

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Дальневосточный федеральный университет»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ

ТЕХНОЛОГИЙ

Петров Сергей Дмитриевич

ОТЧЕТ

К лабораторной работе №2

На тему “Управление процессами: семафоры”

Специальность 02.03.01сцт Математика и компьютерные науки

Студент гр. \_Петров Сергей Дмитриевич\_ (подпись)

Преподаватель \_\_\_\_ КТН, Доцент Мокрицкая Е. Б.

г. Владивосток

2022

# Введение

Семафор представляет собой обрабатываемый ядром целочисленный объект, для которого определены следующие элементарные (неделимые) операции:

* Инициализация семафора, в результате которой семафору присваивается неотрицательное значение;
* Операция типа P, уменьшающая значение семафора. Если значение семафора опускается ниже нулевой отметки, выполняющий операцию процесс приостанавливает свою работу;
* Операция типа V, увеличивающая значение семафора. Если значение семафора в результате операции становится больше или равно 0, один из процессов, приостановленных во время выполнения операции P, выходит из состояния приостанова;
* Условная операция типа P, сокращенно CP (conditional P), уменьшающая значение семафора и возвращающая логическое значение "истина" в том случае, когда значение семафора остается положительным. Если в результате операции значение семафора должно стать отрицательным или нулевым, никаких действий над ним не производится и операция возвращает логическое значение "ложь".

# Реализация семафоров

Семафор – это конструкция синхронизации. Предлагает потоки с синхронизированным доступом к ограниченному количеству ресурсов.

Семафор можно рассматривать как переменную, отражающую количество существующих в настоящее время ресурсов. Например, есть несколько слотов, доступных на определенном уровне на стоянке торгового центра, который является семафором. Значение семафора не может быть меньше нуля и больше общего количества существующих ресурсов.

Семафор связан с двумя операциями – получением и освобождением.

Когда один из ресурсов синхронизируется с использованием семафора, который «захватывается» потоком, значение семафора уменьшается.

Когда один ресурс из них синхронизируется с использованием семафора, “освобождаемого” потоком, значение семафора увеличивается.

Концепция семафора создана голландским ученым-компьютерщиком по имени Эдсгер Дейкстра. Две операции семафора, а именно получение и освобождение, Дейкстрой обозначил как p и v, соответственно, где p и v – первые буквы голландских слов proberen и vehogen.

**Реализация**

Класс модуля threading используется для реализации концепции семафора в Python. Класс Semaphore состоит из конструктора и двух функций, m() и p().

Функция m() используется для уменьшения счетчика семафора в случае, если счетчик больше нуля. В противном случае он блокируется, пока счетчик не станет больше нуля.

Функция p() используется для увеличения счетчика семафора и пробуждения одного из потоков, ожидающих семафор.

Давайте рассмотрим следующий синтаксис, чтобы создать объект Semaphore.

Всякий раз, когда функция m() выполняется потоком, значение параметра «count» будет уменьшаться на единицу. Каждый раз, когда функция p() выполняется потоком, значение параметра «count» увеличивается на единицу. Этот оператор подразумевает, что всякий раз, когда мы вызываем метод m(), значение параметра «count» будет уменьшаться, тогда как при вызове метода p() значение параметра «count» будет увеличиваться.

**Код**

1. # Импортируем зависимости
2. **from** time **import** sleep
3. **import** random
4. **from** threading **import** Thread
6. # Создание класса семафора
7. **class** Semaphore():
9. # Инициализация
10. **def** \_\_init\_\_(self, count **=** 5):
11. # При неверном количестве ставится стандартное значение
12. **if**(count<**=**0):
13. self.count **=** 5
14. **else**:
15. self. count **=** count
17. # Освобождение
18. **def** m(self):
19. **while**(self. count <**=** 0):
20. ...
21. self. count **-=** 1
22. **return** self.count > 0
24. # Захват
25. **def** p(self):
26. self.count **+=** 1
28. # Создание общего семафора
29. sem **=** Semaphore(5)
31. # Создание семафора для печати
32. print\_sim **=** Semaphore(1)
34. # Функция для проверки
35. **def** show(text):
36. # Захват общего семафора
37. sem.m()
39. # Захват семафора для печати
40. print\_sim.m()
41. print("Show start", text)
43. # Освобождение семафора для печати
44. print\_sim.p()
46. sleep(5)
48. # Захват семафора для печати
49. print\_sim.m()
50. print("Show end", text )
52. # Освобождение семафора для печати
53. print\_sim.p()
55. # Освобождение общего семафора
56. sem.p()



61. # Пример работы
63. # Создание потока 1 выполняющего функцию для проверки
64. th1 **=** Thread(target **=** show, args **=** ('Thread1',))
65. # Создание потока 2 выполняющего функцию для проверки
66. th2 **=** Thread(target **=** show, args **=** ('Thread2',))
67. # Создание потока 3 выполняющего функцию для проверки
68. th3 **=** Thread(target **=** show, args **=** ('Thread3',))
69. # Создание потока 4 выполняющего функцию для проверки
70. th4 **=** Thread(target **=** show, args **=** ('Thread4',))
71. # Создание потока 5 выполняющего функцию для проверки
72. th5 **=** Thread(target **=** show, args **=** ('Thread5',))
73. # Создание потока 6 выполняющего функцию для проверки
74. th6 **=** Thread(target **=** show, args **=** ('Thread6',))
76. # Запуск потока 1
77. th1.start()
78. # Запуск потока 2
79. th2.start()
80. # Запуск потока 3
81. th3.start()
82. # Запуск потока 4
83. th4.start()
84. # Запуск потока 5
85. th5.start()
86. # Запуск потока 6
87. th6.start()

Вывод

Show start Thread1

Show start Thread2

Show start Thread3

Show start Thread4

Show start Thread5

Show end Thread5

Show end Thread4

Show end Thread3

Show end Thread1

Show start Thread6

Show end Thread2

Show end Thread6

[Finished in 10.5s]

**Заключение**

В ходе выполнения данной лабораторной работы был изучен принцип работы с семафорами.

**Список литературы**

Работа семафоров: <https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)>

Документация библиотеки threading: https://docs.python.org/3/library/threading.html