**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

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

**ОТЧЁТ**

**По лабораторной работе №3 По дисциплине “Системное программирование в линукс”**

**На тему “Разработка системного инструмента для аудита системы”**

**Студент гр. 23Б16-пу**

**Пушкарев Н.П.**

**Преподаватель**

**Киямов Ж.У.**

**Санкт-Петербург**

**2024 г.**

[Цель работы 3](#_gjdgxs)

[Описание задачи 3](#_30j0zll)

[Теоретическая часть 3](#_1fob9te)

[Описание программы 5](#_3znysh7)

[Основные шаги программы 6](#_3dy6vkm)

[Рекомендации пользователя 6](#_1t3h5sf)

[Рекомендации программиста 6](#_4d34og8)

[Контрольный пример 6](#_2s8eyo1)

[Вывод 8](#_35nkun2)

[**Исходный код 8**](#)

[Список источников 8](#_1ksv4uv)

# Цель работы

Разработать системный инструмент, который будет использоваться для аудита и мониторинга системы Linux.

# Описание задачи

1. Создать программу, которая будет регистрировать события в системе, такие как запуск и завершение процессов, изменение файлов, сетевые операции и другие значимые действия.

2. Программа должна сохранять информацию о событиях в журнале событий.

3. Реализовать возможность фильтрации и поиска событий в журнале по различным критериям, таким как пользователь, тип события, время и другие.

4. Разработать механизмы оповещения о событиях, такие как отправка уведомлений по электронной почте или внутренних сообщений.

5. Обеспечить возможность создания отчетов на основе данных из журнала событий, включая статистику и графики.

6. Программа должна быть разработана с учетом безопасности и иметь ограниченные привилегии доступа к системным ресурсам.

7. Подготовьте документацию к программе, включая описание ее возможностей и инструкции по установке и использованию.

# Теоретическая часть

Аудит в Linux представляет собой процесс мониторинга и регистрации событий, происходящих в операционной системе. Основная цель аудита — обеспечение безопасности, поддержание интегритета системы, выявление уязвимостей и контроль за соблюдением политики безопасности. Аудит позволяет системным администраторам анализировать поведение системы, выявлять аномалии и предотвращать несанкционированные действия.

Аудит используется для:

1. Регистрации событий, таких как запуск и завершение процессов, изменение файлов, сетевые операции.
2. Выявления уязвимостей в системе и контроля за действиями пользователей.
3. Формирования журналов событий для последующего анализа и поиска закономерностей.
4. Реализации автоматического оповещения о критически важных событиях.
5. Генерации отчетов с графическим представлением статистики для улучшения видимости процессов в системе.

Для реализации аудита в Linux широко используются системные вызовы, такие как ptrace, которые позволяют отслеживать действия процессов на низком уровне. Другие важные инструменты:

* Журналы системы: стандартные логи системы, такие как /var/log/syslog и /var/log/auth.log.
* Файловые системы: позволяют отслеживать операции чтения, записи, создания и удаления файлов.
* Уведомления: инструменты для отправки сообщений пользователям или администраторам.

Для контроля процессов в рамках данной работы используется вызов ptrace, предоставляющий механизмы наблюдения и управления процессами. Он позволяет:

1. Перехватывать системные вызовы, выполняемые отслеживаемым процессом.
2. Считывать регистры процессора для анализа операций.
3. Останавливать выполнение процесса для обработки событий.

Основные функции программы аудита

1. Регистрация событий: запись информации о действиях процессов (например, вызовы системных функций).
2. Журналирование: сохранение событий в лог-файл с указанием времени, идентификатора процесса (PID) и типа события.
3. Фильтрация и поиск: поиск событий по пользователям, времени, типу событий для упрощения анализа.
4. Оповещения: отправка уведомлений администраторам при критически важных событиях.
5. Ротация логов: предотвращение переполнения диска за счет автоматического создания архивов старых журналов.

# Описание программы

Программа написана на C++ и разделена на модули, выполняющие отдельные функции:

исходный код: audit.cpp

Таблица 1. Описание исходного кода

| Название функции | Входные данные | Описание |
| --- | --- | --- |
| check\_log\_size | (const string &log\_file)    На вход получает путь к лог-файлу | проверяет размер лог-файла и архивирует его при превышении допустимого объема |
| current\_time | () | возвращает текущее системное время в формате дд.мм.гггг чч:мм:сс. |
| send\_email | (const string &message)    На вход получает сообщение с уведомлением, которое надо отправить на почту | отправка уведомлений по email |
| write\_system | (unsigned long long system\_code, ofstream &file, int pid)    На вход получает код системного вызова, поток для записи логов, идентификатор отслеживаемого процесса | логирует вызов системной функции |
| write\_command | (const char\* line, ofstream &file, int pid)    На вход получает текст сообщения, поток для записи логов, идентификатор отслеживаемого процесса | записывает произвольное сообщение в лог с указанием PID и времени |
| main | (int argc, char\*\* argv)  На вход получает PID | основная функция программы, запускающая процесс аудита |

# 

# Основные шаги программы

1. Чтение конфигурации из backup\_config.ini, содержащей sourceDir, backupDir и backupFrequency.
2. Переход демона в фоновый режим.
3. Запуск цикла, который:
   1. Ожидает установленный интервал времени.
   2. Копирует файлы с временной меткой.
   3. Записывает данные о копировании в системный журнал.
4. Остановка демона при получении сигнала.

# Рекомендации пользователя

Для запуска программы:

1. Скомпилируйте программу командой g++ audit.cpp -o audit -std=c++17.
2. Запустите программу, передав ей PID одним из двух способов
   1. ./audit PID
   2. ./audit $(pidof -s <name>)
3. Откройте logs.log и проанализируйте работу выбранного процесса
4. Проверьте уведомление о работе программы указанной вами почте

# Рекомендации программиста

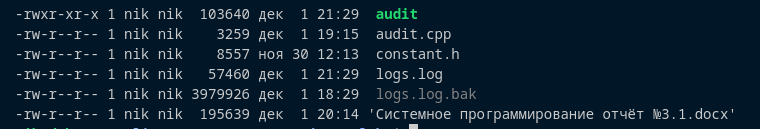
* Следите за актуальностью используемых библиотек
* Очищайте лог, когда это необходимо

# Контрольный пример

Пример файла компиляции и запуска программы для системного процесса Xwayland:

Рис1. Пример запуска

Пример рабочей директории программы, для понимания того, какие файлы для чего нужны.

Рис2. Пример рабочей директории

Пример constant.h, содержащего номера и названия системных вызовов.

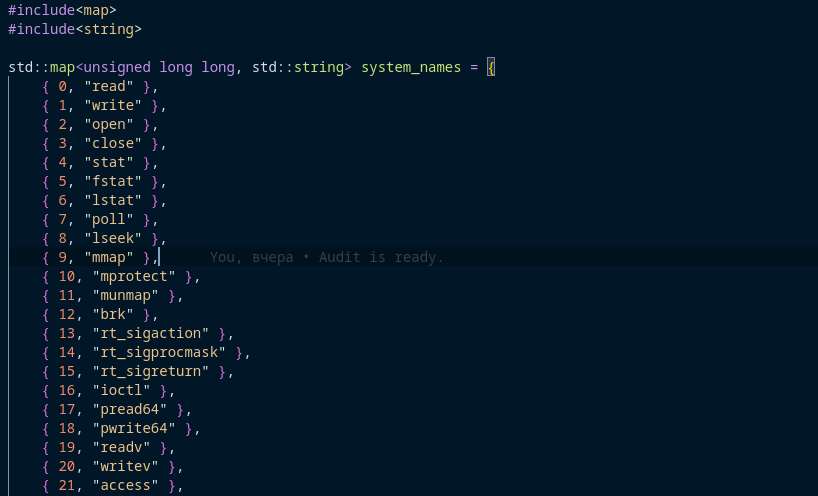


Рис3. Пример constant.h

Пример логов работы выбранного процесса.

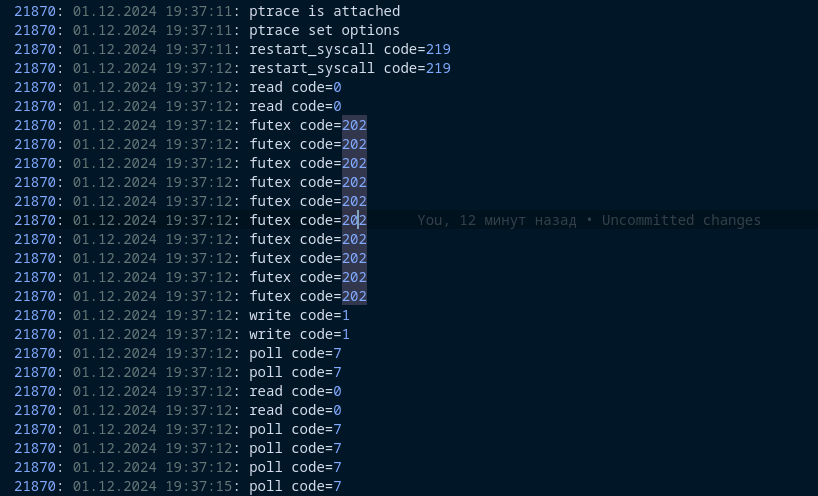


Рис4. Пример logs.log

# Вывод

В результате выполнения работы была разработана программа-аудит для мониторинга системных событий в Linux. Также были получены все необходимые сведения о разработке подобных программ.

# Исходный код

Посмотреть все используемые файлы можно на GitHub.  
https://github.com/nikitopus/functional\_programming\_24/tree/master/3\_laba

# Список источников

* Man-страницы Linux (man ptrace, man syscalls, man syslog)
* Интернет-источник: Форум Хабр для разработчиков: https://habr.com/ru/companies/selectel/articles/267833/