МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ  
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Поволжский Государственный Университет Телекоммуникаций и Информатики

Отчет по практической работе № 5

по дисциплине «Операционные системы и оболочки»

Выполнил:  
студент группы ИВТ-34У  
Горлов А.А.

Самара, 2024

Задание

Смоделировать ситуацию взаимоблокировки согласно варианту

(запрограммировать и показать данную ситуацию).

Построить граф распределения ресурсов и пояснить его. Применить

алгоритм для обнаружения взаимоблокировки при использовании одного

ресурса каждого типа.

Произвести выход из взаимоблокировки и обосновать метод выхода.

Вариант 7. Первая транзакция - T1. Вторая транзакция - T2. Если эти

транзакции стартуют одновременно, то произойдет взаимоблокировка по

причине очевидного нарушения порядка доступа. T1 сначала обращается к

записи X = 1, а затем к записи X = 3. Т2 же, наоборот, сначала обращается к

записи X = 3, а затем к X = 1

При одновременном старте Т1 захватывает запись X = 1, в это время

Т2 успевает захватить запись X = 3. Затем T1 хочет захватить запись X = 3,

но она уже захвачена T2, поэтому T1 ожидает T2 на блокировке, и в граф

добавляется ребро T1->T2. Примерно в это же время T2 хочет захватить

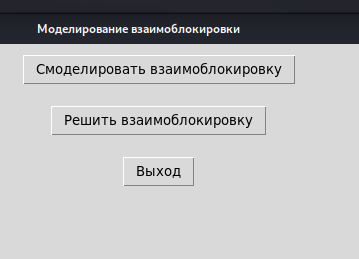
запись X = 1, которая также уже захвачена T1. В графе ожидания появляется

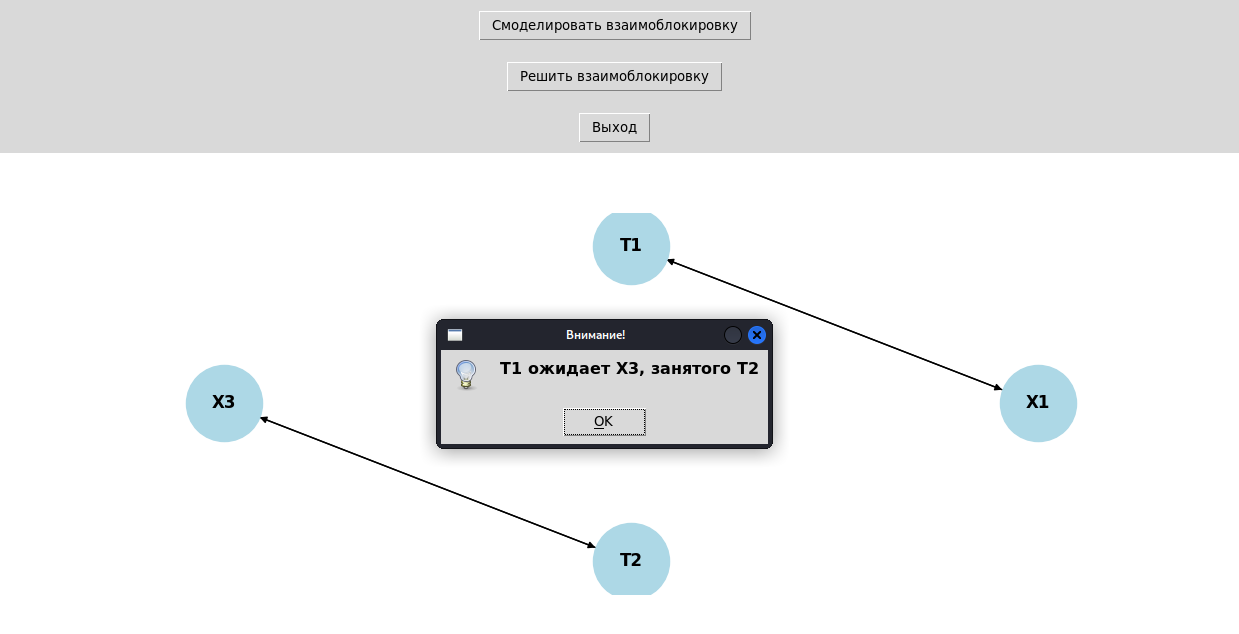
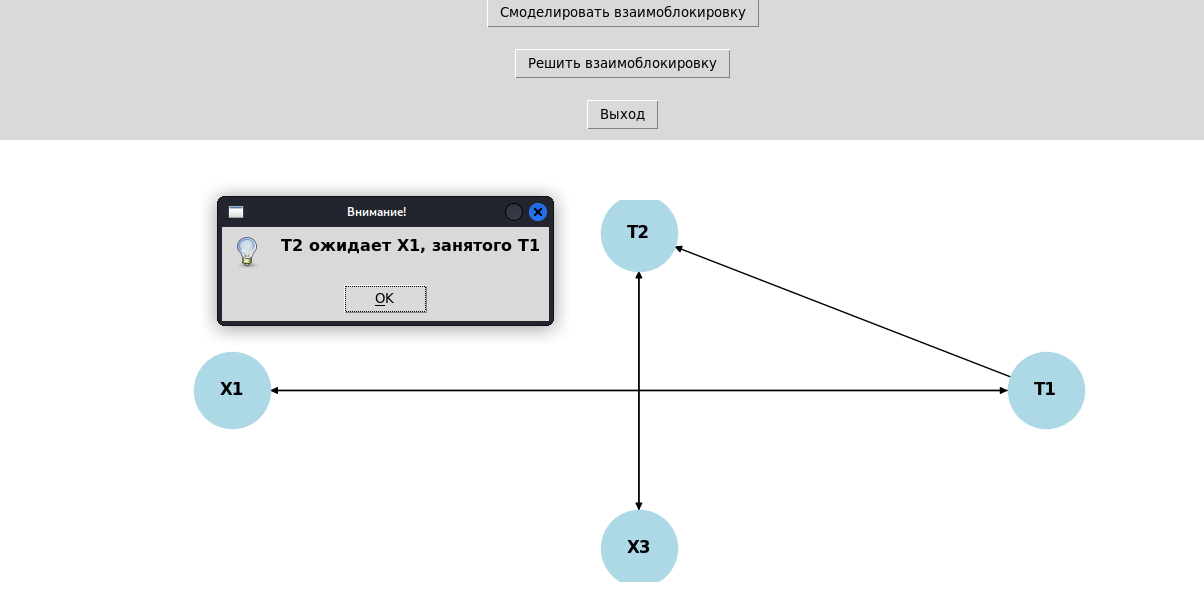
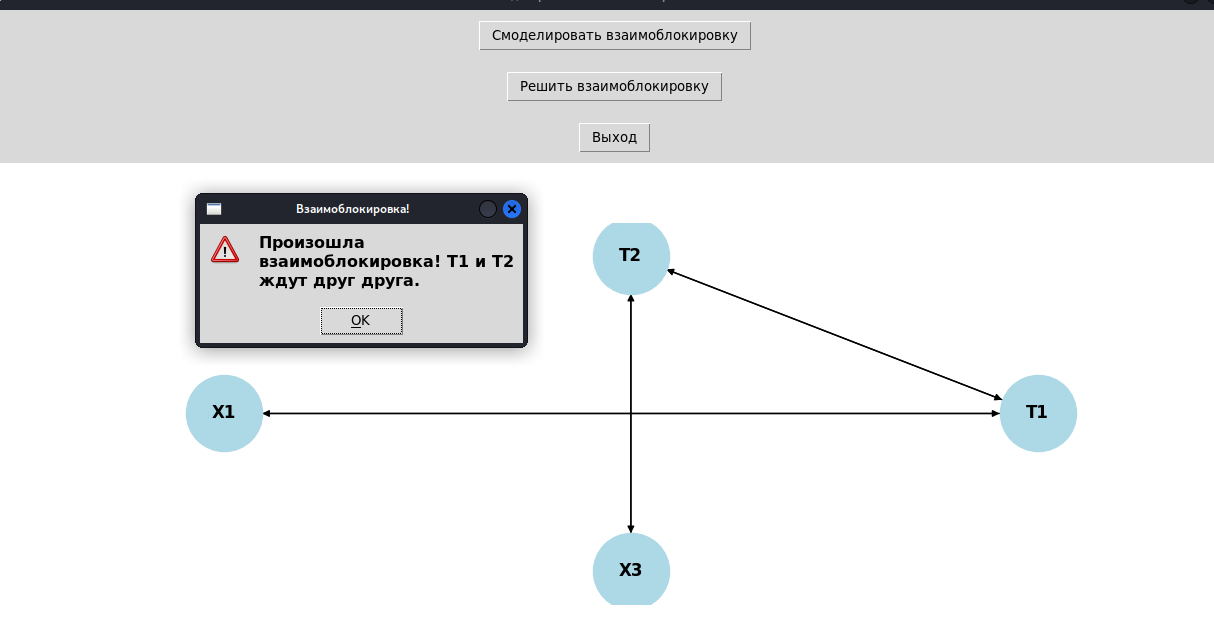
второе ребро T2->T1 и он становится цикличным. Ну а поскольку подобная

ситуация без грубого вмешательства неразрешима, то одна из транзакций

будет отменена, другая же, пользуясь тем, что блокировка исчезла, спокойно

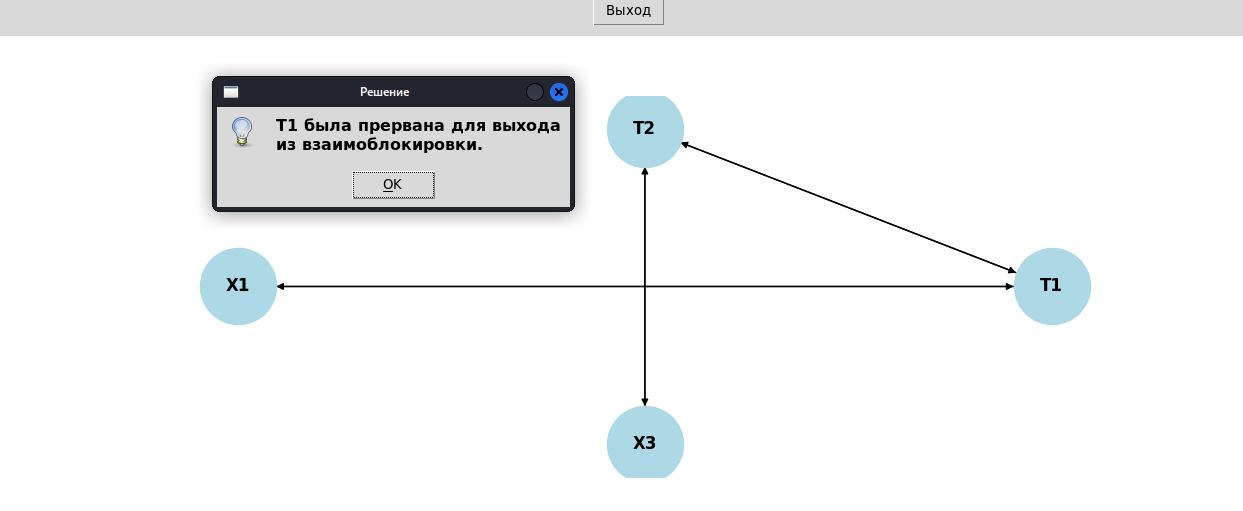
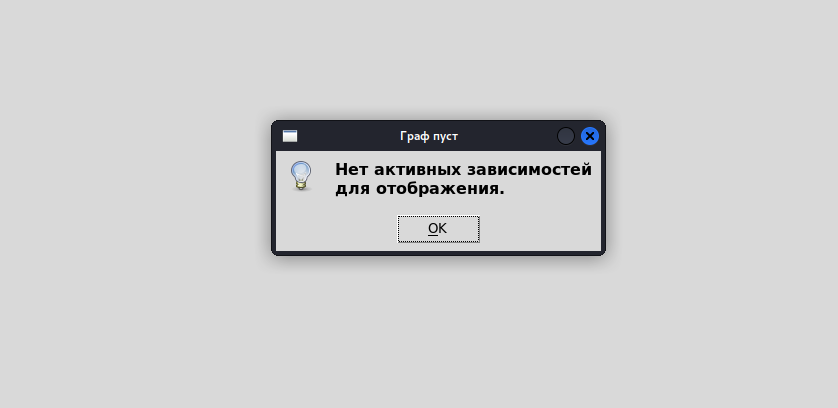
завершит свою работу

Окно программы  


Выполнение симуляции  
  

На графике видно, что Т2 занял Х3, а Т1 Х1, а при попытке обращения к другим ресурсам, они вынуждены ждать друг друга

Попытка разрешения

По скольку мы прервали Т1, то Т2 смогла спокойно выполниться и все задачи успешно завершились

main.py  
import tkinter as tk

from tkinter import messagebox

from resource import Resource

from transaction import Transaction

from graph\_drawer import draw\_graph

# Функция для захвата ресурсов транзакциями

def acquire\_resource(transaction, resource, t1, t2, x1, x3, window):

    if resource.locked\_by is None:

        resource.locked\_by = transaction

    else:

        transaction.waiting\_for = resource.locked\_by

        messagebox.showinfo("Внимание!", f"{transaction.name} ожидает {resource.name}, занятого {resource.locked\_by.name}")

    draw\_graph(t1, t2, x1, x3, window)

# Функция моделирования взаимоблокировки

def simulate\_deadlock(window):

    for r in [x1, x3]:

        r.locked\_by = None

    t1.waiting\_for = None

    t2.waiting\_for = None

    # T1 захватывает X1

    acquire\_resource(t1, x1, t1, t2, x1, x3, window)

    # T2 захватывает X3

    acquire\_resource(t2, x3, t1, t2, x1, x3, window)

    # T1 пытается захватить X3

    acquire\_resource(t1, x3, t1, t2, x1, x3, window)

    # T2 пытается захватить X1

    acquire\_resource(t2, x1, t1, t2, x1, x3, window)

    # Проверка на взаимоблокировку

    if t1.waiting\_for and t2.waiting\_for:

        if t1.waiting\_for == t2 and t2.waiting\_for == t1:

            messagebox.showwarning("Взаимоблокировка!", "Произошла взаимоблокировка! T1 и T2 ждут друг друга.")

def resolve\_deadlock(window):

    # Прерывание T1 для выхода из взаимоблокировки

    t1.waiting\_for = None  # T1 больше ни за что не отвечает

    t2.waiting\_for = None

    x1.locked\_by = None

    x3.locked\_by = None

    messagebox.showinfo("Решение", "T1 была прервана для выхода из взаимоблокировки.")

    draw\_graph(None, t2, x1, x3, window)

# Создание интерфейса

window = tk.Tk()

window.title("Моделирование взаимоблокировки")

# Ресурсы

x1 = Resource("X1")

x3 = Resource("X3")

# Транзакции

t1 = Transaction("T1", [x1, x3])

t2 = Transaction("T2", [x3, x1])

simulate\_button = tk.Button(window, text="Смоделировать взаимоблокировку", command=lambda: simulate\_deadlock(window))

simulate\_button.pack(pady=10)

resolve\_button = tk.Button(window, text="Решить взаимоблокировку", command=lambda: resolve\_deadlock(window))

resolve\_button.pack(pady=10)

exit\_button = tk.Button(window, text="Выход", command=window.quit)

exit\_button.pack(pady=10)

window.mainloop()

graph\_drawer.py  
import networkx as nx

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from tkinter import messagebox

current\_canvas = None

def draw\_graph(t1, t2, x1, x3, window):

    global current\_canvas

    # Удаляем предыдущий график

    if current\_canvas:

        current\_canvas.get\_tk\_widget().destroy()

    G = nx.DiGraph()

    # Добавляем зависимости транзакций и ресурсов

    if t1 and hasattr(t1, 'waiting\_for') and t1.waiting\_for:

        G.add\_edge(t1.name, t1.waiting\_for.name)

    if t2 and hasattr(t2, 'waiting\_for') and t2.waiting\_for:

        G.add\_edge(t2.name, t2.waiting\_for.name)

    # Если ресурсы заблокированы транзакциями, добавляем рёбра от ресурсов к транзакциям

    if x1 and hasattr(x1, 'locked\_by') and x1.locked\_by:

        G.add\_edge(x1.name, x1.locked\_by.name)

    if x3 and hasattr(x3, 'locked\_by') and x3.locked\_by:

        G.add\_edge(x3.name, x3.locked\_by.name)

    # Если ресурсы ожидаются транзакциями

    if t1 and hasattr(t1, 'resources') and t1.resources:

        for res in t1.resources:

            if res.locked\_by == t1:

                G.add\_edge(t1.name, res.name)

    if t2 and hasattr(t2, 'resources') and t2.resources:

        for res in t2.resources:

            if res.locked\_by == t2:

                G.add\_edge(t2.name, res.name)

    if not G.nodes:

        messagebox.showinfo("Граф пуст", "Нет активных зависимостей для отображения.")

        return

    pos = nx.circular\_layout(G)

    fig, ax = plt.subplots(figsize=(4, 4))

    nx.draw(G, pos, with\_labels=True, ax=ax, node\_color='lightblue', node\_size=3000, font\_size=12, font\_weight='bold')

    current\_canvas = FigureCanvasTkAgg(fig, master=window)

    current\_canvas.get\_tk\_widget().pack(side="top", fill="both", expand=True)

    current\_canvas.draw()

    plt.close(fig)