[Федеральное государственное автономное образовательное учреждение высшего образования "Российский университет транспорта"](https://www.miit.ru/depts/37) РУТ(МИИТ)

Кафедра «Управление и защита информации»

**ОТЧЁТ**

по практической работе № 1  
по дисциплине  
Операционные системы  
на тему

«Процессы и потоки»

Вариант № 3

|  |  |
| --- | --- |
| Выполнил | Соснин М.А.  Митин Г.А.  Анисимов Д.М.  Ситников Д.Ю.  Мишин В.П.  Кочетов И.И. |
| Группа | ТКИ-411 |
| Проверил | доц., к.т.н. Логинова Л.Н. |

МОСКВА – 2022 г.

Оглавление

[**Задание** 3](#_Toc98108952)

[**Листинг программы** 4](#_Toc98108953)

[**Результаты работы программы** 6](#_Toc98108954)

[**Вывод** 8](#_Toc98108955)

# **Задание**

Реализовать на языке программирования Python задачу обедающих философов. Определить сколько времени пройдет до возникновения состязательной ситуации. Разрешить проблему с помощью выбранного примитива синхронизации.

# **Листинг программы**

from tkinter import \*  
import threading  
from time import sleep, time  
from random import randint  
from tkinter import messagebox  
  
start\_time = time()  
philosophers = 5  
forks\_count = philosophers  
stop = False  
  
forks = []  
for i in range(forks\_count):  
 forks.append(threading.Lock())  
  
  
def check\_thread():  
 work = False  
 global procs  
 for proc in procs:  
 if proc.is\_alive():  
 work = True  
 return work  
  
  
def eat(left, right, i):  
 p = True  
 while p:  
 while left.locked() or right.locked():  
 sleep(1)  
 if not left.locked() and not right.locked():  
 left.acquire()  
 right.acquire()  
 try:  
 print(f"{threading.currentThread().name} {i+1} ест \n", end="")  
 window.c.itemconfigure(window.phil[i], text=f"{threading.currentThread().name}{i + 1} ест",  
 fill="#652828", justify="left")  
 sleep(randint(5, 10))  
 print(f"{threading.currentThread().name} {i+1} закончил есть \n", end="")  
 window.c.itemconfigure(window.phil[i], text=f" {threading.currentThread().name}{i + 1} закончил есть",  
 fill="#652828", justify="left")  
 finally:  
 left.release()  
 right.release()  
 sleep(1)  
 if time() - start\_time > 5 \* 60 or stop:  
 p = False  
 print(f"{threading.currentThread().name} {i+1} Процесс завершен")  
 if not stop:  
 window.c.itemconfigure(window.phil[i], text=f" {threading.currentThread().name}{i+1} завершен",  
 fill="#652828")  
  
  
class Window(Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.geometry('300x180')  
 self.title("Философы")  
 self.btn = []  
 self.phil = []  
 self.c = Canvas(width=280, height=100)  
 self.c.place(x=0, y=70)  
 for i in range(philosophers):  
 self.btn.append(Button(self, text=" ", bg="green", activebackground="green"))  
 self.btn[i].grid(row=1, column=i)  
 self.phil.append(self.c.create\_text((85, 10 + i \* 20), text=f"Философ №{i} запущен", fill="#652828"))  
 self.lbl = Label(self, text="")  
 self.lbl.place(x=2, y=40)  
 self.update()  
 self.after(1, self.updater)  
  
 def updater(self):  
 self.u = threading.Thread(target=self.upd, name="Update")  
 self.u.daemon = True  
 self.u.start()  
  
 def upd(self):  
 while check\_thread():  
 for i, fork in enumerate(forks):  
 if fork.locked():  
 self.btn[i].configure(bg='red')  
 else:  
 self.btn[i].configure(bg='green')  
 self.update()  
 sleep(0.5)  
 if threading.main\_thread().is\_alive():  
 self.lbl.configure(text="Философы наелись. Окно можно закрыть")  
  
 def on\_close(self):  
 if check\_thread():  
 if messagebox.askokcancel("Выйти", "Философы не закночили есть.\nДействительно выйти?"):  
 self.lbl.configure(text="Ожидание завершения потоков")  
 self.update()  
 global stop  
 stop = True  
 while check\_thread():  
 pass  
 self.destroy()  
 else:  
 self.destroy()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 procs = []  
 window = Window()  
 for i in range(philosophers):  
 procs.append(threading.Thread(target=eat, name="Философ №", args=(forks[i], forks[(i+1)%philosophers], i)))  
 for proc in procs:  
 proc.start()  
 window.protocol("WM\_DELETE\_WINDOW", window.on\_close)  
 window.mainloop()

# **Результаты работы программы**

Время до состязательной ситуации задаётся рандомно в диапазоне от 5 до 10 секунд.

Результаты работы программы представлены на Рисунок 1-4.

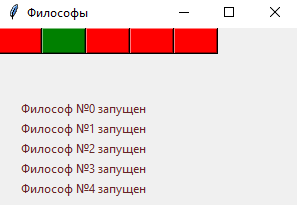


Рисунок 1 – Графическая реализация начала работы процессов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Начало работы процессов.

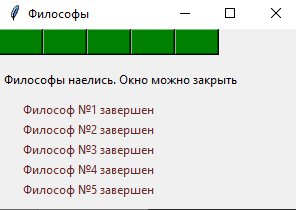


Рисунок 3 – Графическая реализация завершения работы процессов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Завершение работы процессов.

# **Вывод**

Работа может считаться успешно выполненной, т.к. реальный результат программы совпадает с ожидаемым. Были получены практические навыки работы с управлением процессами и потоками.