**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Системное программирование в Linux»**

**на тему**

**«Создание демона для регулярного резервного копирования данных»**

**Студент гр. 23Б15-пу**

**Овчинников С.А.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc179756233)
2. [Описание задачи (формализация задачи) 4](#_Toc179756234)
3. [Теоретическая часть 6](#_Toc179756235)
4. [Основные шаги программы 8](#_Toc179756236)
5. [Описание программы 10](#_Toc179756237)
6. [Рекомендации пользователя 11](#_Toc179756238)
7. [Рекомендации программиста 13](#_Toc179756239)
8. [Контрольный пример 14](#_Toc179756240)
9. [Вывод 15](#_Toc179756241)

# Цель работы

Цель данной работы заключается в разработке системного демона, предназначенного для автоматического и периодического выполнения резервного копирования данных из одного каталога в другой. Демон будет работать в фоновом режиме, обеспечивая надежное и эффективное копирование данных в соответствии с заданными пользователем настройками.

# Описание задачи (формализация задачи)

Есть несколько основных компонентов и этапов задачи:

1. **Настройка конфигурации**  
Разработка конфигурационного файла в формате INI или JSON, который позволит пользователю задать основные параметры:

* **Исходный каталог**: директория, откуда будет производиться копирование данных.
* **Каталог для резервного копирования**: место, куда сохраняются копии файлов.
* **Частота выполнения**: интервал времени для автоматизации процесса (например, ежечасно, ежедневно).
* **Дополнительные параметры**: фильтрация файлов по типу или размеру, настройка уровня логирования.

2. **Создание демона**  
Разработка фоновой программы с использованием модуля Python Daemon для выполнения резервного копирования:

* **Чтение конфигурации**: загрузка параметров из файла при запуске демона.
* **Фоновая работа**: организация непрерывного цикла для контроля частоты выполнения задач.
* **Резервное копирование**: реализация процесса копирования данных, включая добавление временных меток к файлам для идентификации версий.
* **Журналирование**: запись логов операций в системный журнал для отслеживания событий и диагностики.

3. **Интеграция с системой**

* Настройка демона для автоматического запуска при старте системы с использованием системы инициализации **systemd**.
* Обеспечение стабильной работы демона в соответствии с заданными параметрами.

4. **Управление и мониторинг**  
Разработка интерфейса командной строки для взаимодействия с демоном, позволяющего:

* Управлять процессом: останавливать, перезапускать, изменять параметры работы.
* Просматривать статус выполнения операций и результаты последних задач.

5. **Тестирование и оптимизация**

* Проведение тестирования в различных сценариях для проверки корректности работы и надежности.
* Оптимизация производительности для снижения нагрузки на систему.

6. **Безопасность**

* Реализация ограниченного доступа к конфигурационным файлам и каталогам резервного копирования.
* Защита данных от несанкционированного доступа или изменений, включая использование прав доступа и шифрования.

# Теоретическая часть

#### 1. **Демон и его назначение**

Демон представляет собой программу, функционирующую в фоновом режиме и выполняющую задачи без непосредственного взаимодействия с пользователем. Он автоматически запускается при старте системы и работает до её выключения. Примеры задач демонов:

* Управление процессами.
* Мониторинг сетевых подключений.
* Организация резервного копирования данных для их сохранности.

#### 2. **Системы инициализации**

В современных Linux-дистрибутивах управление демонами осуществляется через систему **systemd**.

* **systemd** позволяет централизованно управлять службами (запускать, останавливать, перезапускать).
* Каждый демон представляется в виде отдельного юнита, который содержит инструкции для управления.

#### 3. **Конфигурационные файлы**

Демон настраивается через конфигурационные файлы, которые задают параметры его работы.

* Форматы конфигурационных файлов:
  + **INI**: простой и удобный для небольших настроек.
  + **JSON**: используется для структурированных и сложных данных.
  + **YAML**: читабельный и поддерживающий вложенные структуры.
* Параметры могут включать: исходный каталог, директорию для резервных копий, частоту выполнения и фильтры для файлов.

#### 4. **Файловая система и резервное копирование**

Резервное копирование — это создание дубликатов данных для защиты от потерь.

* Возможные причины потерь:
  + Аппаратные сбои.
  + Ошибки пользователя.
  + Атаки вредоносного ПО.
* Рекомендации:
  + Использовать временные метки для резервных копий.
  + Поддерживать несколько версий файлов для восстановления в случае ошибок.

#### 5. **Журналирование**

Журналирование фиксирует события и действия, выполняемые демоном.

* Преимущества:
  + Отслеживание текущего состояния.
  + Упрощение диагностики и устранения ошибок.
* Логи могут сохраняться в файлы или системный журнал, доступный через **journalctl**.

#### 6. **Безопасность**

Безопасность резервных данных — приоритетный аспект.

* Доступ к конфигурационным файлам и директориям должен быть строго ограничен.
* Использование:
  + Прав доступа (chmod, chown).
  + Аутентификации для управления.

#### 7. **Мониторинг и отладка**

Контроль за работой демона позволяет своевременно выявлять сбои.

* Инструменты мониторинга:
  + **top**, **htop**, **ps**: отображают активные процессы.
* Для улучшения стабильности демона:
  + Использовать встроенные механизмы отладки.
  + Логировать ошибки для их последующего анализа.

# Основные шаги программы

* **Разработка конфигурационного файла**: Создан конфигурационный файл **CONFIG.ini**, в котором указаны необходимые параметры, такие как исходный каталог для резервного копирования, каталог для сохранения резервных копий, частота выполнения резервного копирования и другие настройки.
* **Создание основного кода демона**: Написан код на языке **Python** c помощью модуля **Daemon** в файле **protodaemon.py**, который включает в себя следующие ключевые компоненты:
  + Считывание параметров из конфигурационного файла **CONFIG.ini** с использованием подходящих библиотек для работы с файлами.
  + Основной цикл, который позволяет демону работать в фоновом режиме, ожидая наступления времени для следующего резервного копирования.
  + Логика для создания резервных копий, которая копирует файлы из исходного каталога в каталог резервных копий с добавлением временной метки в имя резервной копии.
* **Реализация журналирования**: В файле **protodaemon.py** добавлены механизмы для журналирования событий, связанных с выполнением резервного копирования, включая успешный старт и завершение операций.
* **Интеграция с системой**: Подготовлен файл **autostart.py**, с помощью которого будет возможно запускать работу демона при запуске системы
* **Создание интерфейса управления**: Интерфейс написан на основе модуля **Tkinter,** находится в файле **interface.py**
* **Обеспечение безопасности**: Настроены права доступа к конфигурационному файлу **CONFIG.ini** и каталогам резервного копирования, чтобы только определенные пользователи могли их изменять или просматривать. Это повышает уровень безопасности и защищает данные от несанкционированного доступа.

# Описание программы

Программная реализация демона резервного копирования написана на языке **Python 3.12.7**. В программе используются такие модули как Tkinter,Daemon,os,shutil,configparser,logging,datetime,signal

В процессе разработки программы использовалось 8 основных функций, каждая из которых отвечает за определенные действия:

Таблица 1. protodaemon.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| Read\_config | Чтение конфигурационного файла CONFIG.INI | Словарь с параметрами конфигурации |
| setup\_logging | Настройка логирования для записи информации о выполнении операций в указанный лог-файл. | None |
| Reserve\_files | |  | | --- | |  |  |  | | --- | | Функция, с помощью которой осуществляется резервирование файлов | | None |
| Write\_pidfile | |  | | --- | |  |  |  | | --- | | Функция, с помощью которой в файл записывается PID нынешнего работающего демона | | None |
| run\_daemon | |  | | --- | |  |  |  | | --- | | Основной цикл демона: чтение конфигурации, создание резервных копий и ожидание следующего запуска. | | None |
| Run\_Daemon | |  | | --- | |  |  |  | | --- | | Функция запуска демона. | | None |
| Start\_daemon | Функция для кнопки в графическом интерфейсе. С помощью нее запускается основная функция демона run\_daemon | None |
| Kill\_daemon | Функция для кнопки в графическом интерфейсе. С помощью нее убивается демон и удаляется PID-FILE | None |

# Рекомендации пользователя

* **Настройка конфигурационного файла:**
  + Откройте файл **CONFIG.ini** в любом текстовом редакторе.
  + Убедитесь, что в разделе **[Settings]** указаны корректные пути для исходного каталога, каталога для резервного копирования и лог-файла, а также задан правильный интервал времени для резервного копирования в секундах.

Пример содержимого файла **backup\_config.ini** (Рис. 1):

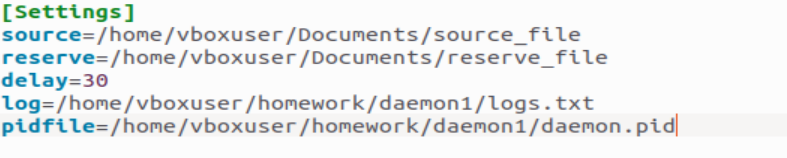


Рис. 1 Файл конфигурации

* **Запуск демона:** Для автоматического запуска демона при загрузке системы и корректного управления им, используется bash-скрипт, который запускается при запуске системы с помощью Startup Applications (Рис. 2,3):

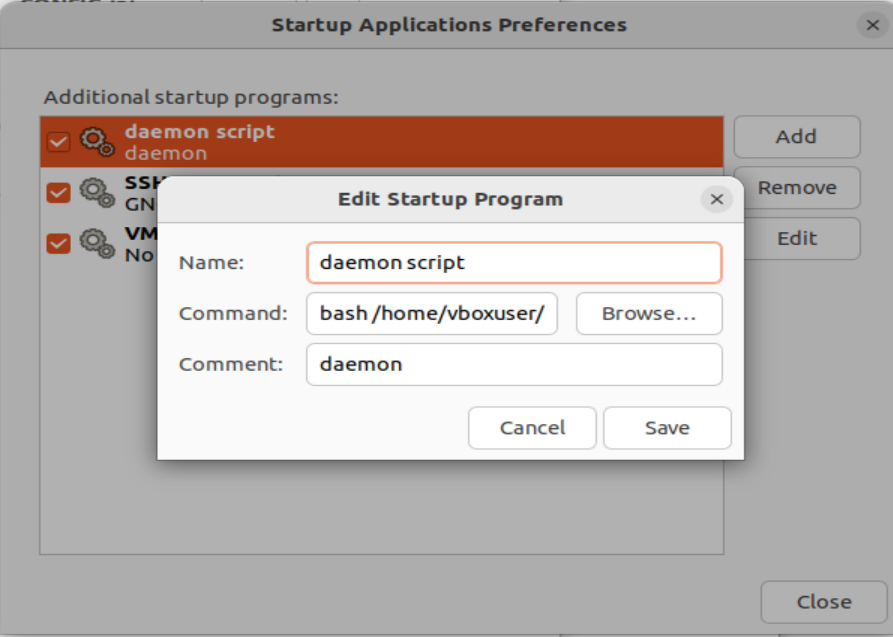


Рис. 2 добавление в систему

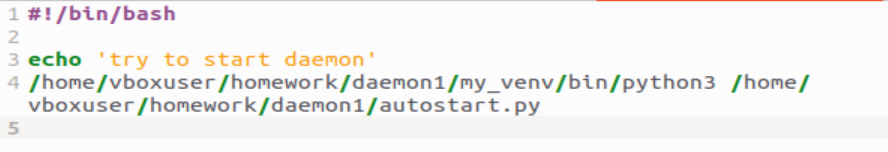


Рис. 3 bash-script

* **Просмотр логов:** Вся информация о выполнении резервного копирования записывается в лог-файл, указанный в конфигурационном файле (**logs.txt)**/

# Рекомендации программиста

Для корректной работы программы вам понадобятся следующие библиотеки Python:

* **configparser**: Для работы с конфигурационными файлами.



* **logging**: Для ведения логов. Входит в стандартную библиотеку Python, дополнительной установки не требуется.
* **shutil**: Для копирования файлов и директорий. Входит в стандартную библиотеку Python, установка не требуется.
* **os**: для работы с системой. Установка не требуется
* **signal** и **sys**: Для обработки сигналов. Эти библиотеки входят в стандартную библиотеку Python.
* **Daemon: для создания демона**
* **Tkinter:** для запуска графического интерфейса

**Структура проекта:**

* Убедитесь, что в конфигурационном файле указаны корректные пути к исходной папке для резервного копирования и папке назначения.
* Убедитесь, что у программы есть права на чтение и запись в эти каталоги.
* Для автоматического запуска демона после перезагрузки, добавьте скрипт в Startup Applications

# Исходный код программы

https://github.com/Browochka/linux-y2

# Контрольный пример

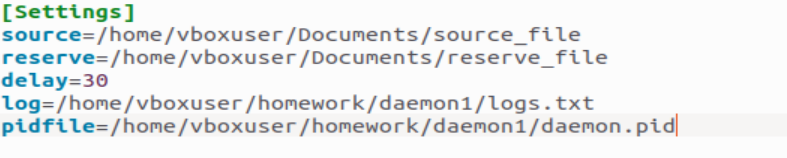
* 1. **Подготовка конфигурационного файла:** Подготовьте файл конфигурации **CONFIG.ini** со следующим содержимым (Рис. 4): 

Рис. 4 Файл конфигурации

* 1. **Запуск демона через interface.py:** Запустите графический интерфейс, на котором можно активировать, остановить или перезапустить демона (Рис. 5):

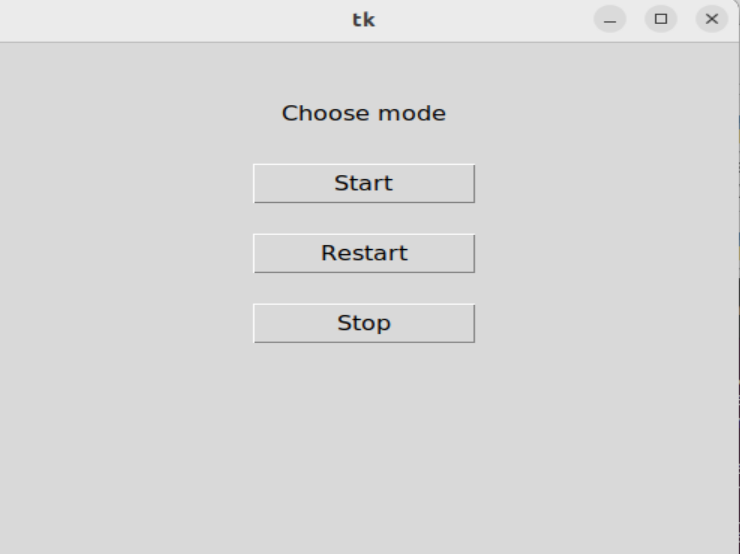


Рис. 5 Графический интерфейс

* 1. **Проверка работы демона:** В daemon.pid вы сможете найти PID демона и проверить его существование в системе (Рис.6):



Рис. 6 Проверка работоспособности

* 1. **Создание резервной копии:** Раз в небольшой промежуток времени осуществляется резервное копирование файлов из одного файла в другой, по указанному маршруту в **CONFIG.INI**

# Вывод

В ходе работы был создан демон для автоматического резервного копирования данных, который позволяет регулярно создавать копии файлов в фоновом режиме. Программа реализована на языке **Python** с использованием библиотек для работы с файлами, логированием и управления процессами. Демон можно легко интегрировать в систему и настроить для автоматического запуска при старте операционной системы.

Разработанный демон обладает гибкой конфигурацией, которая позволяет пользователю указать исходные и целевые каталоги для резервного копирования, настроить интервал выполнения задач, а также просматривать логи операций. Программа протестирована в различных сценариях и позволяет эффективно управлять резервным копированием данных с минимальной нагрузкой на систему.

# Источники

* **configparser** — Для работы с конфигурационными файлами.  
  [Документация configparser](https://docs.python.org/3/library/configparser.html)  
  Дата обращения: (12.10.2024)
* **logging** — Для ведения логов. Входит в стандартную библиотеку Python, дополнительной установки не требуется.  
  [Документация logging](https://docs.python.org/3/library/logging.html)  
  Дата обращения: (12.10.2024)
* **shutil** — Для копирования файлов и директорий. Входит в стандартную библиотеку Python, установка не требуется.  
  [Документация shutil](https://docs.python.org/3/library/shutil.html)  
  Дата обращения: (12.10.2024)
* **argparse** — Для обработки аргументов командной строки. Входит в стандартную библиотеку Python, установка не требуется.  
  [Документация argparse](https://docs.python.org/3/library/argparse.html)  
  Дата обращения: (12.10.2024)
* **signal и sys** — Для обработки сигналов завершения и управления процессом. Эти библиотеки входят в стандартную библиотеку Python.  
  [Документация signal](https://docs.python.org/3/library/signal.html)  
  Дата обращения: (12.10.2024)