

**Санкт-Петербургский государственный университет**

Направление "Большие данные и распределенная цифровая платформа"

**Лабораторная работа по дисциплине  
Системное программирование в Linux**

**"Создание системного инструмента для аудита и  
мониторинга Linux."**

Выполнил:

Зайнуллин Мансур Альбертович

Группа: 23.Б16-пу

Руководитель:

Киямов Жасур Уткирович

Санкт-Петербург

2024

# Оглавление

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Описание задачи</b>	<b>4</b>
<b>3</b>	<b>Теоретическая часть</b>	<b>6</b>
3.1	Системные вызовы и мониторинг . . . . .	6
3.2	Процессы и PID . . . . .	6
3.3	Мониторинг файловой системы с использованием inotify . . .	6
3.4	Сетевые соединения и их мониторинг . . . . .	6
3.5	Базы данных для логирования событий . . . . .	7
3.6	Отправка уведомлений по электронной почте . . . . .	7
<b>4</b>	<b>Описание программы</b>	<b>8</b>
4.1	Общая структура программы . . . . .	8
4.2	Алгоритм работы программы . . . . .	8
4.3	Функции и их описание . . . . .	10
4.3.1	main.py . . . . .	10
4.3.2	monitor.py . . . . .	10
4.3.3	gui.py . . . . .	11
4.3.4	logger.py . . . . .	11
4.3.5	config.py . . . . .	12
4.3.6	notifier.py . . . . .	12
4.3.7	reports.py . . . . .	12
<b>5</b>	<b>Рекомендации пользователю</b>	<b>13</b>
5.1	Инструкция по установке сканера на Ubuntu 24 и ниже . . . .	13
5.1.1	Шаг 1: Установка Python 3.9 . . . . .	13
5.1.2	Шаг 2: Установка Git . . . . .	13
5.1.3	Шаг 3: Клонирование репозитория . . . . .	13
5.1.4	Шаг 4: Создание виртуального окружения и установка зависимостей . . . . .	14
5.1.5	Шаг 5: Настройка конфигурации . . . . .	14
5.2	Инструкция по эксплуатации сканера . . . . .	15

5.2.1	Шаг 1: Запуск программы с привилегиями суперпользователя . . . . .	15
5.2.2	Шаг 2: Настройка config.json . . . . .	15
5.2.3	Шаг 3: Использование графического интерфейса . . .	17
5.2.4	Шаг 4: Остановка программы . . . . .	17
<b>6</b>	<b>Контрольный пример</b>	<b>18</b>
<b>7</b>	<b>Выводы по работе</b>	<b>19</b>
<b>8</b>	<b>Полезные ссылки</b>	<b>20</b>

# **1 Цель работы**

Цель данной работы разработать системный инструмент, который будет использоваться для аудита и мониторинга системы Linux.

## **2 Описание задачи**

Необходимо разработать системный инструмент для аудита и мониторинга операционной системы Linux. Инструмент должен выполнять следующие функции:

### **1. Мониторинг процессов:**

- Отслеживание запуска и завершения процессов.
- Сбор информации о пользователе, PID и командной строке процесса.

### **2. Мониторинг файловой системы:**

- Отслеживание создания, удаления и изменения файлов и каталогов в указанных директориях.
- Регистрация путей изменений и пользователей, выполняющих операции.

### **3. Мониторинг сетевых операций:**

- Отслеживание установленных и завершенных сетевых соединений.
- Сбор информации о пользователе и процессе, связанном с соединением.

### **4. Логирование событий:**

- Сохранение информации о всех зарегистрированных событиях в базе данных SQLite.
- Реализация механизма ротации и архивации журналов для предотвращения переполнения диска.

### **5. Фильтрация и поиск событий:**

- Возможность поиска событий по различным критериям: пользователь, тип события, дата и время.

## **6. Оповещения:**

- Отправка уведомлений по электронной почте при возникновении определенных событий.

## **7. Создание отчетов:**

- Генерация статистических отчетов с использованием графиков для анализа собранных данных.

## **3 Теоретическая часть**

### **3.1 Системные вызовы и мониторинг**

Системные вызовы являются интерфейсом между пользовательскими приложениями и ядром операционной системы. Для мониторинга событий в системе Linux используются различные системные вызовы и механизмы, такие как `ptrace`, `inotify` и библиотеки, предоставляющие доступ к информации о процессах и файловой системе.

### **3.2 Процессы и PID**

Процесс — это выполняющаяся программа, которая содержит код, данные и системные ресурсы, необходимые для выполнения. Каждый процесс в системе идентифицируется уникальным числовым идентификатором — PID (Process ID). PID используется для управления процессами, отслеживания их состояния и взаимодействия между ними.

### **3.3 Мониторинг файловой системы с использованием `inotify`**

`inotify` — это интерфейс ядра Linux, позволяющий приложениям отслеживать изменения в файловой системе в реальном времени. С помощью `inotify` можно регистрировать события создания, удаления, изменения и других операций с файлами и каталогами. Это позволяет эффективно мониторить важные директории и реагировать на изменения.

### **3.4 Сетевые соединения и их мониторинг**

Сетевые соединения устанавливаются между процессами на различных узлах сети. Мониторинг сетевых соединений включает отслеживание установленных и закрытых соединений, сбор информации о портах, IP-адресах и связанных процессах. Инструменты, такие как `psutil`, предоставляют доступ к информации о сетевых соединениях и их состоянии.

### **3.5 Базы данных для логирования событий**

Для хранения информации о зарегистрированных событиях используется база данных SQLite. SQLite — легковесная реляционная база данных, которая не требует отдельного сервера и легко интегрируется с приложениями на Python. Использование базы данных позволяет эффективно хранить, организовывать и быстро выполнять запросы к журналу событий.

### **3.6 Отправка уведомлений по электронной почте**

Отправка уведомлений по электронной почте осуществляется с помощью протокола SMTP (Simple Mail Transfer Protocol). В Python для этой цели используется модуль `smtplib`, который позволяет устанавливать соединение с SMTP-сервером, аутентифицироваться и отправлять электронные письма. Это обеспечивает возможность своевременного информирования пользователей о важных событиях в системе.



## 4 Описание программы

### 4.1 Общая структура программы

Программа состоит из нескольких модулей, каждый из которых отвечает за определённую функциональность системы аудита и мониторинга Linux. Основные модули включают:

- `main.py`: Точка входа в приложение. Инициализирует базу данных, запускает процессы мониторинга и запускает графический интерфейс.
- `monitor.py`: Содержит классы для мониторинга процессов, файловой системы и сетевых соединений.
- `gui.py`: Реализует графический интерфейс пользователя для отображения и управления событиями.
- `logger.py`: Обрабатывает логирование событий в базу данных и управление ротацией логов.
- `config.py`: Загружает конфигурационные параметры из файла `config.json`.
- `notifier.py`: Отвечает за отправку уведомлений по электронной почте.
- `reports.py`: Генерирует статистические отчёты на основе данных из базы данных.

### 4.2 Алгоритм работы программы

#### 1. Инициализация:

- `main.py` вызывает функцию `init_db()` из модуля `logger.py` для инициализации базы данных SQLite, если она ещё не создана.

#### 2. Ротация логов:

- Запускается отдельный поток, который периодически (раз в сутки) вызывает функцию `rotate_logs()` из `logger.py` для удаления старых записей из базы данных на основе параметра `log_rotation_days` из `config.json`.

### 3. Запуск мониторов:

- **ProcessMonitor:** Отслеживает запуск и завершение процессов, используя библиотеку `psutil`. Новые процессы добавляются в кэш и логируются, завершённые процессы удаляются из кэша и также логируются.
- **FileMonitor:** Использует `pyinotify` для отслеживания изменений в указанных директориях (`monitor_paths`). Регистрирует события создания, удаления и изменения файлов.
- **NetworkMonitor:** Мониторит сетевые соединения, отслеживая установку и завершение соединений, а также связанные с ними процессы.

### 4. Логирование событий:

- Все зарегистрированные события сохраняются в базе данных SQLite через функции из `logger.py`.

### 5. Оповещения:

- При регистрации определённых событий (например, запуск процесса или изменение файла) функция `send_email_notification()` из `notifier.py` отправляет уведомления по электронной почте, если это настроено в `config.json`.

### 6. Графический интерфейс:

- `gui.py` запускает окно приложения, которое позволяет пользователю просматривать события, фильтровать их по различным критериям, искать определённые события и генерировать отчёты.

## 4.3 Функции и их описание

### 4.3.1 `main.py`

- `main()`: Главная функция приложения. Инициализирует базу данных, запускает поток ротации логов, запускает мониторы процессов, файловой системы и сетевых соединений в отдельных потоках, и запускает графический интерфейс.

### 4.3.2 `monitor.py`

- **ProcessMonitor:**
  - `start_monitoring()`: Постоянно отслеживает текущие PID, выявляет новые и завершённые процессы, логирует соответствующие события.
  - `trace_process(pid)`: Сбор информации о новом процессе и логирование события его запуска.
  - `handle_terminated_process(pid)`: Логирование события завершения процесса.
  - `cleanup_cache()`: Удаление неактуальных данных из кэша процессов.
- **FileMonitor:**
  - `process_IN_CREATE(event)`: Обработка события создания файла или каталога.
  - `process_IN_DELETE(event)`: Обработка события удаления файла или каталога.
  - `process_IN_MODIFY(event)`: Обработка события изменения файла.
  - `log_event(event_type, event)`: Логирование события файловой системы.
  - `get_file_owner(filepath)`: Получение имени пользователя-владельца файла.

- `start_monitoring()`: Запуск цикла мониторинга файловой системы.

- **NetworkMonitor:**

- `start_monitoring()`: Постоянно отслеживает текущие сетевые соединения, выявляет новые и завершённые соединения, логирует соответствующие события.

### 4.3.3 `gui.py`

- **AuditApp:**

- `create_widgets()`: Создание элементов графического интерфейса, включая поля для фильтрации, таблицу событий и кнопки управления.
- `load_events()`: Загрузка последних событий из базы данных и отображение их в таблице.
- `search_events()`: Поиск событий по заданным критериям и отображение результатов.
- `show_report()`: Генерация и отображение статистического отчёта на основе данных из базы данных.

### 4.3.4 `logger.py`

- `init_db()`: Инициализация базы данных SQLite, создание таблицы `events` при необходимости.
- `log_event(timestamp, user, pid, event_type, description)`: Вставка записи о событии в таблицу `events`.
- `rotate_logs()`: Удаление записей из таблицы `events`, старше заданного количества дней (`log_rotation_days`).

#### 4.3.5 `config.py`

- `load_config()`: Загрузка конфигурационных параметров из файла `config.json`.
- `config`: Словарь с параметрами конфигурации, доступный для других модулей.

#### 4.3.6 `notifier.py`

- `send_email_notification(subject, body, is_html=False)`: Отправка уведомления по электронной почте с заданной темой и телом сообщения. Использует параметры SMTP из `config.json`.

#### 4.3.7 `reports.py`

- `generate_statistics()`: Генерация статистики событий, подсчёт количества событий по типам для отчётов.

## 5 Рекомендации пользователю

### 5.1 Инструкция по установке сканера на Ubuntu 24 и ниже

Следуйте приведённым ниже шагам для установки и настройки системного инструмента аудита и мониторинга Linux на Ubuntu 24 и ниже.

#### 5.1.1 Шаг 1: Установка Python 3.9

1. Обновите список пакетов:

---

```
sudo apt update
```

---

2. Установите Python 3.9:

---

```
sudo apt install python3.9 python3.9-venv python3.9-dev -y
```

---

3. Убедитесь, что Python 3.9 установлен:

---

```
python3.9 --version
```

---

#### 5.1.2 Шаг 2: Установка Git

1. Установите Git:

---

```
sudo apt install git -y
```

---

2. Проверьте установку Git:

---

```
git --version
```

---

#### 5.1.3 Шаг 3: Клонирование репозитория

1. Перейдите в директорию, куда хотите клонировать проект:

---

```
cd ~
```

---

2. Клонировать репозиторий:

---

```
git clone https://github.com/MansurYa/audit_of_Linux_system.git
```

---

3. Перейдите в директорию проекта:

---

```
cd audit_of_Linux_system
```

---

#### 5.1.4 Шаг 4: Создание виртуального окружения и установка зависимостей

1. Создайте виртуальное окружение:

---

```
python3.9 -m venv venv
```

---

2. Активируйте виртуальное окружение:

---

```
source venv/bin/activate
```

---

3. Установите необходимые пакеты из requirements.txt:

---

```
pip install --upgrade pip  
pip install -r requirements.txt
```

---

#### 5.1.5 Шаг 5: Настройка конфигурации

1. Откройте файл config.json для редактирования:

---

```
nano config.json
```

---

2. Отредактируйте параметры согласно вашим требованиям. Пример конфигурации:

---

```
{  
    "db_path": "event_log.db",  
    "log_rotation_days": 7,  
    "monitor_paths": ["/var/log", "/etc"],  
    "email_notifications": true,  
    "email_recipients": ["example@gmail.com"],  
    "smtp_server": "smtp.yandex.ru",
```

```
"smtp_port": 587,  
"smtp_user": "your_email@yandex.ru",  
"smtp_password": "your_password"  
}
```

---

3. Сохраните изменения и закройте редактор (в nano: нажмите Ctrl + O, затем Enter, затем Ctrl + X).

## 5.2 Инструкция по эксплуатации сканера

После установки и настройки сканера выполните следующие шаги для его эксплуатации.

### 5.2.1 Шаг 1: Запуск программы с привилегиями суперпользователя

1. Перейдите в директорию проекта, если вы ещё не там:

---

```
cd ~/audit_of_Linux_system
```

---

2. Активируйте виртуальное окружение:

---

```
source venv/bin/activate
```

---

3. Запустите программу с правами суперпользователя:

---

```
sudo python main.py
```

---

Программа требует привилегий суперпользователя для доступа к системным ресурсам.

### 5.2.2 Шаг 2: Настройка config.json

Файл config.json содержит параметры конфигурации, которые управляют работой программы. Рассмотрим каждое поле:

- **db\_path**: Путь к базе данных SQLite для хранения событий.

- Пример: "db\_path": "event\_log.db"



- **log\_rotation\_days:** Количество дней, по истечении которых старые записи будут удалены из базы данных.
  - Пример: `"log_rotation_days": 7`
- **monitor\_paths:** Список директорий, которые необходимо мониторить на изменения файловой системы.
  - Пример: `"monitor_paths": ["/var/log", "/etc"]`
- **email\_notifications:** Включение (true) или отключение (false) отправки email-уведомлений.
  - Пример: `"email_notifications": true`
- **email\_recipients:** Список адресов электронной почты, на которые будут отправляться уведомления.
  - Пример: `"email_recipients": ["example@gmail.com"]`
- **smtp\_server:** Адрес SMTP-сервера для отправки электронных писем.
  - Пример: `"smtp_server": "smtp.yandex.ru"`
- **smtp\_port:** Порт SMTP-сервера.
  - Пример: `"smtp_port": 587`
- **smtp\_user:** Логин (адрес электронной почты) для аутентификации на SMTP-сервере.
  - Пример: `"smtp_user": "your_email@yandex.ru"`
- **smtp\_password:** Пароль для аутентификации на SMTP-сервере.
  - Пример: `"smtp_password": "your_password"`

### 5.2.3 Шаг 3: Использование графического интерфейса

После запуска программы откроется окно графического интерфейса с основными функциями:

- **Просмотр событий:**
  - Таблица отображает последние зарегистрированные события, включая время, пользователя, PID, тип события и описание.
- **Фильтрация событий:**
  - В верхней части окна находятся поля для фильтрации событий по пользователю, типу события и дате.
  - Введите необходимые параметры и нажмите кнопку " " для отображения соответствующих событий.
- **Обновление списка событий:**
  - Нажмите кнопку "Обновить" для загрузки последних событий из базы данных.
- **Генерация отчётов:**
  - Нажмите кнопку "Генерировать отчет" для создания статистического отчёта. Откроется новое окно с графиком, отображающим количество событий по типам.

### 5.2.4 Шаг 4: Остановка программы

Для остановки программы закройте окно графического интерфейса или нажмите `Ctrl + C` в терминале, где запущена программа.

# 6 Контрольный пример

Системный аудит

Пользователь:

Тип события:

Дата (YYYY-MM-DD):

Поиск

Время	Пользователь	PID	Тип события	Описание
2024-12-20 23:55:56	root	37	Process Started	Process (PID: 37) started
2024-12-20 23:55:56	root	36	Process Started	Process (PID: 36) started
2024-12-20 23:55:56	root	35	Process Started	Process (PID: 35) started
2024-12-20 23:55:55	mansurya	1551	Process Started	Process /usr/bin/dbus-da
2024-12-20 23:55:55	root	22	Process Started	Process (PID: 22) started
2024-12-20 23:55:55	root	12	Process Started	Process (PID: 12) started
2024-12-20 23:55:56	root	31	Process Started	Process (PID: 31) started
2024-12-20 23:55:55	root	10	Process Started	Process (PID: 10) started
2024-12-20 23:55:55	mansurya	1537	Process Started	Process /usr/libexec/at-s
2024-12-20 23:55:56	root	28	Process Started	Process (PID: 28) started
2024-12-20 23:55:55	root	1029	Process Started	Process /usr/libexec/upo
2024-12-20 23:55:55	root	2	Process Started	Process (PID: 2) started
2024-12-20 23:55:55	root	6	Process Started	Process (PID: 6) started
2024-12-20 23:55:55	root	5	Process Started	Process (PID: 5) started
2024-12-20 23:55:55	root	4	Process Started	Process (PID: 4) started
2024-12-20 23:55:55	root	21	Process Started	Process (PID: 21) started
2024-12-20 23:55:55	root	3	Process Started	Process (PID: 3) started
2024-12-20 23:55:55	root	19	Process Started	Process (PID: 19) started
2024-12-20 23:55:55	root	18	Process Started	Process (PID: 18) started
2024-12-20 23:55:55	root	17	Process Started	Process (PID: 17) started
2024-12-20 23:55:55	root	16	Process Started	Process (PID: 16) started
2024-12-20 23:55:55	root	15	Process Started	Process (PID: 15) started
2024-12-20 23:55:55	root	14	Process Started	Process (PID: 14) started
2024-12-20 23:55:55	root	13	Process Started	Process (PID: 13) started
2024-12-20 23:55:55	mansurya	1538	Process Started	Process /usr/bin/gnome-
2024-12-20 23:55:55	root	7	Process Started	Process (PID: 7) started
2024-12-20 23:55:55	root	1	Process Started	Process /sbin/init (PID: 1)
2024-12-20 18:04:19	mongodb	None	File Modified	/var/log/mongodb/mong
2024-12-20 18:04:18	mansurya	11786	Network Connection	addr(ip='212.34.139.173
2024-12-20 18:04:18	mansurya	11786	Network Connection	addr(ip='212.34.139.173

Обновить

Генерировать отчет

рис. 1 Демонстрация работы программы

## **7 Выводы по работе**

В ходе работы был разработан системный инструмент для аудита и мониторинга операционных систем Linux. Программа способна отслеживать запуск и завершение процессов, изменения файловой системы, сетевые операции, логировать события в базу данных, отправлять уведомления по электронной почте и генерировать статистические отчёты. Инструмент обладает механизмами ротации логов и обеспечивает безопасность доступа к системным ресурсам. Реализация проекта позволила освоить навыки системного программирования в Linux, работы с базами данных, сетевыми протоколами и разработкой графических интерфейсов пользователя.

## 8 Полезные ссылки

Документация для ptrace: <https://python-pttrace.readthedocs.io/en/latest/>

Документация для inotify: <https://docs.kernel.org/filesystems/inotify.html>

Ссылка на репозиторий проекта: [https://github.com/MansurYa/audit\\_of\\_Linux\\_system.git](https://github.com/MansurYa/audit_of_Linux_system.git)