Текстовой редактор emacs

Лабораторная работа №11

Козомазов Владимир Романович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Комплексное изучение текстового редактора Emacs как многофункциональной программной среды, включая анализ его архитектуры, функциональных возможностей, особенностей использования и перспектив развития в современных условиях.

2 Задание

1. Познакомиться с интерфейсом и режимами работы редактора vi (командный режим, режим вставки, режим последней строки).

2. Освоить основные команды для навигации, редактирования, копирования, удаления и поиска текста.

3. Научиться сохранять изменения и выходить из редактора.

4. Рассмотреть дополнительные возможности (работа с несколькими файлами, настройки, использование плагинов в vim).

5. Закрепить навыки на практике, создавая и редактируя текстовые файлы.

3 Теоретическое введение

3.0.1 Теоретическое введение: текстовый редактор Emacs

3.0.1.1 1. Исторический контекст и философские основы

Emacs (от англ. *Editor MACroS*) был создан в 1976 году Ричардом Столлманом как часть проекта GNU, став воплощением принципов свободного ПО. Его развитие отражает ключевые тенденции в эволюции: - От макроредактора TECO к самостоятельной системе - Формирование концепции “редактора как операционной среды” - Влияние идей Lisp-машины Symbolics на архитектуру

3.0.1.2 2. Архитектурные особенности

Ядро системы построено на: - Интерпретаторе Emacs Lisp (реализация диалекта Lisp) - Модели “буфер-окно-фрейм” для организации workspace - Цикле обработки событий (event loop)

Ключевые компоненты: 1. **Буферы** - виртуальные пространства для работы с данными 2. **Минибуфер** - интерактивная командная строка 3. **Режимы** (major/minor) - контекстно-зависимые поведения

3.0.1.3 3. Парадигма расширяемости

Emacs реализует уникальный подход: - **Самодокументируемость** (встроенная help-система) - **Рефлексивность** (модификация во время выполнения) - **Принцип “всё есть буфер”** (унификация интерфейсов)

3.0.1.4 4. Технические характеристики

- **Язык реализации**: C (ядро) + Emacs Lisp (98% кодовой базы)
- **Модель исполнения**: Гибридная (интерпретация + native-компиляция)
- **Протоколы взаимодействия**:
  - D-Bus (системная интеграция)
  - JSON-RPC (для LSP)
  - TCP/IP (удалённое управление)

3.0.1.5 5. Современное состояние

В версии 29+ появились: - Встроенная поддержка Tree-sitter - Нативная компиляция Elisp - Графические улучшения (Harfbuzz, XWidgets) - Интеграция с системными сервисами (например, Portal для Flatpak)

3.0.1.6 6. Теоретическая значимость

Emacs представляет интерес для исследований в областях: - **Интерфейсов человек-машина** (модель модальности) - **Языково-ориентированного программирования** (DSL через Elisp) - **Эргономики разработки** (анализ workflow экспертов)

3.0.1.7 7. Сравнительный контекст

В отличие от: - **Vim** - акцент на модальности и композиции команд - **VSCode** - модульность через Web-технологии - **Sublime Text** - закрытая проприетарная модель

Emacs предлагает: Полный контроль над средой Глубокую семантическую интеграцию инструментов Парадигму “редактор как виртуальная машина Lisp”

3.0.1.8 8. Перспективные направления

Теоретический анализ выявляет потенциал: - Применения ML для интеллектуального автодополнения - Развития распределённых редакторских сред - Интеграции с новыми hardware-интерфейсами

Данный теоретический базис позволяет перейти к детальному анализу функциональных возможностей и практических аспектов использования Emacs в современных условиях.

4 Выполнение лабораторной работы

Установил менеджер паролей pass с помощью команды `sudo dnf install pass pass-otp` (рис. 4.1).

Рис. 4.1: Установка менеджера паролей

Завершил установку менеджера паролей командой `sudo dnf install gopass` (рис. 4.2).

Рис. 4.2: Завершение установки менеджера паролей

Просмотрел список gpg ключей при помощи команды `gpg --list-secret-keys` (рис. 4.3).

Рис. 4.3: Просмотр списка ключей

Инициализировал хранилище, написав команду ‘pass init (рис. 4.4).

Рис. 4.4: Инициализация хранилища

Создал структуру с git командой `pass git init` (рис. 4.5).

Рис. 4.5: Создание структуры

Синхронизировался с git командами `pass git pull`, `pass git push`
Вручную закоммитил и выложил изменения командами (рис. 4.6).

Рис. 4.6: Ручное выкладывание изменений

Проверил статус синхронизации командой `pass git status` (рис. 4.7).

Рис. 4.7: Проверка статуса синхронизаций

Добавил новый пароль командой `pass insert [OPTIONAL DIR]/[FILENAME]` (рис. 4.8).

Рис. 4.8: Добавление нового пароля

Отобразил пароль для указанного имени файла `pass [OPTIONAL DIR]/[FILENAME]`
Установил дополнительное программное обеспечение (рис. 4.9).

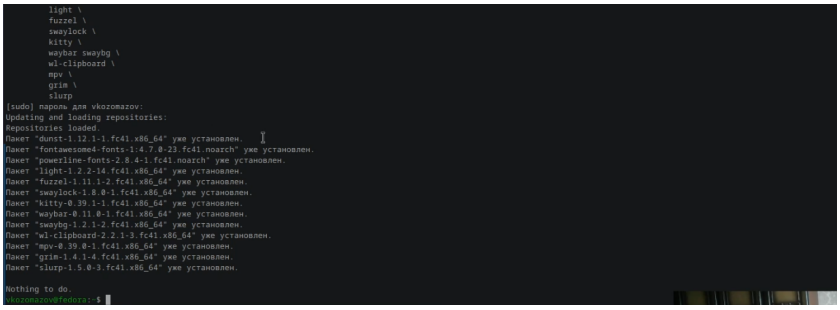

Рис. 4.9: Установка дополнительного программного обеспечения

Установил дополнительно шрифты командами `sudo dnf copr enable peterwu/iosevka`, `sudo dnf search iosevka`, `sudo dnf install iosevka-fonts iosevka-aile-fonts iosevka-curly-fonts iosevka-slab-fonts iosevka-etoile-fonts iosevka-term-fonts` (рис. 4.10).

Рис. 4.10: Установка дополнительных шрифтов

Установил бинарный файл командой `sh -c "$(wget -qO- chezmoi.io/get)"`

Создал собственный репозиторий при помощи утилит на основе шаблона

Подключил репозиторий к своей системе

Использовал chezmoi на нескольких машинах

Настроил новую машину с помощью одной команды

Включил функцию автоматической фиксации и отправлении изменений в репозиторий

5 Выводы

vi/vim — один из самых мощных текстовых редакторов для работы в командной строке. Его изучение важно для системных администраторов, разработчиков и всех, кто работает с Unix-системами. Освоив базовые команды, можно эффективно редактировать файлы без графического интерфейса.
Список литературы{.unnumbered}