Systemd service

В этой статье мы подробнее посмотрим на юниты **SystemD** с типом **Service**. Разберём параметры из юнита **ssh.service** и не только.

Юниты SystemD типа service

В **SystemD** все сервисы которые можно запускать или останавливать описываются в специальных юнитах **service**. Это значит, что в таких юнитах описывается вся информация, которая поможет запустить сервис (службу).

Наверно юниты SystemD следовало изучать после изучения процессов. И скорее всего в будущем я поменяю очерёдность в оглавлении. Но сейчас я постарался описать службы (юниты типа service) с минимальным объяснением процессов.

Как вам известно, юниты — это обычные файлы. В юнитах типа **service** описывается способ запуска исполняемых файлов (бинарных или скриптов).

В **SystemD** есть специальная утилита, которая позволяет управлять службами — **systemctl**. Вот некоторые её подкоманды:

- **systemctl start <unit.service>** — запуск службы. При этом происходит какая-то подготовка и запуск одного или нескольких исполняемых файлов. А когда запускаются исполняемые файлы, то в системе стартуют соответствующие процессы. Из этого следует, что после запуска службы в системе запустится один или несколько процессов, которые будут работать в одной службе.

- **systemctl stop <unit.service>** — остановка службы. Эта команда должна остановить службу. При этом должны остановиться все связанные со службой процессы. Хотя процессы не обязательно должны завершиться, это зависит от того, что написано в юните.
- **systemctl status <unit.service>** — статус службы. С помощью этой команды можно узнать статус службы. А именно запущена ли она или нет, какие в неё работают процессы и какой из них главный процесс. А также можно посмотреть последние логи службы.
- **systemctl reload <unit.service>** — перечитывание конфигурации. Эта команда заставит работающие процессы перечитать свои конфиги. Хотя на самом деле, она выполнит то, что написано в юните.
- **systemctl restart <unit.service>** — перезапуск службы. То есть остановка и последующее включение службы.
- **systemctl enable <unit.service>** — включение автозапуска. Эта команда включает автозапуск для службы, а вот когда будет срабатывать автозапуск, зависит от написанного в юните.
- **systemctl disable <unit.service>** — выключение автозапуска. А эта команда выключает автозапуск у службы.
- **systemctl cat <unit.service>** — вывод содержимого файла юнита. То есть делает тоже самое, что и команда **cat /путь/unit.service**.

Мне кажется лучше всего изучать написание юнитов с помощью просмотра уже подготовленных юнитов. Давайте посмотрим на один такой юнит — **ssh.service**.

Юнит SystemD — ssh.service

Статус службы

Для начала с помощью команды **systemctl status ssh** посмотрим статус этой службы:

```
alex@deb:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
Active: active (running) since Thu 2022-07-21 13:59:59 MSK; 5min ago
Docs: man:sshd(8)
man:sshd_config(5)
Process: 10614 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 10615 (sshd)
Tasks: 1 (limit: 2340)
Memory: 1.1M
CPU: 76ms
CGroup: /system.slice/ssh.service
└─10615 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```

В этом выводе можно увидеть файл юнита — **/lib/systemd/system/ssh.service**. Также видно что включена автозагрузка службы — **enabled**. И фраза **vendor preset: enabled** — означает что служба поставляется со включенной автозагрузкой. Другими словами, при установке пакета ssh, сразу включится автозапуск этой службы.

Содержимое юнита

Давайте теперь посмотрим на содержимое юнита. Это можно сделать с помощью двух разных команд:

```
alex@s-deb:~$ cat /lib/systemd/system/ssh.service
*** Этот вывод я пропущу, так как он аналогичен выводу следующей команды ***


alex@s-deb:~$ systemctl cat ssh.service
# /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
```

```
[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755


[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Здесь мы видим 3 блока:

- **[Unit]** — здесь можно описать юнит, и указать ограничения на его запуск;
- **[Service]** — сюда добавляется информация, которая используется для запуска, работы и остановки службы;
- **[Install]** — в этом блоке описываются дополнительные условия для запуска или автозапуска службы.

Блок [Unit]

Параметры:

- **Description** — описание юнита;
- **Documentation** — источники документации;
- **After** — запускать юнит после запуска перечисленных юнитов, при этом не требуется чтобы эти юниты были запущены, этот параметр влияет только на очерёдность;

- **Requires** — в нашем примере нет этой опции, но про неё следует знать. Эта опция требует чтобы перечисленные юниты работали. Если они не будут работать, то и наш юнит не запустится;
- **ConditionPathExists** — эта опция проверяет наличия указанного файла. Восклицательный знак перед именем файла означает, что юнит запустится только в том случае, если этого файла в системе нет. Получается чтобы запретить запуск службы **ssh** достаточно создать файл **/etc/ssh/sshd_not_to_be_run**.

Блок [Service]

Параметры:

- **EnvironmentFile** — переменные окружения для службы и её процессов будут браться из указанного файла. Минус перед файлом (как в нашем случае **-/etc/default/ssh**) означает что файл может отсутствовать и это не приведёт к ошибке при запуске службы. Это можно использовать как файл с дополнительными настройками.
- **ExecStartPre** — здесь указывается дополнительная команда, которая выполняется перед запуском службы. Это можно использовать, например для подготовки окружения перед запуском.
- **ExecStart** — команда, которая запускает службу. Именно в этом параметре нужно указать главный исполняемый файл (утилиту или скрипт), ради которого мы создаём службу.
- **ExecReload** — здесь нужно указать которая выполнится при выполнении **systemctl reload <unit.service>**. А если утилита не умеет перечитывать свою конфигурацию, без перезагрузки, то можно опустить этот параметр. Параметров **ExecStartPre**, **ExecStart**, **ExecReload** может быть несколько, если нужно выполнить несколько команд.
- **KillMode** — указывает, как процессы службы должны завершаться. Существует несколько вариантов:
  - **control-group** — сигнал остановки будет послан всем процессам службы;
  - **mixed** — первый сигнал остановки будет послан главному процессу, а второй всем остальным процессам;
  - **process** — сигнал остановки будет послан только главному процессу, все остальные процессы (если они есть) останутся

работать;
    - **none** — никому не будет отправлен сигнал остановки, служба будет выключена, но все процессы продолжат работать (настоятельно не рекомендуется);
- **Restart** — в этом параметре можно указать, в каких случаях нужно перезагружать службу:
    - **on-failure** — при ошибке;
    - **on-success** — при успехе (при корректном завершении службы);
    - **no** — никогда (по умолчанию);
    - **always** — всегда, то есть если ваша служба по каким-то причинам останавливается корректно, то этот параметр её перезапустит. Но это не сработает, если вы выполните (**systemctl stop <unit.service>**).
- **RestartPreventExitStatus** — если служба вылетит с указанным кодом завершения (в нашем случае 255), то она больше не перезапускается, даже если установлен **Restart=on-failure**. Про коды завершения я напишу позже.
- **Type** — тип юнита — очень важный параметр. Отвечает за способ запуска службы и её процессов. Бывают следующие типы:
    - **simple** или **exec** — они используются для запуска программы как службы. Этот тип стоит использовать, когда запускаемая программа не умеет сама запускаться в фоновом режиме. То есть запускается на переднем плане. Прервать выполнение такой программы можно с помощью комбинации клавиш **Ctrl+C**. При этом **simple** отрабатывает быстрее, но не следит за запуском исполняемого файла, указанного в **ExecStart**. А **exec** дожидается запуска исполнимого файла, и только после этого служба считается запущенной. Желательно использовать **exec** вместо **simple**, так как в **simple** служба может оказаться успешно запущенной, даже если исполнимый файл не смог запустится;
    - **forking** — этот тип следует использовать если запускаемое приложение умеет само запускаться в фоновом режиме. Можно использовать этот тип, если вы запускаете с помощью скрипта несколько процессов, которые умеют запускаться в фоне. То есть в **ExecStart** указан скрипт, а в скрипте запускаются фоновые процессы. В этом случае сам скрипт выполнится и завершится, а служба будет управлять запущенными фоновыми процессами;

    - **oneshot** — если подразумевается разовый запуск утилиты или скрипта, то подойдет этот тип. Служба такого типа никогда не будет запущенной. Вы запускаете службу, скрипт выполняется и служба останавливается;
    - **dbus** — если служба использует соединение D-Bus, то можно использовать этот тип службы. Когда служба запустится, то получит имя на шине D-Bus. Если это имя освободится, то служба будет считаться неисправной и все процессы службы начнут завершатся. Про D-Bus также будет написана отдельная статья;
    - **notify** — поведение похоже на **exec**, но дополнительно ожидается, что служба отправит сигнал готовности после своего запуска;
    - **idle** — система отложит выполнение двоичного файла службы до окончания запуска остальных (более срочных) задач, максимум на 5 секунд. В остальном поведение аналогично **simple**.
- **RuntimeDirectory** — после запуска службы будет создана указанная директория. В нашем случае — **/run/sshd/**;
- **RuntimeDirectoryMode** — директория будет создана с указанными правами. В нашем случае — **755 (rwx r-x r-x)**. Про систему прав в Linux я уже писал здесь.

Блок [Install]

Параметры:

- **WantedBy** — если мы включим автозагрузку этой службы (с помощью команды **systemctl enable <имя службы>**), то она должна запуститься при загрузке мультипользовательского режима (**multi-user.target**). Про таргеты и загрузочные таргеты будет следующая статья.
- **Alias** — псевдоним службы. Указание псевдонима приведёт к тому, что к службе можно будет обратиться ещё и по имени псевдонима. Например, вместо **ssh.service** можно использовать **sshd.service** (**systemctl status sshd.service**).

Дополнительные опции

В юните **SystemD — ssh.service** использовались далеко не все из возможных опций. Вот некоторые опции, которые можно использовать для создания своих служб (в блоке **[Service]**):

- **User** — с помощью этого параметра можно указать пользователя (username или uid), от чьего имени будут запущены процессы службы. То есть этот пользователь будет запускать исполняемый файл, указанный в **ExecStart**.
- **ExecStop** — команда, которую нужно выполнить при остановке службы (systemctl stop <unit.service>).
- **KillSignal** — здесь можно указать сигнал остановки процессов. А если этот параметр не указать, то будет использован сигнал **SIGTERM**. Про сигналы, которые можно посылать процессам, я напишу позже.
- **LimitNOFILE** — указывается максимальное число открытых файлов, которые смогут открыть процессы службы.
- **LimitCPU** — можем задать максимальное количество процессорного времени в секундах. Когда лимит будет израсходован, служба завершится с ошибкой.
- **LimitRSS** — лимит потребления памяти в байтах. При достижении лимита служба завершится с ошибкой.
- **LimitNPROC** — максимальное число процессов в службе.
- **Nice** — уровень любезности запускаемых процессов службой. Любезность (**Nice**) — это параметр обратный приоритету. Чем выше Nice, тем ниже приоритет у процесса, и тем меньше он нагружает процессор. Может быть от -20 (самый приоритетный) до 19 (наименьший приоритет).

Итог

Это не все опции, на самом деле их очень много. Для тех кто хочет во всём этом разобраться дам несколько ссылок на официальную документацию **SystemD** (на английском):

- Настройка юнитов — описывают сами юниты, их типы, их расположение и приоритет. А также здесь описываются параметры, которые можно использовать в блоке **[Unit]** и **[Install]**.
- Управление выполнением юнитов (не только служб) — описывают параметры, с помощью которых мы можем контролировать запускаемые процессы. Эти параметры принадлежат блоку **[Service]**. Это лимиты, пользователи, параметры безопасности, переменные окружения и тому подобное.

Revision #1
Created 2 May 2024 05:58:34 by Антон Сергеевич Абраменко
Updated 2 May 2024 06:02:22 by Антон Сергеевич Абраменко