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 № 6

по дисциплине «Операционные системы и оболочки»

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

Самара, 2024

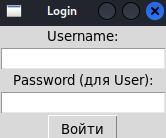
Задание

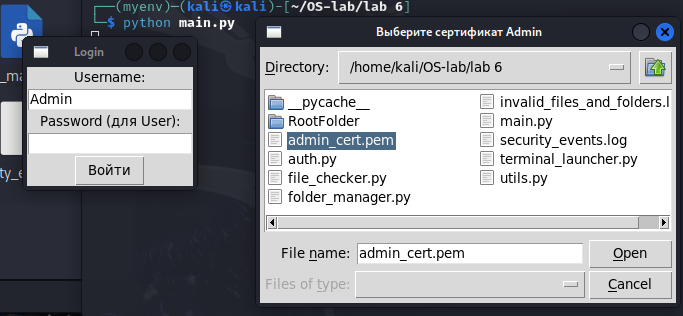
Для выполнения работы потребуется проект из раздела №3. Необходимо модифицировать программу, применив идентификацию, аутентификацию и авторизацию. Создать 2 пользователя: Admin – после входа в программу имеет полный функционал и нет ограничений работы с папками/файлами, и User – После входа имеет программу с урезанным функционалом (урезанное меню, отсутствие объектов на форме и тп.) и есть свои права для работы в приложение (выбираются по варианту). Осуществить вхождение двух пользователей через разные аутенфикаций: User - Базовая аутентификация (логин и пароль), Admin - Аутентификация с предъявлением цифрового сертификата. Также программа должна реализовывать простой механизм логирования событий безопасности в ОС Linux. Дополнительно позволить запуск из приложения собственного терминала (аналог команд Linux). Добавить по 10 команд управления

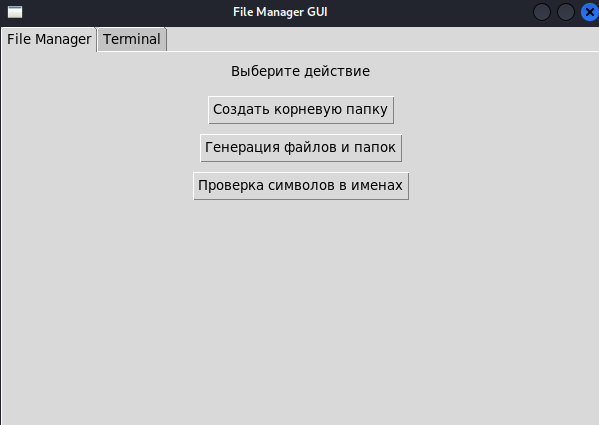
Вариант 7. Доступ к файлу для записи.

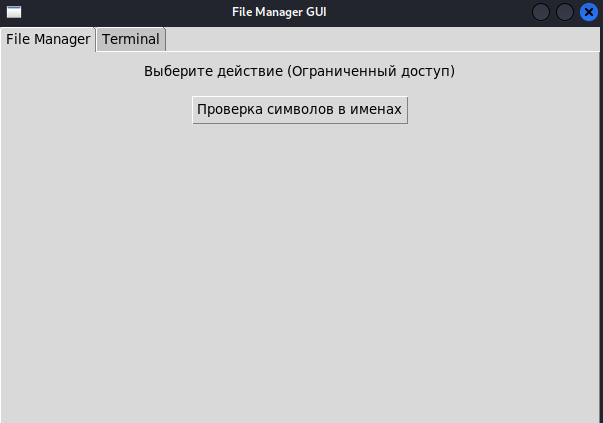
Работа прогораммы

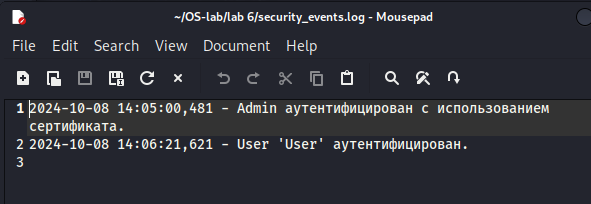
Окно входа



Выбор сертификата для Админа  


Полный доступ к программе, аналогичный в лаболаторной 3  
  
вход под юзером

Ограниченный функционал  


Юзеру доступно записывать резуьлтат проверки в лог файл  
лог файл входов   


main.py

import tkinter as tk

from tkinter import ttk, messagebox

from folder\_manager import create\_root\_folder, generate\_random\_files\_and\_folders

from file\_checker import check\_files\_and\_folders\_in\_directory

from terminal\_launcher import run\_terminal\_command

from utils import select\_directory

from auth import authenticate\_user, authenticate\_admin, is\_admin, is\_user

class Application(tk.Tk):

    def \_\_init\_\_(self, role):

        super().\_\_init\_\_()

        self.role = role

        self.title("File Manager GUI")

        self.geometry("600x400")

        self.create\_widgets()

    def create\_widgets(self):

        tab\_control = ttk.Notebook(self)

        # Создаём вкладки

        tab\_file\_manager = ttk.Frame(tab\_control)

        tab\_control.add(tab\_file\_manager, text='File Manager')

        tab\_terminal = ttk.Frame(tab\_control)

        tab\_control.add(tab\_terminal, text='Terminal')

        # Настраиваем доступность вкладок в зависимости от роли

        if is\_admin(self.role):

            self.create\_file\_manager\_tab(tab\_file\_manager)

        elif is\_user(self.role):

            self.create\_user\_limited\_tab(tab\_file\_manager)

        self.create\_terminal\_tab(tab\_terminal)

        tab\_control.pack(expand=1, fill='both')

    def create\_file\_manager\_tab(self, tab):

        ttk.Label(tab, text="Выберите действие").pack(pady=10)

        btn\_create\_folder = ttk.Button(tab, text="Создать корневую папку", command=self.create\_folder)

        btn\_create\_folder.pack(pady=5)

        btn\_generate\_files = ttk.Button(tab, text="Генерация файлов и папок", command=self.generate\_files)

        btn\_generate\_files.pack(pady=5)

        btn\_check\_files = ttk.Button(tab, text="Проверка символов в именах", command=self.check\_files)

        btn\_check\_files.pack(pady=5)

    def create\_user\_limited\_tab(self, tab):

        ttk.Label(tab, text="Выберите действие (Ограниченный доступ)").pack(pady=10)

        # Ограниченный функционал для User

        btn\_check\_files = ttk.Button(tab, text="Проверка символов в именах", command=self.check\_files)

        btn\_check\_files.pack(pady=5)

    def create\_terminal\_tab(self, tab):

        ttk.Label(tab, text="Управление терминалом").pack(pady=10)

        self.terminal\_output = tk.Text(tab, height=15, width=70, state='disabled')

        self.terminal\_output.pack(pady=5)

        self.command\_entry = ttk.Entry(tab, width=50)

        self.command\_entry.pack(side=tk.LEFT, padx=5, pady=5)

        btn\_run\_command = ttk.Button(tab, text="Выполнить", command=self.run\_terminal\_command)

        btn\_run\_command.pack(side=tk.LEFT, padx=5, pady=5)

        self.command\_entry.bind('<Return>', lambda event: self.run\_terminal\_command())

    def create\_folder(self):

        directory = select\_directory()

        if directory:

            create\_root\_folder(directory)

    def generate\_files(self):

        directory = select\_directory()

        if directory:

            generate\_random\_files\_and\_folders(directory)

    def check\_files(self):

        directory = select\_directory()

        if directory:

            check\_files\_and\_folders\_in\_directory(directory)

    def run\_terminal\_command(self):

        command = self.command\_entry.get()

        if command:

            if command.strip() == 'clear':

                self.clear\_terminal()

            else:

                output = run\_terminal\_command(command)

                self.display\_terminal\_output(output)

            self.command\_entry.delete(0, tk.END)

    def display\_terminal\_output(self, output):

        self.terminal\_output.config(state='normal')

        self.terminal\_output.insert(tk.END, f"$ {output}\n")

        self.terminal\_output.config(state='disabled')

        self.terminal\_output.see(tk.END)

    def clear\_terminal(self):

        self.terminal\_output.config(state='normal')

        self.terminal\_output.delete(1.0, tk.END)

        self.terminal\_output.config(state='disabled')

# Логика аутентификации пользователя при запуске программы

def login():

    login\_window = tk.Tk()

    login\_window.title("Login")

    tk.Label(login\_window, text="Username:").pack()

    username\_entry = tk.Entry(login\_window)

    username\_entry.pack()

    tk.Label(login\_window, text="Password (для User):").pack()

    password\_entry = tk.Entry(login\_window, show="\*")

    password\_entry.pack()

    def on\_login():

        username = username\_entry.get()

        password = password\_entry.get()

        if username == "Admin":

            # Попытка аутентификации Admin с сертификатом

            cert\_path = tk.filedialog.askopenfilename(title="Выберите сертификат Admin")

            if authenticate\_admin(cert\_path):

                login\_window.destroy()

                app = Application("Admin")

                app.mainloop()

            else:

                messagebox.showerror("Ошибка", "Неверный сертификат.")

        elif username == "User":

            # Базовая аутентификация для User

            if authenticate\_user(username, password):

                login\_window.destroy()

                app = Application("User")

                app.mainloop()

            else:

                messagebox.showerror("Ошибка", "Неверный логин или пароль.")

    btn\_login = tk.Button(login\_window, text="Войти", command=on\_login)

    btn\_login.pack()

    login\_window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    login()

auth.py

import os

import logging

from tkinter import messagebox

# Создание отдельного логгера для этого модуля

auth\_logger = logging.getLogger("auth\_logger")

auth\_logger.setLevel(logging.INFO)

auth\_handler = logging.FileHandler("security\_events.log")

auth\_formatter = logging.Formatter('%(asctime)s - %(message)s')

auth\_handler.setFormatter(auth\_formatter)

auth\_logger.addHandler(auth\_handler)

USER\_CREDENTIALS = {"User": "user\_password"}

CERTIFICATE\_NAME = "admin\_cert.pem"

# Аутентификация для User

def authenticate\_user(username, password):

    if USER\_CREDENTIALS.get(username) == password:

        auth\_logger.info(f"User '{username}' аутентифицирован.")

        return True

    auth\_logger.warning(f"Неудачная попытка входа для User '{username}'.")

    return False

# Аутентификация для Admin

def authenticate\_admin(cert\_path):

    cert\_file\_name = os.path.basename(cert\_path)

    if cert\_file\_name == CERTIFICATE\_NAME:

        auth\_logger.info("Admin аутентифицирован с использованием сертификата.")

        return True

    auth\_logger.warning("Неудачная попытка аутентификации Admin.")

    return False

# Авторизация пользователя

def is\_admin(role):

    return role == "Admin"

def is\_user(role):

    return role == "User"