МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по дисциплине

Технология разработки программного обеспечения

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чернобаев И.Д,\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Седов А.М

(подпись) (фамилия, и.,о.)

\_\_\_М24-ИВТ-4\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2024

**Задание:**

**Инструменты**: OS Linux, ReactiveExtensions (для любого удобного языка).

***Для реализации было выбрано два языка программирования – С++ и Ruby***

Реализовать трекер событий клавиатуры (какая клавиша, какой событие с клавишей произошло).

Трекер запустить в отдельном потоке.

Реализовать паттерн "Наблюдатель + Итератор":

* Каждое событие трекер передает всем подписанным "слушателям" (on\_next).
* Подписчики должны отобразить полученную информацию в файл.
* Реализовать завершение работы трекера по вводу определенной комбинации клавиш (on\_complete).
* Реализовать обработку ошибок (on\_error).

**Ход работы**

***Замечание****.* *В отчете рассматриваются основные моменты и используемые технологии, связанные исключительно с реализацией на языке C++. Язык Ruby был использован дополнительно для демонстрации альтернативной реализации на более абстрактном уровне. Ruby, в отличие от C++, ориентирован на упрощение разработки и автоматизацию многих задач, что позволяет быстрее создавать прототипы и работать с высокоуровневыми концепциями.*

**Использованные технологии**

**Reactive Extensions (RxCpp)**

RxCpp — это библиотека для реактивного программирования в C++. Она позволяет создавать асинхронные потоки данных и управлять ими. В данном приложении RxCpp используется для реализации паттерна 'Наблюдатель', где подписчики получают события клавиш в режиме реального времени.

**XKB (libxkbcommon)**

libxkbcommon — библиотека для работы с раскладками клавиатуры в среде X11. Она используется для преобразования кодов клавиш (keycodes) в соответствующие символы с учетом текущей раскладки клавиатуры.

**Компоненты приложения**

1. Основной класс: KeyboardTracker

Класс KeyboardTracker реализует функции трекера клавиш

- running (std::atomic<bool>): Флаг, определяющий состояние работы трекера

- logFile (std::ofstream): Лог-файл для записи событий клавиш

- eventSubject (rxcpp::subjects::subject): Реактивный источник событий, передающий их подписчикам

- context, keymap, state (libxkbcommon): Контекст XKB для преобразования кодов клавиш в символы

Основные методы класса:

- logEvent(const std::string&): Логирует событие в файл и отправляет его подписчикам

- getKeySymbol(uint16\_t): Преобразует код клавиши в символ с учетом раскладки

- trackEvents(): Запускает трекинг событий клавиатуры в отдельном потоке

- start(): Инициализирует поток для обработки событий

- stop(): Завершает работу трекера

Основная функция: main()

Функция main() создает экземпляр KeyboardTracker, запускает его и обрабатывает события через подписчиков.

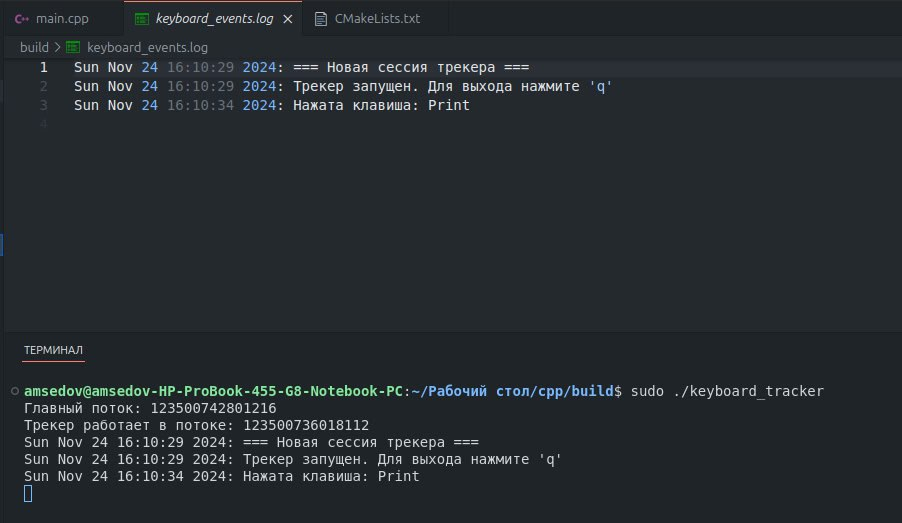
Основные этапы выполнения:

- Создание подписчика, который логирует события в консоль и фай

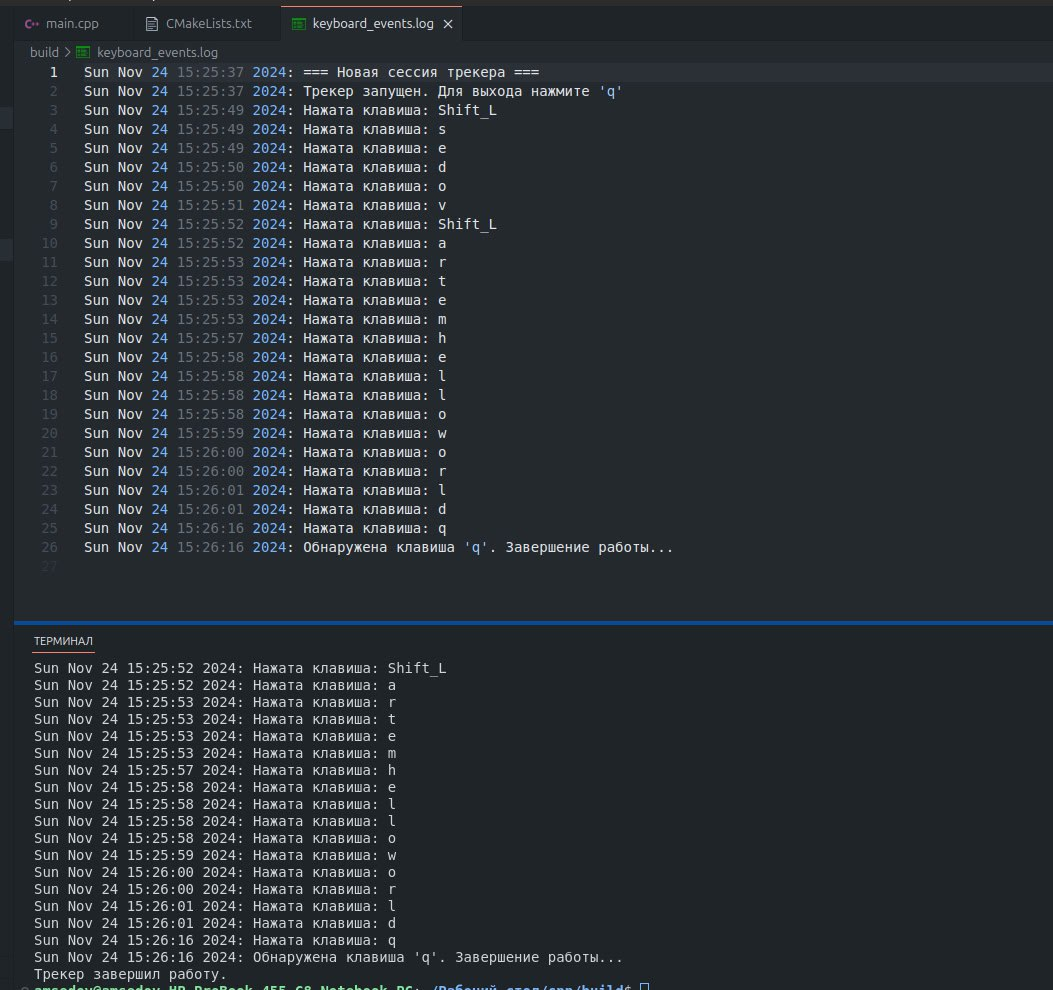
- Запуск трекера

- Поддержание работы приложения до завершения трекера (нажатие «q»)

**Результат и анализ работы:**

После запуска программы в лог помещается информация о начале работы новой сессии трекера, в консоль так же выводятся id потоков:

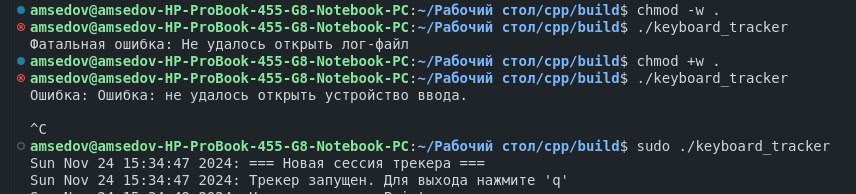
После запуска программы она работает в фоновом режиме и обрабатывает все взаимодействия с клавиатурой:



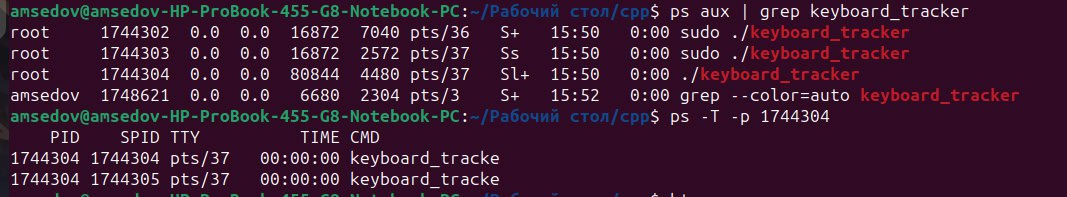
После нажатия “q” трекер завершил работу.

Проверим реализацию работы обработки ошибок для этого намеренно ограничим директорию в которой находится лог-файл в правах с помощью команды «chmod -w .»

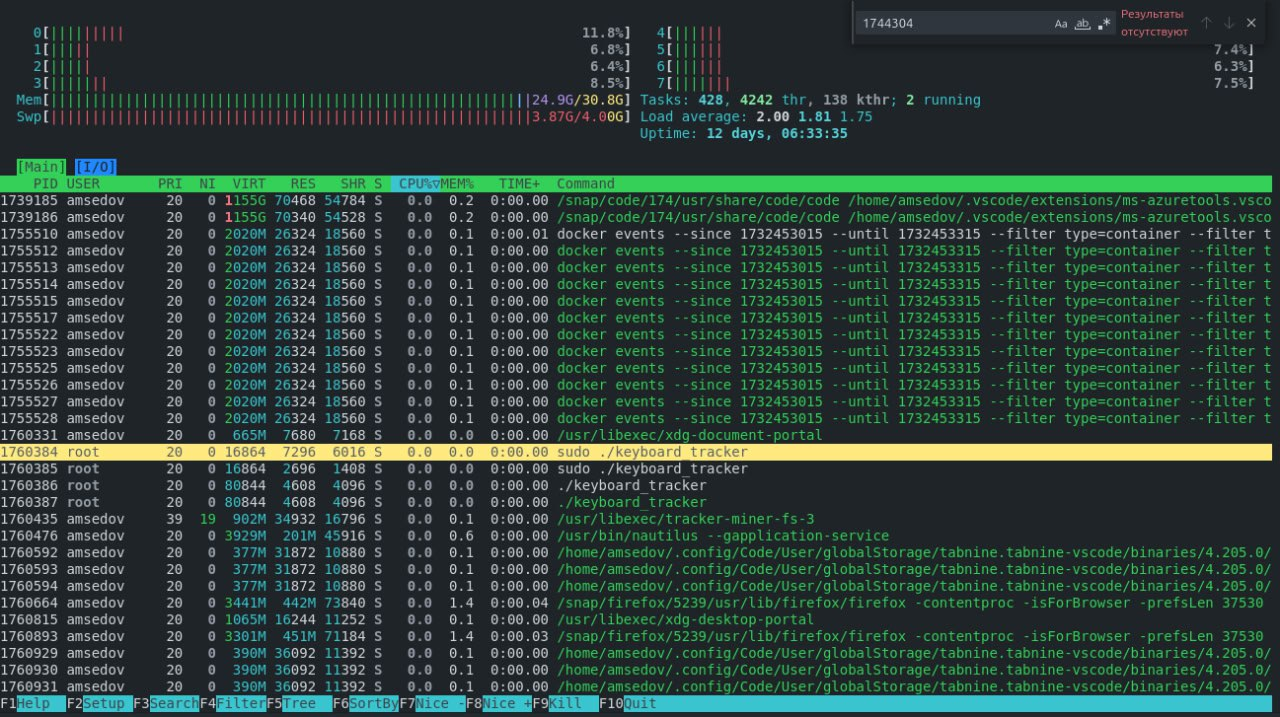
Также мы попробуем запустить нашу программу без прав администратора, чтобы убедиться, что она не начнет выполнение при отсутствии доступа к клавиатуре (в Linux доступ к отслеживанию клавиатуры возможен только в режиме суперпользователя — SU).



При помощи утилит grep и htop проведем анализ потоков при запущенной программе:

**Grep:**

**Htop:**



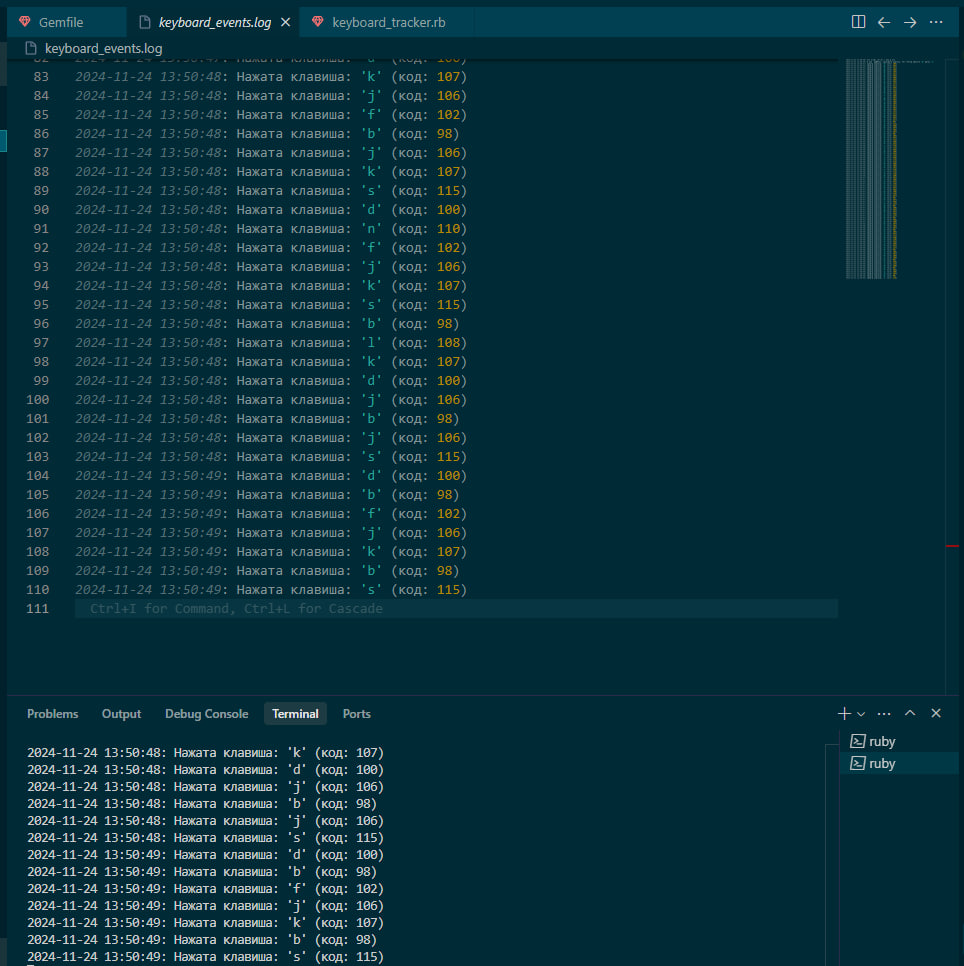
Как видно из вывода у нас есть два потока – Основной и поток трекера

Убедимся, что программа функционирует корректно, и проверим возможность ее завершения с помощью стандартного сочетания клавиш в Linux — Ctrl+C:

  
  
Как видно из лога, программа не завершилась и продолжила работу ожидая “q”

Так же в репозитории можно найти cpp/keyboard\_events.log - Проверка всех клавиш

**Результат работы программы на Ruby**



**Вывод:** В ходе лабораторной работы было написано две программы на языках C++ и Ruby, обе программы реализуют трекер событий клавиатуры. Для выполнения задачи были использованы операционная система Linux и библиотека ReactiveExtensions для реализации паттерна "Наблюдатель + Итератор". Обе программы обрабатывают события клавиатуры, фиксируя, какая клавиша была нажата и какое событие с ней произошло.Программа на C++ предоставляет низкоуровневую реализацию с более глубоким управлением системными ресурсами, а версия на Ruby демонстрирует аналогичную функциональность на более высоком уровне абстракции.

**Приложение:  
https://github.com/arsedov2019/LR2\_sedov/tree/main**