Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Лабораторная работа №3**

**Отчет**

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

**Разработка системного инструмента для аудита системы**

Автор работы: Дацык Р.В.

Группа: 22.Б15-пу

Преподаватель: Киямов Ж.У.

Санкт-Петербург, 2023

**Оглавление**

[**1.** **Цель работы** 3](#_Toc151824592)

[**2.** **Задачи** 3](#_Toc151824593)

[**3.** **Введение** 3](#_Toc151824594)

[**4.** **Теоретическая часть** 3](#_Toc151824595)

[**5.** **Алгоритм метода** 5](#_Toc151824596)

[**6.** **Описание программы** 5](#_Toc151824597)

[**7.** **Рекомендации пользователю** 7](#_Toc151824598)

[**8.** **Рекомендации программисту** 7](#_Toc151824599)

[**9.** **Контрольный пример** 8](#_Toc151824600)

[**10.** **Вывод** 9](#_Toc151824601)

[**11.** **Список использованной литературы** 9](#_Toc151824602)

# **Цель работы**

Целью лабораторной работы является создание функционального инструмента системного аудита, который позволяет пользователям отслеживать и анализировать системные вызовы в произвольном процессе.

# **Задачи**

* Разработка программы для присоединения к целевому процессу.
* Отслеживание системных вызовов целевого процесса.
* Регистрация информации о системных вызовах.
* Вывод полученной информации для анализа.

# **Введение**

В рамках лабораторной работы разрабатывался инструмент системного аудита, предназначенный для отслеживания системных вызовов в заданном процессе. Системный аудит является важным инструментом для анализа поведения программного обеспечения в реальном времени и выявления проблем, таких как бесконечные циклы, зависания и другие аномалии.

# **Теоретическая часть**

Системные вызовы представляют собой ключевой момент взаимодействия между пользовательским пространством и операционной системой. Когда приложение нуждается в выполнении какой-либо операции, такой как чтение или запись в файл, выделение памяти или создание нового процесса, оно делегирует выполнение этой задачи операционной системе через системные вызовы. Эти вызовы обеспечивают абстракцию аппаратных ресурсов и ядра операционной системы, предоставляя удобный интерфейс для программ.

Отслеживание системных вызовов имеет решающее значение при анализе поведения программы. Это позволяет выявлять аномальное или неожиданное поведение, обнаруживать утечки ресурсов, определять причины зависаний и повышать уровень безопасности приложений.

# **Алгоритм метода**

1. **Подключение к процессу:**
   1. Получение идентификатора целевого процесса от пользователя.
   2. Подключение к процессу с использованием системных вызовов, предоставляемых операционной системой.
2. **Установка обработчика системных вызовов:**
   1. Регистрация пользовательской функции в качестве обработчика системных вызовов с использованием соответствующего механизма операционной системы.
3. **Запуск цикла отслеживания:**
   1. Создание бесконечного цикла, в котором осуществляется ожидание системных вызовов в целевом процессе.
   2. При возникновении системного вызова фиксирование соответствующей информации, такой как идентификатор вызова, аргументы и время вызова.
4. **Вывод информации:**
   1. Вывод собранной информации в удобном формате, например, в консоль или в файл, для последующего анализа.
5. **Отключение и завершение:**
   1. По завершении отслеживания освобождение ресурсов и отключение от целевого процесса.
   2. Вывод завершающего сообщения.

# **Описание программы**

Программа реализована на языке C++ и использует функционал, предоставляемый операционной системой для отслеживания системных вызовов. Вывод программы содержит информацию о вызываемых системных функциях, времени их вызова и другие сопутствующие данные.

Таблица 6.1. Описание функций

|  |  |
| --- | --- |
| Название функции | Описание функции |
| attachToProcess(int pid) | Функция отвечает за подключение к целевому процессу с использованием идентификатора процесса (PID). |
| setSyscallListener() | Функция устанавливает обработчик системных вызовов для целевого процесса. |
| syscallHandler(int syscallNumber, ...) | Это обработчик системных вызовов, вызываемый при каждом обращении целевого процесса к системным функциям. |
| startMonitoring() | Функция инициирует цикл отслеживания системных вызовов. |
| stopMonitoring() | Функция завершает мониторинг системных вызовов и отключается от целевого процесса. |
| outputInformation(...) | Функция выводит собранную информацию о системных вызовах в удобном формате. |
| main(...) | Это основная функция программы, где происходит последовательный вызов остальных функций для выполнения программы. |

# **Рекомендации пользователю**

Для успешного запуска программы необходимо устройство с операционной системой Linux, macOS или Windows, а также среда разработки, поддерживающей запуск компилятора С++

Для запуска программы скомпилируйте исходный код с помощью команды g++ -o system\_audit\_tool system\_audit\_tool.cpp -std=c++11. С помощью команды pidof <имя\_программы>, узнайте PID интересующей вас прораммы. Затем с помощью команды sudo ./ system\_audit\_tool <PID> запустите программу. Файл с выводом программы audit.log будет сохранен в директорию программы

# **Рекомендации программисту**

Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с компилятором С++ и библиотеки <iostream>, <ctime>, <cstdlib> и другие.

Исходный код программы и доступен по ссылке:

<https://github.com/CapTopGrade/Linux/blob/main/Audit/system_audit_tool.cpp>

# **Контрольный пример**

В данном разделе представлен контрольный пример, демонстрирующий работу программы.

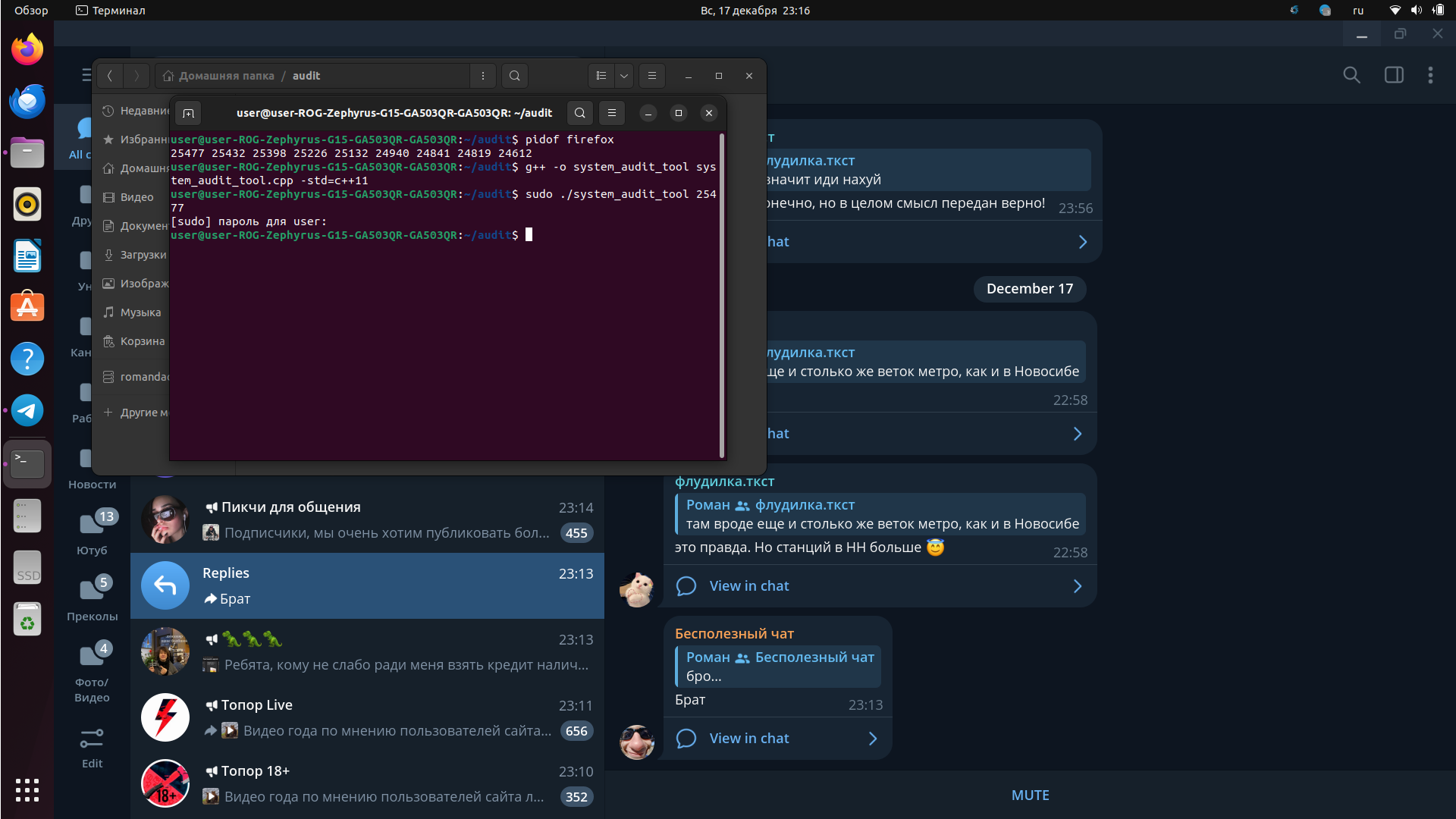


Рис. 9.1 Демонстрация запуска программы

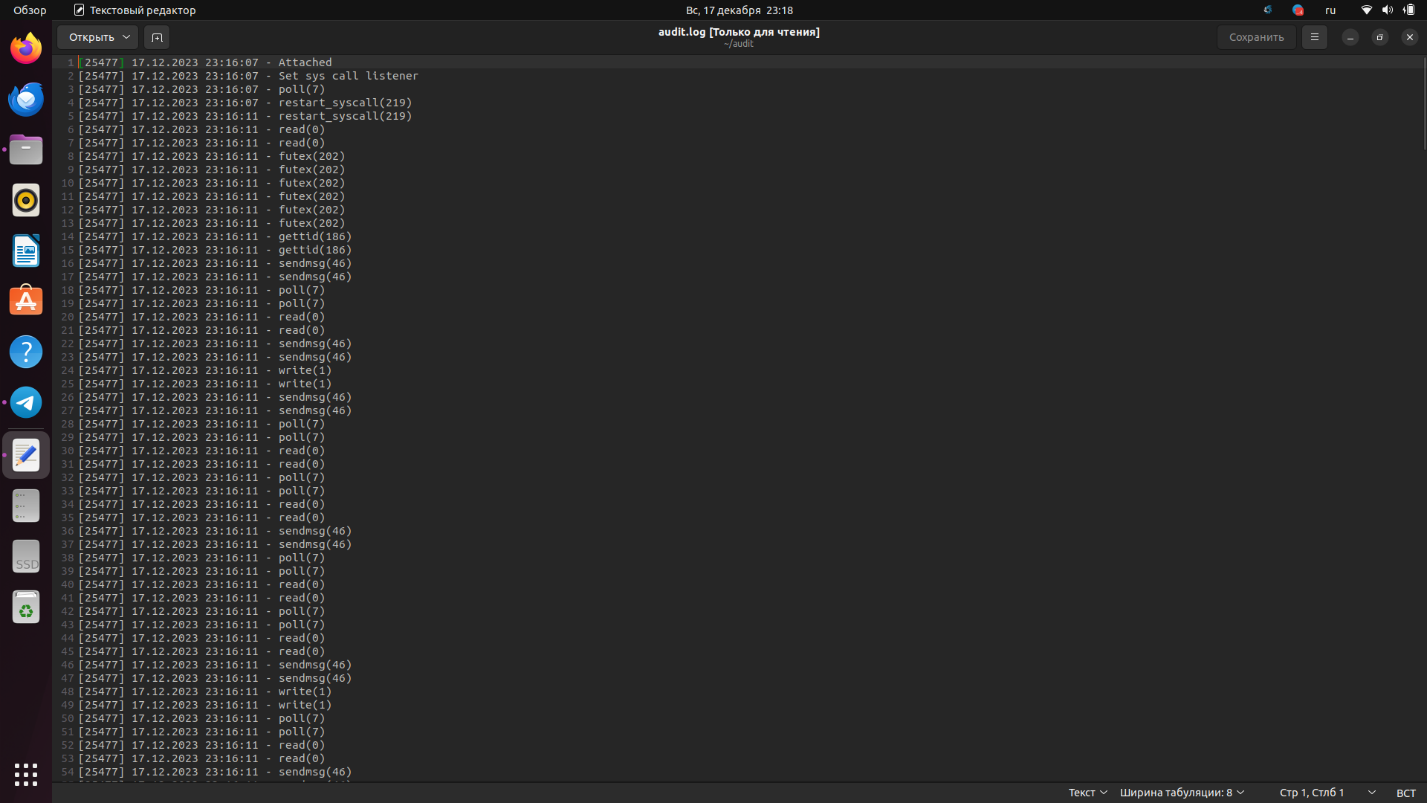


Рис. 9.2 Пример работы программы

# **Вывод**

Лабораторная работа позволила разработать инструмент системного аудита, который успешно отслеживает системные вызовы в выбранном процессе. Полученный инструмент может быть полезен для анализа работы приложений, выявления проблем и оптимизации кода. Отслеживание системных вызовов - важный элемент в арсенале инструментов разработчика и системного администратора.

# **Список использованной литературы**

[1] Документация по работе с С++: <https://ru.cppreference.com/w/>

[2] Рекомендации по улучшению кода: <https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md>