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 № 3

по дисциплине «Операционные системы и оболочки»

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

Самара, 2024

Задание

Разработать программное обеспечение, которое делает проверку по

заданному условию, в создаваемой корневой папке. При запуске ПО,

пользователь выбирает физический диск, где будет создана данная папка, с

которой предстоит работать. Проверка условия может происходить по

нажатию кнопки или ввода команды, также предусмотреть отключение

данной проверки. После вывести уведомление о результатах проверки.

Необходимо подготовить все необходимые файлы/папки для демонстрации

проверки условия.

Дополнительно позволить запуск из приложения собственного

терминала (аналог команд Linux). Добавить по 10 команд для работы с

файловой системой ОС

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

class Application(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("File Manager GUI")

        self.geometry("600x400")

        self.create\_widgets()

    def create\_widgets(self):

        tab\_control = ttk.Notebook(self)

        # Вкладка для управления файлами и папками

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

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

        # Вкладка для терминала

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

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

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

        # Компоненты для вкладки File Manager

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

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

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

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

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

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

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

        # Компоненты для вкладки Terminal

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

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

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

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

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

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

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

    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')

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

    app = Application()

    app.mainloop()

file\_checker.py

import os

import re

import logging

from tkinter import messagebox

# Настройка логирования

logging.basicConfig(filename="invalid\_files\_and\_folders.log", level=logging.INFO,

                    format='%(asctime)s - %(message)s')

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

    invalid\_files\_and\_folders = []

    total\_files\_and\_folders = 0  # Считаем общее количество файлов и папок

    valid\_pattern = re.compile(r'^[A-Z]+$')

    # Обходим директорию и проверяем имена файлов и папок

    for root, dirs, files in os.walk(directory):

        for dir\_name in dirs:

            total\_files\_and\_folders += 1

            if not valid\_pattern.match(dir\_name):

                invalid\_files\_and\_folders.append(os.path.join(root, dir\_name))

        for file\_name in files:

            total\_files\_and\_folders += 1

            if not valid\_pattern.match(file\_name.split('.')[0]):  # Проверяем имя без расширения

                invalid\_files\_and\_folders.append(os.path.join(root, file\_name))

    # Записываем в лог-файл пути недопустимых файлов и папок

    if invalid\_files\_and\_folders:

        logging.info("Найдены файлы и папки с недопустимыми символами:")

        for item in invalid\_files\_and\_folders:

            logging.info(item)

    # Формируем результат для отображения в интерфейсе

    invalid\_count = len(invalid\_files\_and\_folders)

    valid\_count = total\_files\_and\_folders - invalid\_count

    result\_message = (

        f"Проверено объектов: {total\_files\_and\_folders}.\n"

        f"Файлов и папок с недопустимыми символами: {invalid\_count}.\n"

        f"Корректных файлов и папок: {valid\_count}.\n"

        f"Подробности записаны в лог файл."

    )

    messagebox.showinfo("Результат проверки", result\_message)

folder\_manager.py

import os

import random

import string

from tkinter import messagebox

def create\_root\_folder(base\_path):

    root\_folder = "RootFolder"

    root\_path = os.path.join(base\_path, root\_folder)

    try:

        os.makedirs(root\_path, exist\_ok=True)

        messagebox.showinfo("Успех", f"Корневая папка создана: {root\_path}")

        return root\_path

    except Exception as e:

        messagebox.showerror("Ошибка", f"Ошибка при создании корневой папки: {e}")

        return None

def generate\_random\_name(correct=True):

    if correct:

        return ''.join(random.choices(string.ascii\_uppercase, k=5))

    else:

        return ''.join(random.choices(string.ascii\_letters + string.digits, k=8))

def generate\_random\_files\_and\_folders(base\_path, level=0, max\_level=2):

    if level > max\_level:

        return

    try:

        for i in range(random.randint(2, 4)):

            if random.choice([True, False]):

                folder\_name = generate\_random\_name(correct=random.choice([True, False]))

                folder\_path = os.path.join(base\_path, folder\_name)

                os.makedirs(folder\_path, exist\_ok=True)

                generate\_random\_files\_and\_folders(folder\_path, level + 1, max\_level)

            else:

                file\_name = generate\_random\_name(correct=random.choice([True, False])) + ".txt"

                file\_path = os.path.join(base\_path, file\_name)

                with open(file\_path, 'w') as f:

                    f.write("Тестовый файл")

    except Exception as e:

        messagebox.showerror("Ошибка", f"Ошибка при генерации файлов и папок: {e}")

terminal\_launcher.py

import subprocess

import os

import platform

from tkinter import messagebox

def run\_terminal\_command(command):

    command\_parts = command.strip().split()

    if not command\_parts:

        return "Пожалуйста, введите команду."

    cmd = command\_parts[0]

    if cmd == 'help':

        return (

            "Доступные команды:\n"

            "1. help — показывает доступные команды.\n"

            "2. clear — очищает терминал.\n"

            "Работа с файлами:\n"

            "3. ls — показывает список файлов в текущей директории.\n"

            "4. pwd — показывает текущую директорию.\n"

            "5. mkdir <dirname> — создает новую директорию.\n"

            "6. rmdir <dirname> — удаляет пустую директорию.\n"

            "7. touch <filename> — создает новый файл.\n"

            "8. rm <filename> — удаляет файл.\n"

            "9. mv <src> <dst> — перемещает или переименовывает файл или папку.\n"

            "10. cp <src> <dst> — копирует файл или папку.\n"

            "11. cat <filename> — показывает содержимое файла.\n"

            "12. chmod <mode> <filename> — изменяет права доступа к файлу.\n"

            "13. chown <user>:<group> <filename> — изменяет владельца файла.\n"

        )

    elif cmd == 'clear':

        return ""

    elif cmd == 'ls':

        try:

            output = subprocess.check\_output(['ls'], universal\_newlines=True)

            return output

        except Exception as e:

            return f"Ошибка при выполнении команды ls: {e}"

    elif cmd == 'pwd':

        try:

            output = subprocess.check\_output(['pwd'], universal\_newlines=True)

            return output

        except Exception as e:

            return f"Ошибка при выполнении команды pwd: {e}"

    elif cmd == 'mkdir' and len(command\_parts) > 1:

        dirname = command\_parts[1]

        try:

            os.mkdir(dirname)

            return f"Директория '{dirname}' создана."

        except Exception as e:

            return f"Ошибка при создании директории '{dirname}': {e}"

    elif cmd == 'rmdir' and len(command\_parts) > 1:

        dirname = command\_parts[1]

        try:

            os.rmdir(dirname)

            return f"Директория '{dirname}' удалена."

        except Exception as e:

            return f"Ошибка при удалении директории '{dirname}': {e}"

    elif cmd == 'touch' and len(command\_parts) > 1:

        filename = command\_parts[1]

        try:

            open(filename, 'a').close()

            return f"Файл '{filename}' создан."

        except Exception as e:

            return f"Ошибка при создании файла '{filename}': {e}"

    elif cmd == 'rm' and len(command\_parts) > 1:

        filename = command\_parts[1]

        try:

            os.remove(filename)

            return f"Файл '{filename}' удален."

        except Exception as e:

            return f"Ошибка при удалении файла '{filename}': {e}"

    elif cmd == 'mv' and len(command\_parts) > 2:

        src = command\_parts[1]

        dst = command\_parts[2]

        try:

            os.rename(src, dst)

            return f"'{src}' перемещен или переименован в '{dst}'."

        except Exception as e:

            return f"Ошибка при перемещении или переименовании '{src}': {e}"

    elif cmd == 'cp' and len(command\_parts) > 2:

        src = command\_parts[1]

        dst = command\_parts[2]

        try:

            subprocess.check\_output(['cp', src, dst], universal\_newlines=True)

            return f"'{src}' скопирован в '{dst}'."

        except Exception as e:

            return f"Ошибка при копировании '{src}': {e}"

    elif cmd == 'cat' and len(command\_parts) > 1:

        filename = command\_parts[1]

        try:

            with open(filename, 'r') as file:

                content = file.read()

            return content

        except Exception as e:

            return f"Ошибка при чтении файла '{filename}': {e}"

    elif cmd == 'chmod' and len(command\_parts) > 2:

        mode = command\_parts[1]

        filename = command\_parts[2]

        try:

            os.chmod(filename, int(mode, 8))

            return f"Права доступа файла '{filename}' изменены на {mode}."

        except Exception as e:

            return f"Ошибка при изменении прав доступа к файлу '{filename}': {e}"

    elif cmd == 'chown' and len(command\_parts) > 2:

        user\_group = command\_parts[1]

        filename = command\_parts[2]

        try:

            subprocess.check\_output(['chown', user\_group, filename], universal\_newlines=True)

            return f"Владелец файла '{filename}' изменен на {user\_group}."

        except Exception as e:

            return f"Ошибка при изменении владельца файла '{filename}': {e}"

    else:

        return f"Неизвестная команда: {command}. Введите 'help' для списка доступных команд."

utils.py

from tkinter import filedialog

def select\_directory():

    directory = filedialog.askdirectory()

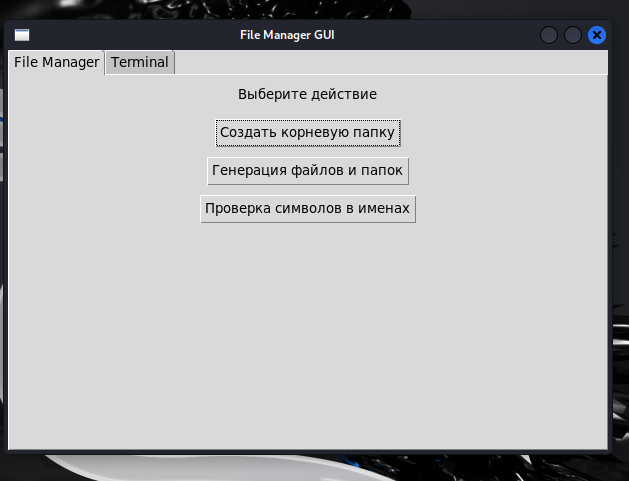
    if not directory:

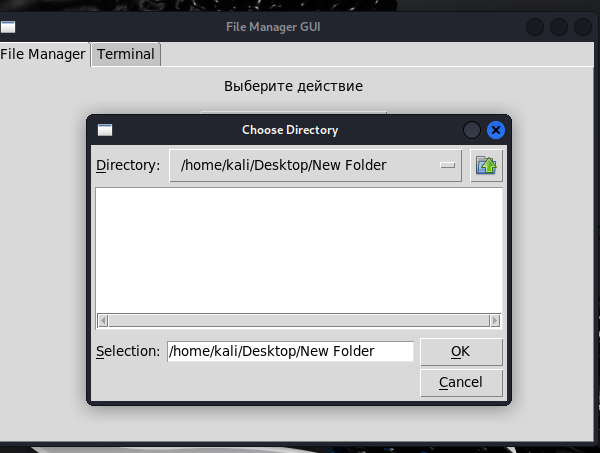
        return None

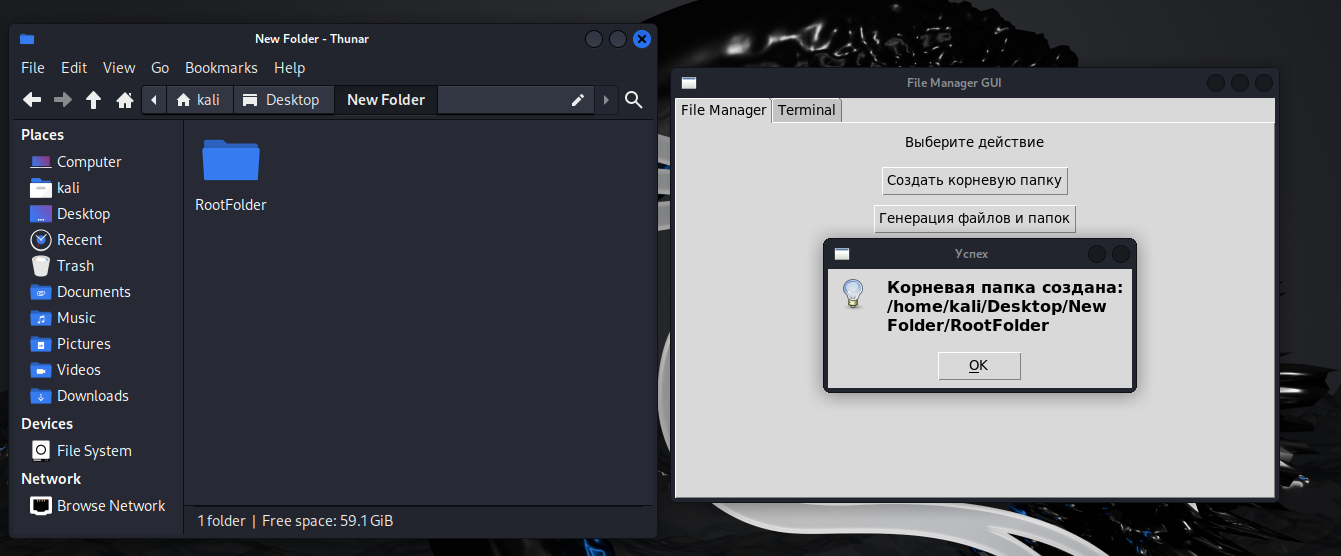
    return directory

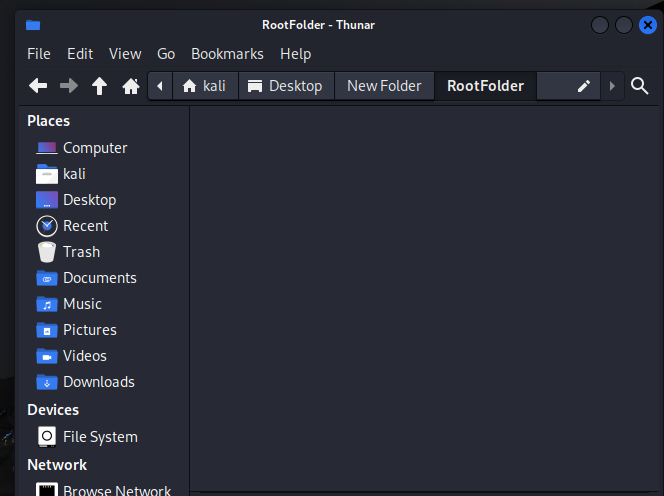
Описание работы

Окно программы

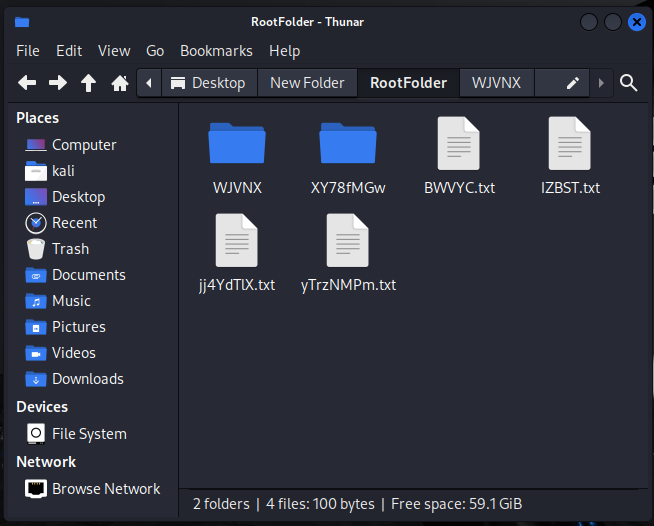


Выбираем куда создать корневую папку  


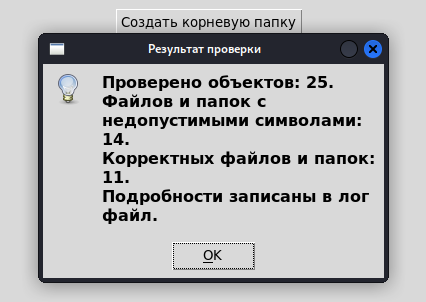




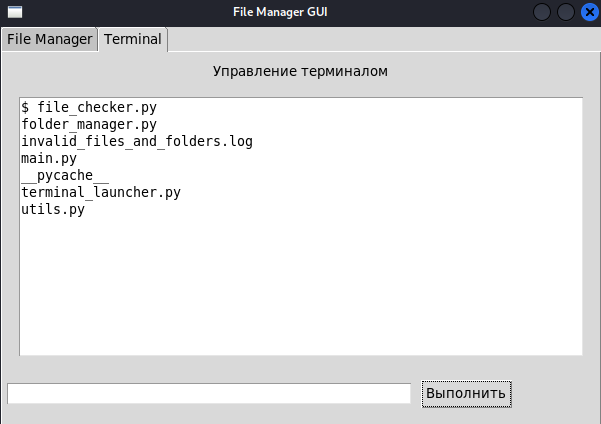
Для тестовых данных была написана функция, которая создает внутри корня, папки и файлы с различным названием

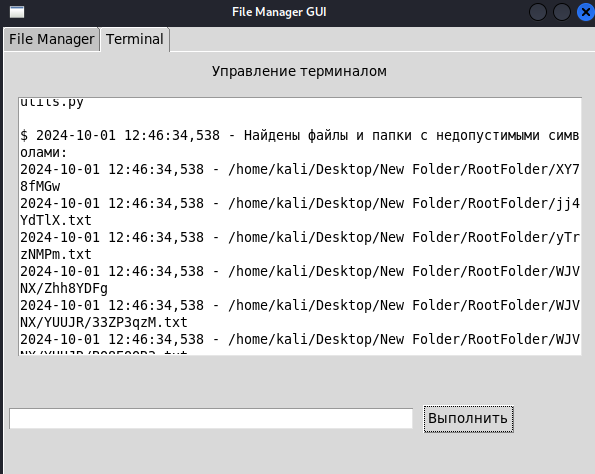


Проверка файлов



Просмотри лог в терминале



Прочтем файл invalid\_files\_and\_folders.log  


Все команды

