Оконный менеджер на Rust

Проскуряков Дмитрий Андреевич
Научный руководитель: Воронцов Илья Евгеньевич

Общее описание работы

Написан оконный менеджер на языке Rust(язык общего назначения активно набирающий популярность, известный своей безопасностью и строгой типизацией) для оконной системы X Window System для операционных систем семейства Unix.

Данный оконный менеджер вдохновлен другими проектами, такими как DWM, XMonad, позволяющими управлять окнами в графическом рабочем пространстве пользователя. Оконный менеджер позволяет гибко настраивать поведение окон на рабочем столе посредством редактирования исходного кода на типобезопасном языке.

Оконный менеджер состоит из нескольких компонентов таких как:
1) Хранилище конфигураций окон
2) Обработчик событий
3) Менеджер сочетаний клавиш

Реализована базовая работа с окнами, есть поддержка нескольких мониторов и некоторых расширений для работы с статус панелями. Предусмотрено создание сочетаний клавиш для быстрого запуска приложений и выполнения действий по типу изменения громкости итд.

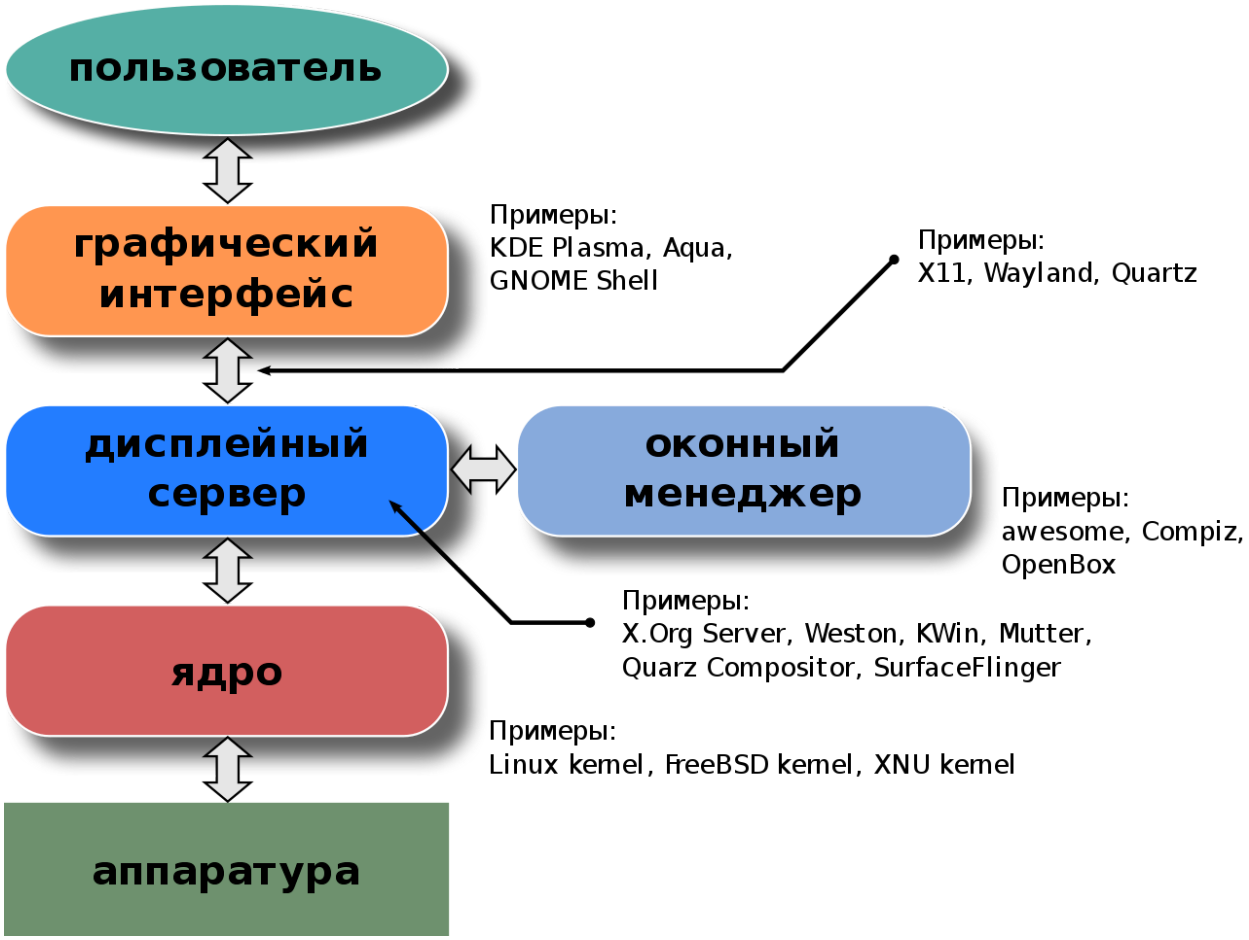
Оконный менеджер доступен на гитхабе: https://github.com/justdprroz/rtde

Легенда:

WM (Window Manager) - оконный менеджер

Введение

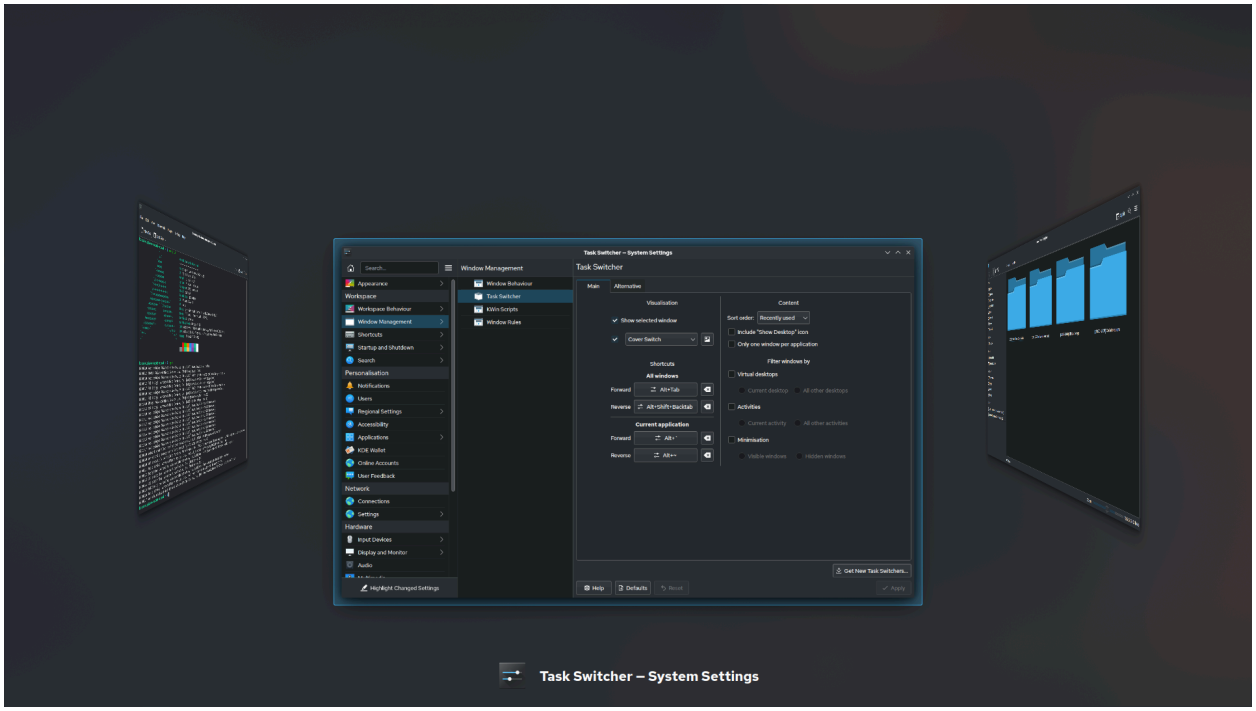
Оконный менеджер - программа отвечающая за расположение и управление окнами в системах графического интерфейса.



Среди популярных оконных менеджеров можно выделить:

- KWin - встроенный оконный менеджер окружения KDE Plasma Desktop для ОС Linux
- Mutter - оконный менеджер GNOME также для Linux
- Quartz - оконный менеджер MacOS

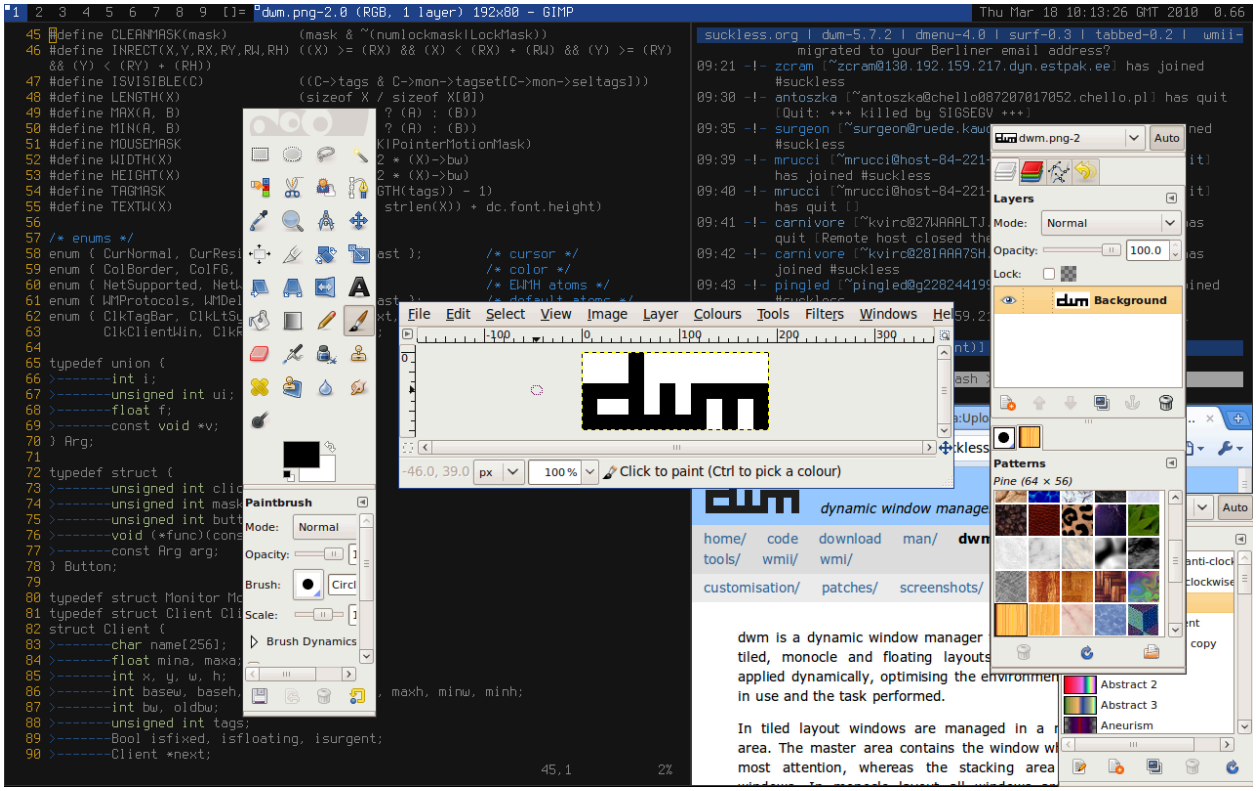
Пример переключения окон в KWin



Вышеописанные оконные менеджеры являются частью и поставляются в составе соответствующих окружений/операционных систем

Существует несколько независимых оконных менеджеров: Awesome WM, dwm, xmonad, QTile, OpenBox. Данные оконные менеджеры являются тайловым - то есть организация окон происходит “плитками“ в одной плоскости рабочего пространства(workspace)

Демо DWM



Оконный менеджер содержит компоненты ответственные за

1. Ввод и вывод пользователя: движения мыши, сочетания клавиш(иногда) и другие;
2. Обработку событий: создание/уничтожение окон, изменение параметров окон, изменение настроек периферии (изменение конфигурации мониторов итд)
    2.1. Использование сообщений(запросы к оконному менеджеру на: изменение размера, состояния полноэкранного режима итд), от клиентов для изменения настроек
3. Хранение информации о текущих окнах (список окон, их местоположения, состояния, заголовки итд), управление окнами в соответствии выбранными/предусмотренными сценариями.

Задача

Среди тайловых оконных менеджеров есть множество менеджеров выполняющих практически одинаковый функционал но имеющих разные реализации использующие разные языки программирования. К примеру: DWM, AwesomeWM - C, XMonad - Haskell, QTile - Python.

Одно из основных отличий для пользователя помимо требуемого им функционала - способ конфигурации оконного менеджера. Некоторые (DWM, XMonad) редактируются путем изменения исходного кода, другие же (i3, Awesome, QTile) используют конфигурационные файлы в виде скриптов или настроек использующих другие языки (i3 - свой конфигурационный язык, Awesome использует Lua скрипты, QTile - Python).

У обоих подходов есть свои минусы и плюсы:

Изменение исходного кода:
Минусы
- Требуется знание языка программирования
- Сложности обновления из-за конфликтов изменений в ядре оконного менеджера с вашими личными модификациями.
- Возможность сломать оконный менеджер допустив ошибку
- Требуется наличие исходного кода для конфигурации

Плюсы
- Производительность
- Гибкость настройки

Конфигурационные файлы:
Минусы
- Ограниченный функционал если конфигурация выполняется используя простые средства
- Требования языка разметки/конфигурации для достижения требуемого функционала(Пример - i3)

Плюсы
- Портативность - вместо всего кода требуется хранить только конфигурационный файл
- Простота в использовании

Среди оконных менеджеров которые я использовал были выявлены следующие недостатки:

1. DWM - Конфигурация производится на Си что несет за собой множество проблем в лице: небезопасности кода который был написал во время модификации и кастомизации, сложности в понимании оригинального кода DWM для его дальнейшего улучшения, сложности в модификации оконного менеджера путем

патчинга (модификация DWM производится применением патчей patches, что становится невозможным в силу конфликтов спустя несколько модификаций)
2. XMonad - Требуется знание Haskell, что может стать тупиком в его конфигурации для человека привыкшего к языкам с Си-подобным синтаксисом.
3. I3 - недостаточно гибкая настройка в силу усложненных методов конфигураций

Взяв во внимание вышеперечисленные проблемы мной был выбран путь написания своего оконного менеджера.

Методы

Для написания моего оконного менеджера был выбран язык Rust и оконная система X11 https://ru.wikipedia.org/wiki/X_Window_System. За основу был взят DWM - выбор пал на него в силу его простоты, элегантности и функционального минимализма, что позволяет написать оконный менеджер концентрируясь на базовых и необходимых вещах.

Язык Rust был выбран по нескольких причинам:
1) Я начал изучать Rust (примерно за год до написания оконного менеджера) и написание оконного менеджера выглядело перспективным способом выучить некоторые тонкости языка
2) Rust - стремительно развивающийся язык, он занимается лидирующую позицию “Самого любимого” языка программирования следуя опросу “StackOverflow Survey”.
3) Rust - является безопасным языком со строгой статической типизацией, что делает его предпочтительным для разработки проектов требующих устойчивости и простоты в исправлении ошибок
4) Масштабируемость проектов на Rust достигается средствами сильной типизации и проектного менеджера cargo
5) Компилируемый язык позволит получать исполняемый файл оконного менеджера содержащий исключительно требуемые пользователю функции и сценарии использования

Оконная система X11 была выбрана в силу большого количества документации и готовых проектов использующих ее. Альтернатива X11 - Wayland несмотря на стремительное развитие не нашла явных преимуществ над X11

Идеи и принципы оконного менеджера:
- Минимализм - хотелось реализовать оконный менеджер с поддержкой базового функционала и возможностью включения других опциональных расширений. (Философия DWM и сообщества suckless)
- Расширяемость - оконный менеджер должен обладать API которое поможет в дальнейшей разработке и кастомизации оконного менеджера

Архитектура:

Оконный менеджер состоит из следующих компонентов:

- Хранилище конфигураций окон
- Обработчик событий о создании/закрытии окон и изменении их параметров
- Менеджер сочетаний клавиш
- Система автозапуска приложений и конфигурации
- Обработчик изменения конфигурации мониторов
- Система логирования

Пример:
При запуске оконного менеджера выполняется файл **~.rtde/autostart.sh** выполняющий заданные пользователем команды(запуск приложений, конфигурация мониторов, раскладки клавиатуры и другие базовые настройки пользовательской сессии)

Например, при нажатии на кнопку закрытия окна, сообщение об этом попадает в обработчик событий. После чего оконный менеджер просит приложение закрыться так как ему требуется. При закрытии, окно сообщается оконному менеджеру что она закрылось и он перемещает окна на освободившееся место.

На данный момент средства для кастомизации не реализованы так как оконный менеджер находится в разработке. В планах реализация должна происходить посредством использования API оконного менеджера на языке Rust.

Результаты и дискуссия

Реализован функционал немного уступающий DWM.
Присутствует поддержка:

- Нескольких мониторов и рабочих пространств (workspace - каждое окно привязано к своему рабочему пространству)
- Расположение окон плитками в виде стека
- Частичная поддержка EWMH (расширенные флаги для работы с оконным менеджером и приложениями, см. документацию) и Polybar (GitHub - polybar/polybar: A fast and easy-to-use status bar)
- Создания сочетаний клавиш (“Actions”) использующихся для запуска команд. К примеру: изменение громкости, управление медиа, запуск приложений
- Поддержка dmenu как лаунчера приложений
- Автозапуск приложений

На данный момент отсутствует поддержка:

- Некоторых клиентских сообщений(изменение текущего рабочего стола с помощью polybar и EWMH и другие) (будет добавлено позже по мере надобности)
- Поддержка мыши для управления окнами (перемещение, изменение размеров итд не предусмотрено)
- Корректной работы некоторых приложений(данная проблема свойственна многим оконным менеджерам по типу DWM, XMonad итд). В силу минимальности

реализации оконного менеджера, приложения по типу Telegram, Spectacle, Obs, использующие нишевые возможности X11, работают некорректно - пропадает возможность взаимодействия с окном, окно отображается в виде черного квадрата, приложение может некорректно завершить работу.

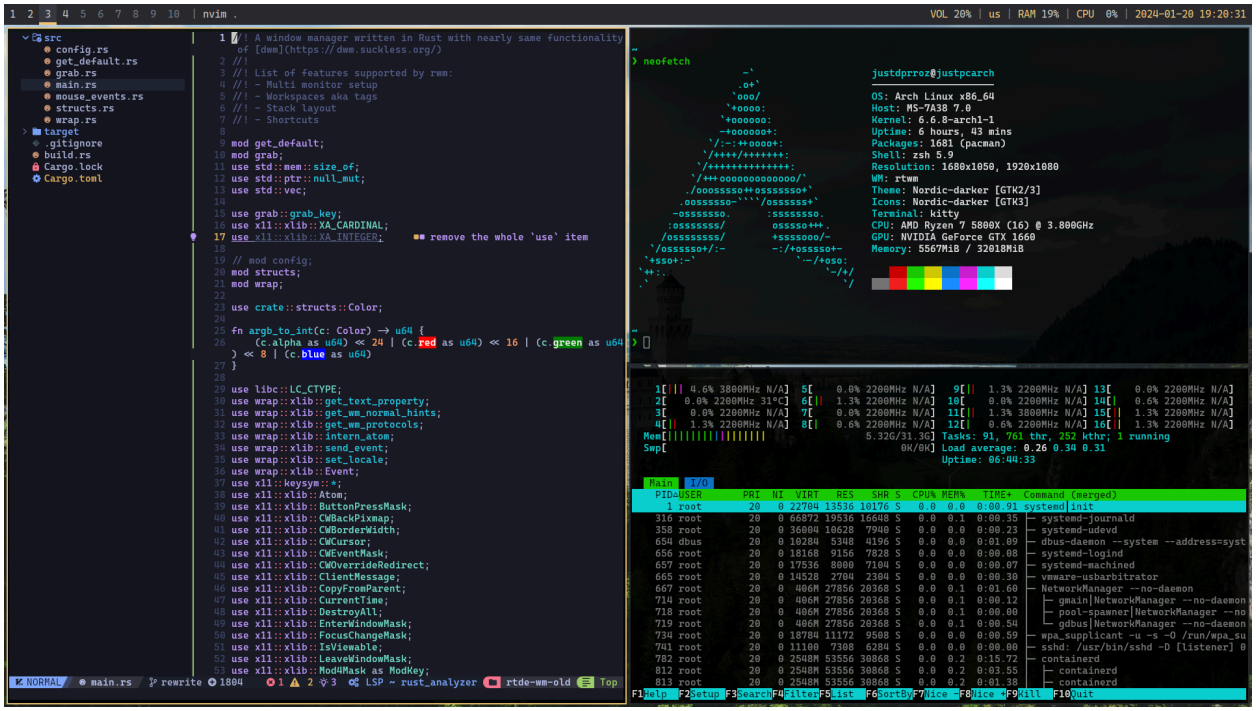
Планы на ближайшее будущее:

- Поддержка конфигурации путем использования безопасного API
- Расширенная поддержка приложений путем добавления большего количества EWMH
- Поддержка мыши

Чтобы опробовать оконный менеджер:
1) Скачайте репозиторий https://github.com/justdprroz/rtde
2) Следуйте инструкции https://github.com/justdprroz/rtde/blob/master/README_RU.md

Демо:

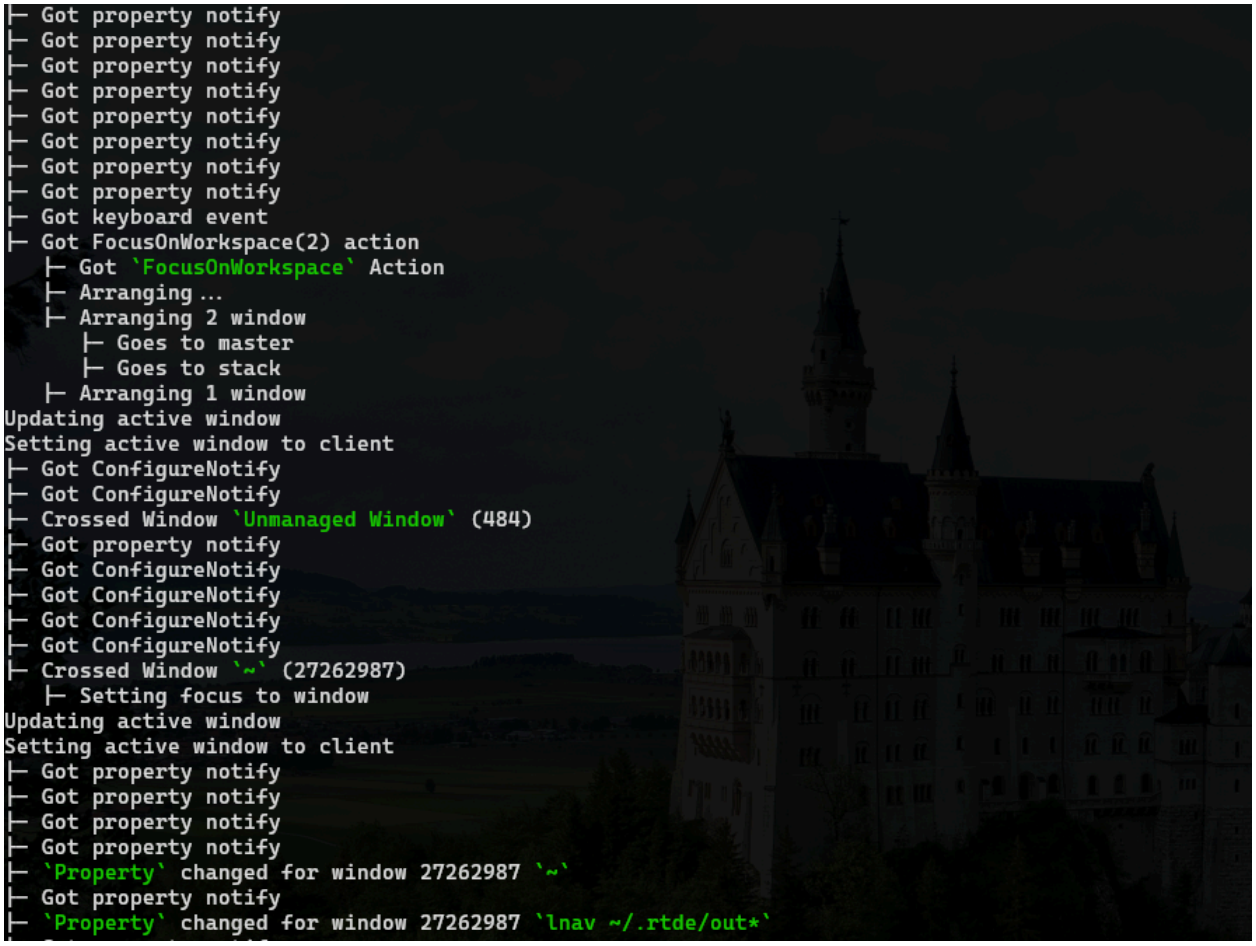
Стековая раскладка:



Конфигурация сочетаний клавиш в исходном коде

```
        KeyAction {
            modifier: ModKey,
            keysym: XK_Return,
            result: ActionResult::Spawn("kitty".to_string()),
        },
        KeyAction {
            modifier: ModKey,
            keysym: XK_e,
            result: ActionResult::Spawn("thunar".to_string()),
        },
        KeyAction {
            modifier: ModKey,
            keysym: XK_p,
            result: ActionResult::Spawn("dmenu_run".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioRaiseVolume,
            result: ActionResult::Spawn("/usr/local/bin/volumeup".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioLowerVolume,
            result: ActionResult::Spawn("/usr/local/bin/volumedown".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioMute,
            result: ActionResult::Spawn("/usr/local/bin/volumemute".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioPlay,
            result: ActionResult::Spawn("playerctl play-pause".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioNext,
            result: ActionResult::Spawn("playerctl next".to_string()),
        },
        KeyAction {
            modifier: 0,
            keysym: XF86XK_AudioPrev,
            result: ActionResult::Spawn("playerctl previous".to_string()),
        },
```

Система логирования для дебага и помощи в разработке



Список литературы

https://dwm.suckless.org/ - Оконный менеджер которым я вдохновлялся
https://github.com/polybar/polybar - Статус панель
https://tronche.com/gui/x/xlib/ - Туториал по работе с xlib (реализация X11 на языке Си)
https://refspecs.linuxbase.org/LSB_3.1.0/LSB-Desktop-generic/LSB-Desktop-generic/libx11-ddefs.html - Референс некоторых переменных xlib
https://specifications.freedesktop.org/wm-spec/wm-spec-1.3.html - Документация по EWMH