Отчет по лабораторной работе №5

Операционные системы

Дмитрий Павлович Стрижов

Содержание

# 1 Цель работы

Настройка рабочей среды.

# 2 Задание

1. Менеджер паролей pass
2. Управление файлами конфигурации

# 3 Теоретическое введение

Менеджер паролей pass

Менеджер паролей pass — программа, сделанная в рамках идеологии Unix.  
Также носит название стандартного менеджера паролей для Unix (The standard Unix password manager).

Основные свойства

Данные хранятся в файловой системе в виде каталогов и файлов.  
Файлы шифруются с помощью GPG-ключа.

Структура базы паролей

Структура базы может быть произвольной, если Вы собираетесь использовать её напрямую, без промежуточного программного обеспечения. Тогда семантику структуры базы данных Вы держите в своей голове.  
Если же необходимо использовать дополнительное программное обеспечение, необходимо семантику заложить в структуру базы паролей.

Семантическая структура базы паролей

Рассмотрим пользователя user в домене example.com, порт 22.  
  
Отсутствие имени пользователя или порта в имени файла означает, что любое имя пользователя и порт будут совпадать:  
  
example.com.pgp  
  
Соответствующее имя пользователя может быть именем файла внутри каталога, имя которого совпадает с хостом. Это полезно, если в базе есть пароли для нескольких пользователей на одном хосте:  
  
example.com/user.pgp  
  
Имя пользователя также может быть записано в виде префикса, отделенного от хоста знаком @:  
  
user@example.com.pgp  
  
Соответствующий порт может быть указан после хоста, отделённый двоеточием (:):  
  
example.com:22.pgp  
example.com:22/user.pgp  
user@example.com:22.pgp  
  
Эти все записи могут быть расположены в произвольных каталогах, задающих Вашу собственную иерархию.

Реализации

Утилиты командной строки

На данный момент существует 2 основных реализации:  
 pass — классическая реализация в виде shell-скриптов (https://www.passwordstore.org/);  
 gopass — реализация на go с дополнительными интегрированными функциями (https://www.gopass.pw/).  
Дальше в тексте будет использоваться программа pass, но всё то же самое можно сделать с помощью программы gopass.

Графические интерфейсы

qtpass  
 qtpass — может работать как графический интерфейс к pass, так и как самостоятельная программа. В настройках можно переключаться между использованием pass и gnupg.  
  
gopass-ui  
 gopass-ui — интерфейс к gopass.  
  
webpass  
 Репозиторий: https://github.com/emersion/webpass  
 Веб-интерфейс к pass.  
 Написано на golang.

Приложения для Android

Password Store  
 URL: https://play.google.com/store/apps/details?id=dev.msfjarvis.aps  
 Репозиторий с кодом: https://github.com/android-password-store/Android-Password-Store  
 Документация: https://android-password-store.github.io/docs/  
 Для синхронизации с git необходимо импортировать ssh-ключи.  
 Поддерживает разблокировку по биометрическим данным.  
 Для работы требует наличия OpenKeychain: Easy PGP.  
  
OpenKeychain: Easy PGP  
 URL: https://play.google.com/store/apps/details?id=org.sufficientlysecure.keychain  
 Операции с ключами pgp.  
 Необходимо будет импортировать pgp-ключи.  
 Не поддерживает разблокировку по биометрическим данным. Необходимо набирать пароль ключа.

Пакеты для Emacs

pass  
 Основной режим для управления хранилищем и редактирования записей.  
 Emacs. Пакет pass  
 Репозиторий: https://github.com/NicolasPetton/pass  
 Позволяет редактировать базу данных паролей.  
  
 Запуск:  
  
 M-x pass  
  
helm-pass  
 Интерфейс helm для pass.  
 Репозиторий: https://github.com/emacs-helm/helm-pass  
  
 Запуск:  
  
 M-x helm-pass  
  
 Выдаёт в минибуфере список записей из базы паролей. При нажатии Enter копирует пароль в буфер.  
  
ivy-pass  
 Интерфейс ivy для pass.  
 Репозиторий: https://github.com/ecraven/ivy-pass

Управление файлами конфигурации

Использование chezmoi для управления файлами конфигурации домашнего каталога пользователя.

Общая информация

Сайт: https://www.chezmoi.io/  
Репозиторий: https://github.com/twpayne/chezmoi

Конфигурация chezmoi

Рабочие файлы

Состояние файлов конфигурации сохраняется в каталоге  
  
~/.local/share/chezmoi  
  
Он является клоном вашего репозитория dotfiles.  
Файл конфигурации ~/.config/chezmoi/chezmoi.toml (можно использовать также JSON или YAML) специфичен для локальной машины.  
Файлы, содержимое которых одинаково на всех ваших машинах, дословно копируются из исходного каталога.  
Файлы, которые варьируются от машины к машине, выполняются как шаблоны, обычно с использованием данных из файла конфигурации локальной машины для настройки конечного содержимого, специфичного для локальной машины.  
  
При запуске  
  
chezmoi apply

вычисляется желаемое содержимое и разрешения для каждого файла, а затем вносит необходимые изменения, чтобы ваши файлы соответствовали этому состоянию.

По умолчанию chezmoi изменяет файлы только в рабочей копии.

Автоматически создавать файл конфигурации на новой машине

При выполнении chezmoi init также может автоматически создать файл конфигурации, если он еще не существует.  
Если ваш репозиторий содержит файл с именем .chezmoi.$FORMAT.tmpl, где $FORMAT есть один из поддерживаемых форматов файла конфигурации (json, toml, или yaml), то chezmoi init выполнит этот шаблон для создания исходного файла конфигурации.  
  
Например, пусть ~/.local/share/chezmoi/.chezmoi.toml.tmpl выглядит так:  
  
{{- $email := promptStringOnce . "email" "Email address" -}}  
  
[data]  
 email = {{ $email | quote }}  
  
 При выполнении chezmoi init будет создан конфигурационный файл ~/.config/chezmoi/chezmoi.toml.  
 promptStringOnce — это специальная функция, которая запрашивает у пользователя значение, если оно еще не установлено в разделе data конфигурационного файла.  
  
Чтобы протестировать этот шаблон, используйте chezmoi execute-template с флагами --init и --promptString, например:  
  
chezmoi execute-template --init --promptString email=me@home.org < ~/.local/share/chezmoi/.chezmoi.toml.tmpl

Пересоздание файл конфигурации

Если вы измените шаблон файла конфигурации, chezmoi предупредит вас, если ваш текущий файл конфигурации не был сгенерирован из этого шаблона.  
  
Вы можете повторно сгенерировать файл конфигурации, запустив:  
  
chezmoi init

Шаблоны

Общая информация

Шаблоны используются для изменения содержимого файла в зависимости от среды.  
Используется синтаксис шаблонов Go.  
Файл интерпретируется как шаблон, если выполняется одно из следующих условий:  
 имя файла имеет суффикс .tmpl;  
 файл находится в каталоге .chezmoitemplates.

Данные шаблона

Полный список переменных шаблона:  
  
chezmoi data  
  
Источники переменных:  
 файлы .chezmoi, например, .chezmoi.os;  
 файлы конфигурации .chezmoidata.$FORMAT. Форматы (json, jsonc, toml, yaml) читаются в алфавитном порядке;  
 раздел data конфигурационного файла.

Способы создания файла шаблона

При первом добавлении файла передайте аргумент --template:  
  
chezmoi add --template ~/.zshrc  
  
Если файл уже контролируется chezmoi, но не является шаблоном, можно сделать его шаблоном:  
  
chezmoi chattr +template ~/.zshrc  
  
Можно создать шаблон вручную в исходном каталоге, присвоив ему расширение .tmpl:  
  
chezmoi cd  
$EDITOR dot\_zshrc.tmpl  
  
Шаблоны в каталоге .chezmoitemplates должны создаваться вручную:  
  
chezmoi cd  
mkdir -p .chezmoitemplates  
cd .chezmoitemplates  
$EDITOR mytemplate

Редактирование файла шаблона

Используйте chezmoi edit:  
  
chezmoi edit ~/.zshrc  
  
Чтобы сделанные вами изменения сразу же применялись после выхода из редактора, используйте опцию --apply:  
  
chezmoi edit --apply ~/.zshrc

Тестирование шаблонов

Тестирование с помощью команды chezmoi execute-template.  
  
Тестирование небольших фрагментов шаблонов:  
  
chezmoi execute-template '{{ .chezmoi.hostname }}'  
  
Тестирование целых файлов:  
  
chezmoi cd  
chezmoi execute-template < dot\_zshrc.tmpl

Синтаксис шаблона

Действия шаблона записываются внутри двойных фигурных скобок, {{ }}.  
Действия могут быть переменными, конвейерами или операторами управления.  
Текст вне действий копируется буквально.  
  
Переменные записываются буквально:  
  
{{ .chezmoi.hostname }}  
  
Условные выражения могут быть записаны с использованием if, else if, else, end:  
  
{{ if eq .chezmoi.os "darwin" }}  
darwin  
  
{{ else if eq .chezmoi.os "linux" }}  
linux  
  
{{ else }}  
other operating system  
  
{{ end }}  
  
Удаление пробелов  
  
 Для удаления проблем в шаблоне разместите знак минус и пробела рядом со скобками:  
  
 HOSTNAME={{- .chezmoi.hostname }}  
  
 В результате получим:  
  
 HOSTNAME=myhostname  
  
Отладка шаблона  
  
 Используется подкоманда execute-template:  
  
 chezmoi execute-template '{{ .chezmoi.os }}/{{ .chezmoi.arch }}'  
  
 Интерпретируются любые данные, поступающие со стандартного ввода или в конце команды.  
  
 Можно передать содержимое файла этой команде:  
  
 cat foo.txt | chezmoi execute-template  
  
Логические операции  
 Возможно выполнение логических операций.  
  
 Если имя хоста машины равно work-laptop, текст между if и end будет включён в результат:  
  
 # common config  
 export EDITOR=vi  
  
 # machine-specific configuration  
 {{- if eq .chezmoi.hostname "work-laptop" }}  
 # this will only be included in ~/.bashrc on work-laptop  
 {{- end }}  
  
 Логические функции  
 eq: возвращает true, если первый аргумент равен любому из остальных аргументов, может принимать несколько аргументов;  
 not: возвращает логическое отрицание своего единственного аргумента;  
 and: возвращает логическое И своих аргументов, может принимать несколько аргументов;  
 or: возвращает логическое ИЛИ своих аргументов, может принимать несколько аргументов.  
  
 Целочисленные функции  
 len: возвращает целочисленную длину своего аргумента;  
 eq: возвращает логическую истину arg1 == arg2;  
 ne: возвращает логическое значение arg1 != arg2;  
 lt: возвращает логическую истину arg1 < arg2;  
 le: возвращает логическую истину arg1 <= arg2;  
 gt: возвращает логическую истину arg1 > arg2;  
 ge: возвращает логическую истину arg1 >= arg2.

Переменные шаблона

Чтобы просмотреть переменные, доступные в вашей системе, выполните:  
  
chezmoi data  
  
Чтобы получить доступ к переменной chezmoi.kernel.osrelease в шаблоне, используйте:  
  
{{ .chezmoi.kernel.osrelease }}

# 4 Выполнение лабораторной работы

Устанавливаем необходимые программы (рис. 1, 2).

Установка менеджера паролей pass

Рис. 1: Установка менеджера паролей pass

Установка менеджера паролей pass

Рис. 2: Установка менеджера паролей pass

Проверяем список ключей (рис. 3).

Список ключей

Рис. 3: Список ключей

Инициализируем хранилище (рис. 4).

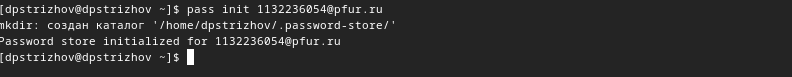


Рис. 4: Инициализация хранилища

Создадим структуру git (рис. 5).

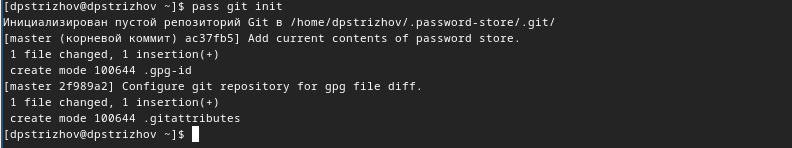


Рис. 5: Создание структуры git

Задаем адрес репозитория на хостинге (рис. 6).

Задание адреса репозитория на хостинге

Рис. 6: Задание адреса репозитория на хостинге

Синхронизируем (рис. 7).

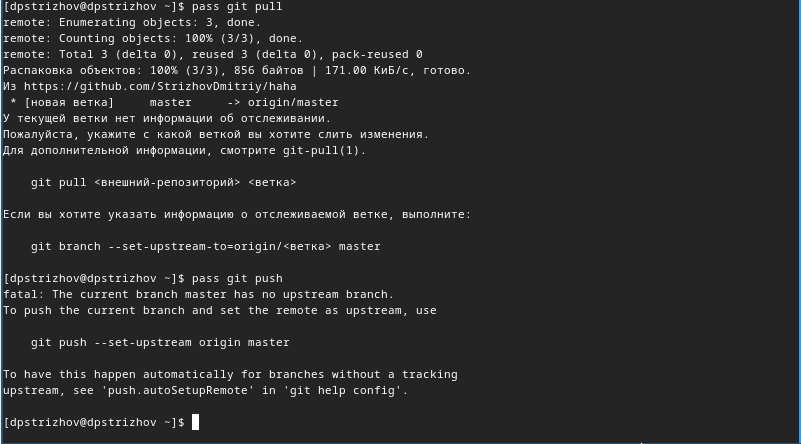


Рис. 7: Синхронизация

Проверяем статус синхронизации (рис. 8).

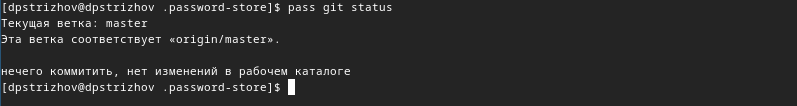


Рис. 8: Проверка статуса синхронизации

Устанавливаем плагин browserpass (рис. 9).

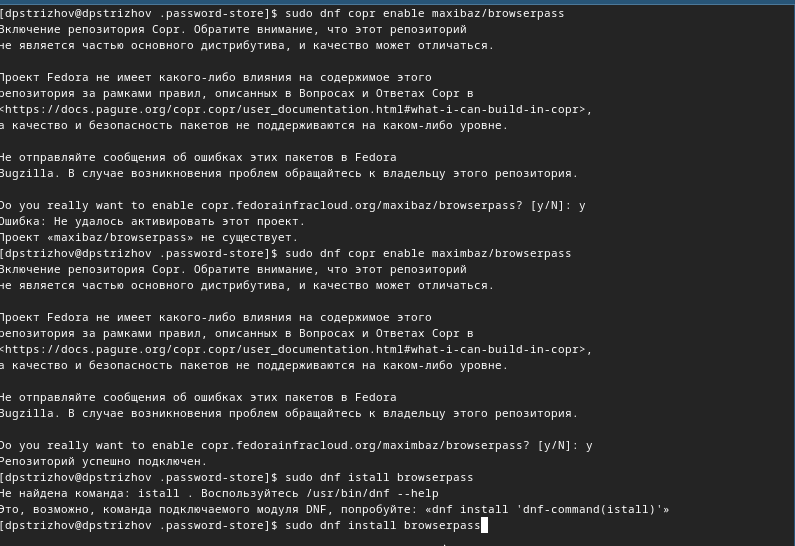


Рис. 9: Установка плагина browserpass

Создаем новый пароль, а потом выводим его, а затем генерируем новый (рис. 10).

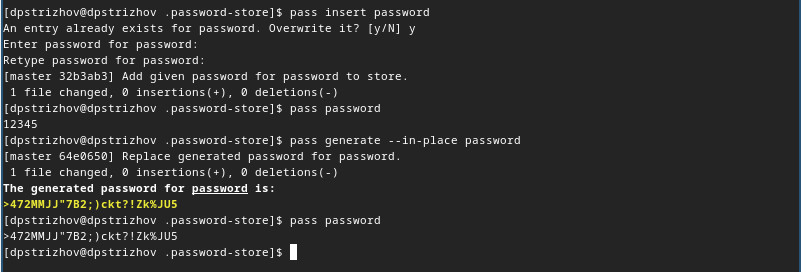


Рис. 10: Новый пароль

Устанавливаем дополнительное програмное обеспечение (рис. 11).

Установка дополнительного програмного обеспечения

Рис. 11: Установка дополнительного програмного обеспечения

Устанавливаем шрифты (рис. 12, 13, 14).

Подключение репозитория

Рис. 12: Подключение репозитория

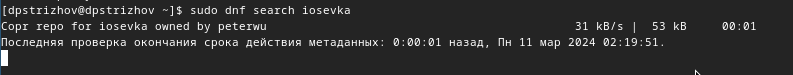


Рис. 13: Подключение репозитория

Установка шрифтов

Рис. 14: Установка шрифтов

Устанавливаем бинарный файл (рис. 15).

Установка бинарного файла

Рис. 15: Установка бинарного файла

Создаем собственный репозиторий с помозью утилит (рис. 16).

Создание собственного репозитория

Рис. 16: Создание собственного репозитория

Проводим инициализацию и проверяем изменения (рис. 17).

Инициализация и проверка последующих изменений

Рис. 17: Инициализация и проверка последующих изменений

Вносим изменения (рис. 18).

Внесение изменений

Рис. 18: Внесение изменений

Извлекаем из репозитория изменений (рис. 19).

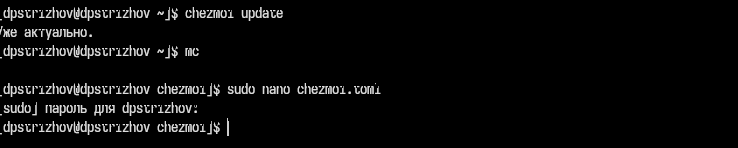


Рис. 19: Извлечение из репозитория изменений

Подключение автоматического вненсения изменений (рис. 20).

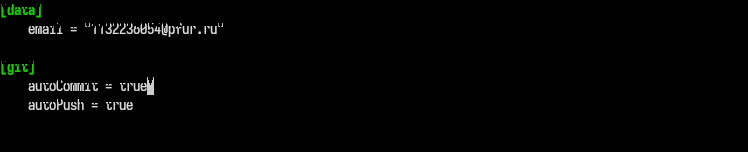


Рис. 20: Подключение автоматического вненсения изменений

# 5 Выводы

За время выполнения лабораторной работы я настроил рабочую систему.

# Список литературы

Лабораторная работа №5: https://esystem.rudn.ru/mod/page/view.php?id=1098796