков далее приведет не приведет к уменьшению времени вычисления, так как необходимо будет замещать одни потоки другими. Таким образом, можно сказать, что превышать количество логических процессоров при создании нескольких потоков не имеет смысла.

## Реализация многопоточного приложения с использованием технологии OpenMP

## Указания к выполнению

1. Создайте приложение, которое вычисляет число пи с точностью N знаков после запятой по следующей формуле:



* Распределите работу по потокам с помощью OpenMP-директивы **for**;
* Используйте динамическое планирование блоками итераций (размер блока = 10 \* Nстудбилета).

2. Произведите замеры времени выполнения приложения для разного числа потоков (1, 2, 4, 8, 12, 16). По результатам измерений постройте график и определите число потоков, при котором достигается наибольшая скорость выполнения. Запротоколируйте результаты в отчет, сравните с результатами прошлой работы.

## Результаты выполнения программы

Написанное консольное приложение отображает вычисление числа Pi и время этого вычисления для количества 1, 2, 4, 8 и 16 потоков. Вновь было выполнено 5 замеров, чтобы сравнение двух методов реализации многопоточности было справедливым. Замеры представлены на рисунках 7, 8, 9, 10 и 11 соответственно. Округление времени производилось также до 3-х знаков после запятой.

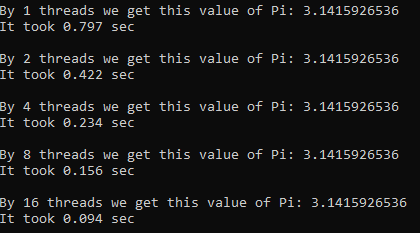


Рисунок . Замер 1

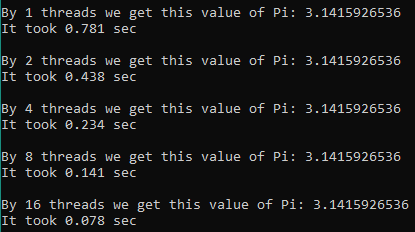


Рисунок . Замер 2

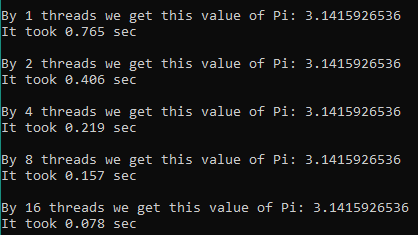


Рисунок . Замер 3

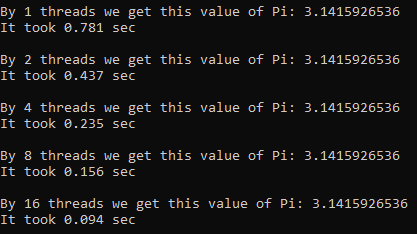


Рисунок . Замер 4

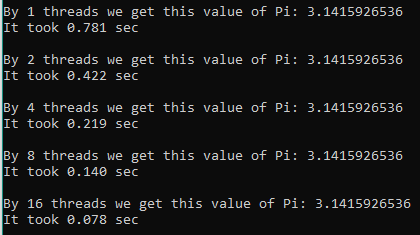


Рисунок . Замер 5

На рисунке 12 представлен график зависимости времени вычисления от кол-ва потоков. Для значения времени было взято среднее арифметическое для каждого количества потоков.

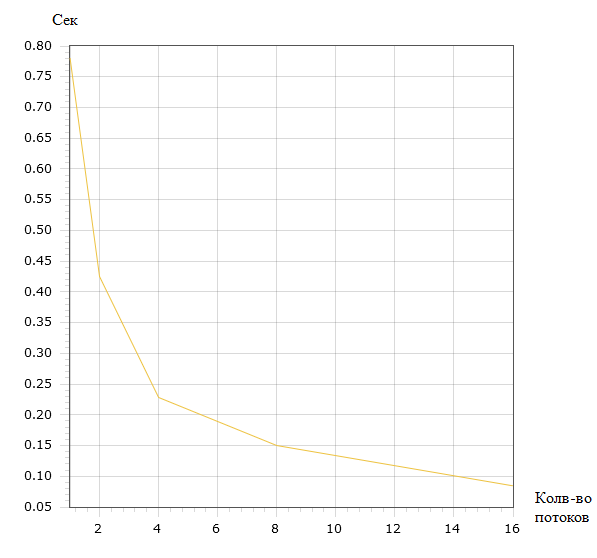


Рисунок . График зависимости времени вычисления от кол-ва потоков

## Вывод по заданию

В ходе задания была изучена работа с OpenMP-директивой, с помощью которой также производится реализация многопоточности в приложениях. Судя по графику, результаты примерно совпадают с теми, что были получены в первом задании с использованием WinAPI. Небольшие различия в значениях обусловлены тем, что на процессоре так же могли параллельно выполнятся некоторые другие задачи.

# Вывод

В ходе выполнения лабораторной работы были исследованы создание и управление процессами и потоками в ОС Windows с использованием WinAPI и технологии OpenMP, а также проводилось сравнение этих способов, в ходе которого было установлено, что результаты примерно равны для обоих подходов.