Создание простого systemd unit

Экспорт в PDF 

Дата создания: 2023/04/22 17:28 (C) mihanik

Источники

- https://newadmin.ru/sozdanie-prostogo-systemd-unit/?ysclid=lgs8xf61vl418127116

Что такое systemd?

systemd — это набор базовых строительных блоков для системы Linux. Он предоставляет диспетчер систем и служб, который работает как PID 1 и запускает остальную часть системы. systemd предоставляет агрессивные возможности распараллеливания, использует активацию сокетов и D-Bus для запуска служб. Предлагает запуск демонов по требованию, отслеживает процессы, использующие группы управления Linux. Поддерживает точки монтирования и автоматического монтирования. Реализует сложную логику управления службами на основе транзакционных зависимостей. systemd поддерживает сценарии SysV и LSB init и работает как замена для sysvinit. Другие части включают демон ведения журнала, утилиты для управления базовой конфигурацией системы. В него входит: имя хоста, дата, локаль, ведение списка зарегистрированных пользователей и запущенных контейнеров. А также виртуальных машин, системные учетные записи, каталоги времени выполнения и параметры. Не забудем о демонах управления простой конфигурацией сети, синхронизацией сетевого времени, пересылкой журналов и разрешением имен.

unitd работает с сервисами описанными в своей конфигурации. Основой для создания сервиса служит юнит (unit) — это текстовый файл с описанием на подобии ini файла в системе windows. Конфигурационный файл состоит из секций. Внутри каждой секции указываются необходимые параметры. Обязательными во всех юнитах являются две секции, остальные используются в зависимости от типа юнита.

Каталоги хранения юнитов

- */usr/lib/systemd/system* — юниты поставляемые вместе с системой и устанавливаемыми приложениями
- */run/systemd/system* — юниты созданные динамически (в рантайме)
- */etc/systemd/system* — юниты системного администратора (тут и будем хранить наши)

Юниты systemd

- **target** — группирует модули
- **service** — отвечает за запуск сервисов (служб) и поддерживает вызов интерпретаторов для исполнения пользовательских скриптов
- **mount** — занимается монтированием файловых систем
- **automount** — автомонтирование файловых систем, используется при обращении к точке монтирования;
- **swap** — отвечает за подключение файла подкачки
- **timer** — запускает модули по расписанию, аналог cron
- **socket** — запуск модуля при подключению к сокету
- **slice** — группировка других модулей в контейнер (дерево) cgroups
- **device** — использует реакцию на подключение какого-либо устройства
- **path** — запуск модуля по событию доступа по конкретному пути в файловой системе

Пример юнита sshd

```
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.target
Wants=sshd-keygen.target

[Service]
Type=notify
EnvironmentFile=-/etc/crypto-policies/back-ends/opensshserver.config
EnvironmentFile=-/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

Давайте рассмотрим данные секции подробнее. В секции [Unit] перечислена основная информация о юните и его зависимостях. В секции [Service] указано какими командами нужно запускать, останавливать, перезапускать сервис. От какого пользователя он должен быть запущен и т.п. В секции [Install] указываем на каком уровне должен стартовать сервис (по аналогии с runlevel).

[Unit]

- **Description** — описание юнита для большего понимания
- **Documentation** — документация по процессу sshd
- **After** — зависимость, т.е. в данном случае запускать юнит только после запуска network.target и sshd-keygen.target

- **Wants** — еще одна зависимость, означает желательно. В примере Wants=sshd-keygen.target, т.е. желательно чтобы было запущено sshd-keygen.target . Желательно но не обязательно.

[Service]

Type — типы запуска служб. Могут быть

- **simple** (по умолчанию) — происходит незамедлительный запуск этой службы, с учетом того что процесс не разветвляется (fork). Не используйте simple если пользуетесь очередностью запуска. Одно исключение это активация сокета.
- **forking** — служба считается запущенной после того, после разветвления процесса с завершением родительского процесса. Используется для запуска классических демонов исключая случаи, когда в таком поведении процесса нет необходимости. Также желательно указать PIDFile=, чтобы systemd мог отслеживать основной процесс.
- **oneshot** — удобен для скриптов, которые выполняют одно задание и завершаются. При необходимости можно задать параметр RemainAfterExit=yes, чтобы systemd считал процесс активным даже после его завершения.
- **notify** — идентичен параметру simple, но с оговоркой, что демон пошлет systemd сигнал о своей готовности. Эталонная реализация данного уведомления представлена в libsystemd-daemon.so.
- **dbus** — служба считается находящейся в состоянии готовности, когда указанный параметр BusName появляется в системной шине DBus.
- **idle** — откладывается выполнение двоичного файла службы до момента выполнения всех остальных задач. В остальном поведение аналогично simple.

Далее в разделе **Service**

- **EnvironmentFile** — файлы переменного окружения
- **ExecStart** — полный путь к исполняемому файлу программы с параметрами запуска
- **ExecReload** — полный пусть к исполняемому файлу программы с параметрами перезапуска программы
- **KillMode** — указывается как будет завершен процесс. В данному случае параметр process говорит о том что будет закрыт только главный процесс
- **Restart** — перезагрузка процесса, параметр on-failure указывает на автоматическую перезагрузку в случает отказа процесса
- **RestartSec** — время ожидания через которое процесс должен перезагрузиться

[Install]

- **WantedBy** — указывает на каком урове запуска стартует сервис, параметр multi-user.target указывает на запуск в многопользовательском режиме без графики

Файлы с юнитами кладем в каталог */etc/systemd/system/*

Итак начнем создание простого systemd unit, подопытным будет утилита мониторинга системы nmon. Откроем на редактирование файл юнита

```
mcedit /etc/systemd/system/nmon.service
```

Добавим туда следующий код

```
[Unit]
Description=NMON monitoring
After=network.target

[Service]
Type=notify
ExecStart=/usr/bin/nmon -V
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Добавить юнит в автозагрузку можно командой

```
systemctl enable nmon
```

Команда запуска юнита

```
systemctl start nmon
```

Статус можно посмотреть командой

```
systemctl status nmon
```

В данном случае процесс после запуска закрылся, потом как при старте был указан ключ -V. Он запустился вывел версию nmon и закрылся.

Важно помнить, если в системе работаем не под root, то перед каждой командой добавляем sudo. В данном случае я взял программу nmon просто для примера. Указанный ключ при запуске показывает номер версии nmon. SystemD имеет огромное количество настроек, в данной статье мы рассмотрели необходимый минимум для создания автозагрузки программы либо скрипта. Возможно в следующих статьях мы капнем systemd поглубже для изучения всех его интересных особенностей.

Для получения подробной информации по systemd, можно зайти на сайт разработчика.

⇑ Наверх ⇑



В моей WIKI постоянно ведётся какая-то работа со статьями.
Если у вас возникли вопросы или замечания,
можете их отправлять на почту **support@mihanik.net**

From:
https://wiki.mihanik.net/ - **wiki.mihanik.net**

Permanent link:
https://wiki.mihanik.net/doku.php/%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%88%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%88:linux:systemd:%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B3%D0%BE_systemd_unit

Last update: **2023/04/22 17:41**