**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

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

**ОТЧЕТ**

**по лабораторной работе №3**

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

**на тему**

**«Разработка системного инструмента для аудита системы»**

**Студент гр. 23Б16-пу**

**Шарабарин М.С.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc182212872)
2. [Описание задачи (формализация задачи) 4](#_Toc182212873)
3. [Теоретическая часть 5](#_Toc182212874)

**[Методы обнаружения подозрительного трафика](#_Toc182212875)** [5](#_Toc182212875)

**[Библиотека Scapy для анализа трафика](#_Toc182212876)** [5](#_Toc182212876)

**[Блокировка подозрительного трафика с помощью iptables](#_Toc182212877)** [6](#_Toc182212877)

1. [Основные шаги программы 7](#_Toc182212878)
2. [Описание программы 9](#_Toc182212879)
3. [Рекомендации пользователя 11](#_Toc182212880)
4. [Рекомендации программиста 12](#_Toc182212881)

**[1. Установка необходимых библиотек Python](#_Toc182212882)**[: 12](#_Toc182212882)

**[2](#_Toc182212883)**[.](#_Toc182212883) **[Установка утилит для работы с iptables:](#_Toc182212883)** [12](#_Toc182212883)

1. [Исходный код программы 12](#_Toc182212884)
2. [Контрольный пример 13](#_Toc182212885)
3. [Вывод 14](#_Toc182212886)
4. [Источники 15](#_Toc182212887)

# Цель работы

Целью данной работы является разработка системного инструмента для аудита и мониторинга операционной системы Linux. Разрабатываемая программа будет предназначена для регистрации и отслеживания значимых событий, происходящих в системе, таких как запуск и завершение процессов, изменение файлов, сетевые операции и другие важные действия. Вся информация о событиях будет записываться в журнал событий, что позволит администратору системы осуществлять эффективный мониторинг и анализ происходящих процессов для обеспечения безопасности и стабильности системы.

# Описание задачи (формализация задачи)

Задача заключается в разработке системного инструмента для комплексного аудита и мониторинга операционной системы **Linux**. Программа должна выполнять следующие функции:

1. **Мониторинг файловой системы**:
   * Отслеживание создания, удаления, перемещения и изменения файлов и директорий в заданной папке.
   * Логирование информации о типе файловой операции, пути к файлу или папке, времени операции, а также данных о временных файлах, созданных редакторами, чтобы избежать излишнего логирования.
2. **Мониторинг процессов**:
   * Регистрация запуска и завершения процессов в системе, включая идентификаторы процессов (PID) и команды, которые они выполняют.
   * Обеспечение корректного обновления списка процессов для отслеживания новых запусков и завершений.
3. **Мониторинг сетевых операций**:
   * Логирование информации о TCP и UDP соединениях, включая локальные IP-адреса и порты.
   * Преобразование шестнадцатеричных IP-адресов из системных файлов в читаемый формат.
4. **Журналирование данных**:
   * Запись всех зарегистрированных событий в отдельные журнальные файлы для дальнейшего анализа. Журналы должны содержать метку времени и подробности о событии (например, путь к файлу или PID процесса).
   * Использование библиотеки **logging** для записи логов и организации их формата.

**Ожидаемый результат**: Программа должна корректно фиксировать указанные события в реальном времени и записывать их в файлы журналов. Мониторинг процессов и сетевых операций должен обеспечивать сбор подробной информации о системных изменениях, что поможет в последующем анализе состояния системы и ее безопасности.

# Теоретическая часть

1. **Системные вызовы Linux**:
   * Операционная система **Linux** предоставляет богатый набор системных вызовов, которые позволяют взаимодействовать с ядром и получать информацию о процессе и файловой системе. В данном инструменте используются вызовы для чтения содержимого папки **/proc**, чтобы отслеживать запущенные процессы и их идентификаторы.
2. **Механизм работы с процессами**:
   * В **Linux** каждый процесс имеет уникальный идентификатор (PID), и информация о процессах хранится в папке **/proc**, где файлы с именами, совпадающими с **PID**, содержат данные о каждом процессе. В частности, файл **/proc/[PID]/cmdline** содержит команду, с помощью которой был запущен процесс.
3. **Аудит сетевых операций**:
   * Сетевые соединения можно отслеживать, анализируя содержимое файлов в папке **/proc/net**, где хранятся данные о текущих сетевых соединениях. Файл **/proc/net/tcp** содержит информацию о **TCP-соединениях**, а файл **/proc/net/udp** — о **UDP-соединениях**.
   * Для обработки адресов, представленных в шестнадцатеричном формате, используется функция преобразования **socket.inet\_ntoa**, которая конвертирует их в читаемый формат **IP-адреса**.
4. **Логирование и обработка событий**:
   * Для записи данных о событиях используется библиотека **logging**, которая позволяет создавать и конфигурировать логгер с настройкой уровня логирования, формата записи и вывода логов в файл.
5. **Многопоточность**:
   * Для одновременного выполнения различных операций, таких как мониторинг процессов и сетевых соединений, используется библиотека **threading**, которая позволяет запускать функции в отдельных потоках. Это обеспечивает асинхронное выполнение задач и повышает эффективность работы программы, позволяя ей обрабатывать различные виды событий параллельно.
6. **Библиотека watchdog для мониторинга файлов**:
   * Библиотека **watchdog** предоставляет простой способ мониторинга файловой системы, позволяя обрабатывать события, такие как создание, удаление, модификация и перемещение файлов. Это позволяет удобно отслеживать изменения в указанных каталогах и фиксировать их в журналах.

# Основные шаги программы

Программа состоит из нескольких ключевых компонентов, которые выполняют функции мониторинга и логирования различных операций в системе **Linux**. Ниже описаны основные шаги, выполняемые при запуске программы:

* **Инициализация и настройка логирования**:
  + Создаются логгеры для записи данных о событиях файловой системы, процессах и сетевых соединениях. Для этого используется функция **setup\_logger**, которая конфигурирует логгеры и устанавливает формат записи и путь к файлу логов.
  + Логгеры инициализируются с разными именами и записывают данные в отдельные файлы (**file\_system\_log.txt, process\_log.txt, network\_log.txt**).
* **Запуск мониторинга процессов**:
  + В отдельном потоке запускается функция **monitor\_processes**, которая постоянно проверяет изменения в списке процессов, находящихся в системе.
  + Для каждого нового процесса, обнаруженного в папке **/proc**, программа записывает в лог сообщение с его **PID** и командой запуска.
  + Для завершившихся процессов также записывается соответствующее сообщение в лог.
* **Запуск мониторинга сетевых соединений**:
  + В отдельном потоке запускается функция **monitor\_network**, которая читает данные о сетевых соединениях из файлов **/proc/net/tcp** и **/proc/net/udp**.
  + Для каждого соединения программа извлекает **IP-адрес** и порт и записывает информацию о соединении в лог.
* **Настройка и запуск мониторинга файловой системы**:
  + Создается экземпляр класса **FileAuditHandler**, который является наследником **FileSystemEventHandler** из библиотеки **watchdog**. Этот класс определяет методы для обработки событий создания, удаления, перемещения и модификации файлов.
  + Путь к папке, за которой требуется наблюдать, передается в метод **observer.schedule**, чтобы настроить наблюдателя для мониторинга указанного каталога и его подкаталогов.
* **Обработка завершения работы программы**:
  + При завершении программы с помощью **KeyboardInterrupt** вызывается метод **observer.stop(),** который останавливает мониторинг.
  + Программа выводит сообщение о завершении мониторинга и ожидает завершения всех потоков.

# Описание программы

Программа представляет собой инструмент для мониторинга системы на уровне файловой системы, процессов и сетевых соединений, написанный на языке **Python**. Она использует библиотеку **watchdog** для отслеживания изменений в файловой системе и различные стандартные модули Python для мониторинга процессов и сетевых соединений. Основные функции программы включают:

* **Мониторинг процессов**: программа отслеживает запуск и завершение процессов в системе, записывая информацию о них в лог.
* **Мониторинг сетевых соединений**: программа отслеживает текущие TCP и UDP соединения, преобразует их IP-адреса из шестнадцатеричного формата в обычный и логирует данные о них.
* **Аудит файловой системы**: программа отслеживает создание, удаление и перемещение файлов и папок, игнорируя временные файлы, и записывает эту информацию в лог.

Таблица 1. audit\_tool.py

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Описание | Параметры функции | Описание параметров |
| setup\_logger | Настройка логгера для записи сообщений в файл | log\_file, name | log\_file — путь к файлу для логирования; name — имя логгера |
| monitor\_processes | Мониторинг процессов в системе | logger | logger — объект логгера для записи данных о сетевых соединениях |
| monitor\_network | Мониторинг сетевых соединений | logger | logger — объект логгера для записи данных о сетевых соединениях |
| hex\_to\_ip | |  | | --- | |  |  |  | | --- | | Преобразование IP-адреса из шестнадцатеричного формата в строковый вид | | hex\_ip | hex\_ip — строка с шестнадцатеричным представлением IP-адреса |
| FileAuditHandler | |  | | --- | |  |  |  | | --- | | Обработчик событий файловой системы | | logger | logger — объект логгера для записи данных о сетевых соединениях |
| on\_created | Обработка события создания файла или папки | event | |  | | --- | |  |  |  | | --- | | event — объект события с информацией о перемещении | |
| on\_deleted | Обработка события удаления файла или папки | event | |  | | --- | |  |  |  | | --- | | event — объект события с информацией о перемещении | |
| on\_moved | Обработка события перемещения или переименования файла | event | |  | | --- | |  |  |  | | --- | | event — объект события с информацией о перемещении | |
| find\_original\_file | Поиск оригинального файла для временного файла | temp\_path | temp\_path — путь к временному файлу |

# 

Таблица 2. filtering.py

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Описание | Параметры функции | Описание параметров |
| \_\_init\_\_ | Инициализирует приложение и создает интерфейс пользователя. | root | |  | | --- | |  |  |  | | --- | | Основное окно приложения (объект Tk из библиотеки tkinter). | |
| load\_file | Загружает выбранный лог-файл и отображает его содержимое. | event | Событие выбора файла в Combobox (по умолчанию None). |
| display\_file\_content | Отображает содержимое текущего выбранного лог-файла в текстовой области. | Нет | Нет |
| apply\_filter | Применяет фильтры по пользователю, событию и временному интервалу к лог-файлу. | Нет | Нет |
| search | Выполняет поиск по ключевому слову в содержимом лог-файла. | Нет | Нет |
| display\_filtered\_content | Отображает отфильтрованные строки в текстовой области. | lines | |  | | --- | |  |  |  | | --- | | Список строк, которые необходимо отобразить. | |
| show\_error | Отображает сообщение об ошибке в диалоговом окне. | message | Текст сообщения об ошибке. |

# Рекомендации пользователя

 **Запуск программы**:

* Откройте терминал или командную строку и перейдите в папку с программой.
* Запустите программу с помощью команды:



Здесь <путь\_к\_папке> — это путь к папке, за которой вы хотите вести мониторинг.

 **Использование программы**:

* После запуска программа начнет отслеживать изменения в указанной папке. Она будет логировать события, такие как создание, удаление и перемещение файлов и папок, а также запуск и завершение процессов и сетевые соединения.
* Логи будут записываться в файлы:
  + file\_system\_log.txt — для событий файловой системы.
  + process\_log.txt — для мониторинга процессов.
  + network\_log.txt — для сетевых соединений.
* Вы можете просмотреть эти логи, открыв их с помощью текстового редактора или командой cat в терминале.
* При желании можно воспользоваться фильтром запустив в терминале интерфейс при помощи команды: python3 filtering.py

# Рекомендации программиста

Для успешного запуска и работы программы важно проверить несколько аспектов и убедиться, что установлены необходимые библиотеки. Вот шаги, которые нужно выполнить:

### **1.** **Проверка установленной версии Python**

* Убедитесь, что у вас установлен Python 3.x. Проверьте версию Python с помощью команды:



* Если Python не установлен, загрузите и установите его с официального сайта [python.org](https://www.python.org/downloads/" \t "_new).

### **2.** **Установка необходимых библиотек**

* Проверьте, что у вас установлены все необходимые библиотеки. В программе используются следующие модули:
  + **os**, **sys**, **logging**, **time**, **socket**, **struct** — встроенные модули Python, устанавливать их не нужно.
  + **threading** — также встроенный модуль.
  + **watchdog** — внешний модуль для отслеживания изменений файловой системы.
* Установите **watchdog**, если он не установлен, с помощью команды:



# Исходный код программы

# Контрольный пример

* 1. **Запуск программы:** запустите файл **audit\_tool.py** в терминале при помощи команды (Рис. 1):

Рис. 1 запуск программы

* 1. **Ожидание:** Для примера можно поменять, удалить, создать файлы или папки в выбранной директории.
  2. **Просмотр логов:** В директории с кодом появятся три файла (Рис.2) в которых можно будет обнаружить все изменения которые вы делали или происходили в системе (Рис.3) (Рис.4) (Рис.5):

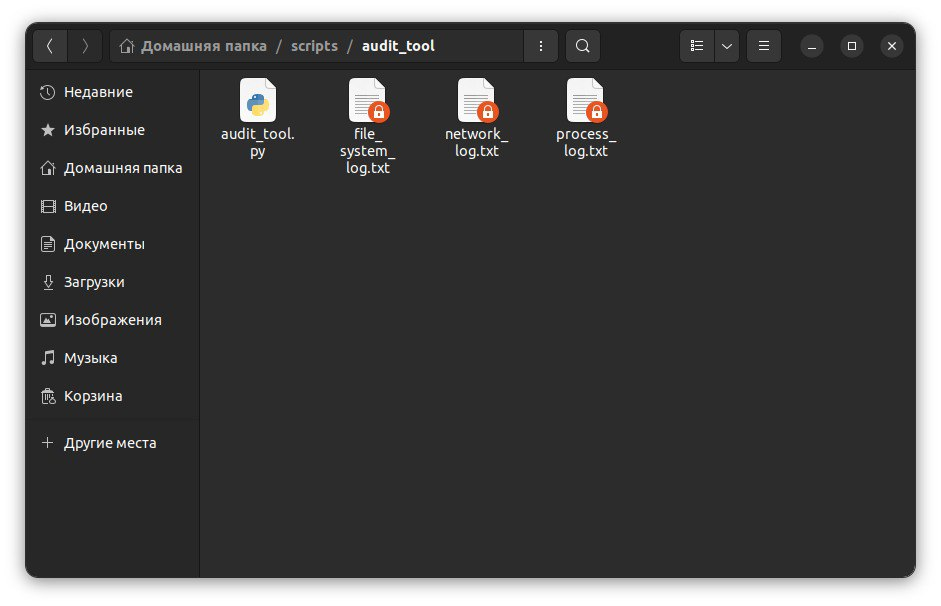


Рис. 2 Файлы в директории

Рис. 3 file\_system\_log.txt

Рис. 4 network\_log.txt

Рис. 5 process\_log.txt

* 1. **Фильтрация:** Если необходимо изучить лишь часть данных из этих файлов то можно воспользоваться filtering.py. (Рис. 6)

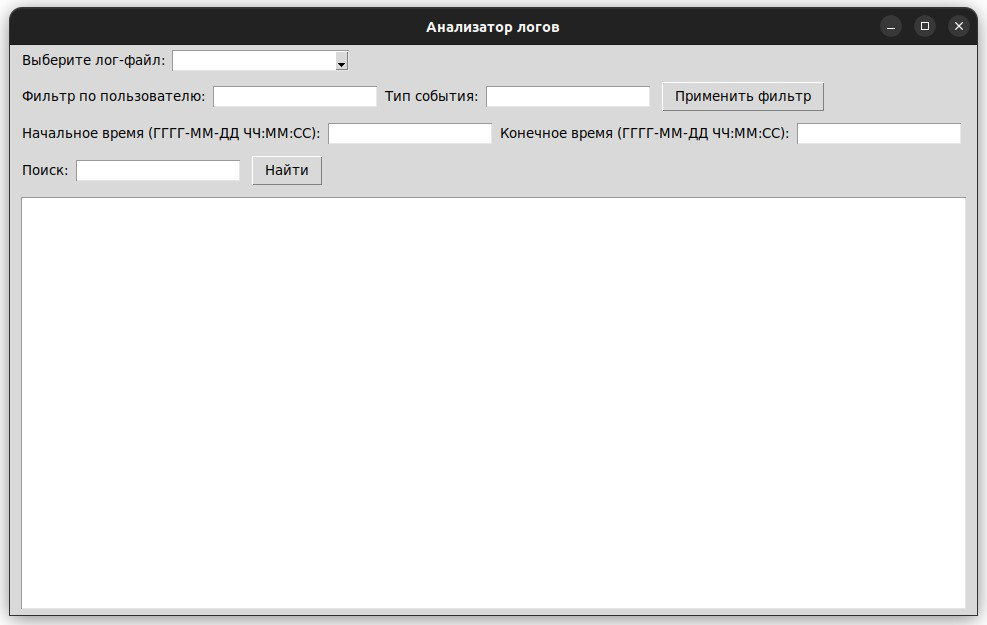


Рис. 6 filtering.txt

# Вывод

Разработанная программа является полезным инструментом для мониторинга изменений в файловой системе, процессов и сетевых соединений на **Linux**. Она позволяет в реальном времени отслеживать создание, удаление и перемещение файлов, а также запуск и завершение процессов, что может быть полезно для мониторинга активности на сервере или в рабочей среде.

Программа эффективно использует многозадачность, что позволяет одновременно мониторить процессы и сетевые операции, минимизируя при этом нагрузку на основной процесс. Все события логируются в файлы, что позволяет сохранять информацию для дальнейшего анализа. Такой подход помогает выявлять несанкционированные изменения или следить за активностью программ и пользователей в системе.

Кроме того, программа использует популярные Python-библиотеки, такие как **watchdog** для мониторинга файловой системы и **logging** для ведения журналов. Мониторинг процессов и сетевых соединений реализован через чтение данных из директории **/proc** в **Linux**, что делает программу особенно полезной для системных администраторов и специалистов по безопасности, работающих в средах **Linux**.

# Источники

• **watchdog** — Библиотека для мониторинга изменений в файловой системе.  
*[https://python-watchdog.readthedocs.io/en/stable/](https://python-watchdog.readthedocs.io/en/stable/" \t "_new)*  
Дата обращения: 08.12.2024

• **logging** — Модуль для ведения журналов в Python.  
*[https://docs.python.org/3/library/logging.html](https://docs.python.org/3/library/logging.html" \t "_new)*  
Дата обращения: 08.12.2024

• **socket** — Библиотека для работы с сетевыми соединениями в Python.  
*[https://docs.python.org/3/library/socket.html](https://docs.python.org/3/library/socket.html" \t "_new)*Дата обращения: 08.12.2024

• **struct** — Библиотека для работы с двоичными данными и упаковкой/распаковкой данных в Python.  
*[https://docs.python.org/3/library/struct.html](https://docs.python.org/3/library/struct.html" \t "_new)*  
Дата обращения: 08.12.2024