УПРАВЛЕНИЕ СЕРВИСАМИ И ЮНИТАМИ SYSTEMD С ПОМОЩЬЮ SYSTEMCTL

Декабрь 18, 2017 12:13 пп 10 014 views | Комментариев нет
Linux | Amber | 0 Comments

Systemd – это система инициализации и системный менеджер, который становится новым стандартом для Linux-машин. Споры о продуктивности systemd по сравнению с традиционными системами инициализации SysV ведутся до сих пор, тем не менее, эту систему планируют внедрить большинство дистрибутивов, а многие уже сделали это.

Изучение инструментов и демонов systemd и умение работать с ними поможет вам лучше оценить мощность, гибкость и другие возможности системы или, по крайней мере, справиться с минимальными трудностями.

Данный мануал научит вас работать с командой systemctl, главным инструментом управления системы инициализации systemd. Вы узнаете, как управлять сервисами, проверять состояние и работать с конфигурационными файлами.

Управление сервисами

Основная цель init-системы – инициализировать компоненты, которые должны запускаться после загрузки ядра Linux (традиционно они называются «пользовательскими» компонентами). Система инициализации также используется для управления сервисами и демонами сервера. Имея это в виду, начнем знакомство с systemd с простых операций управления сервисами.

В systemd целью большинства действий являются юниты – ресурсы, которыми systemd может управлять. Юниты делятся на категории по типам ресурсов, которые они представляют. Юниты определяются в так называемых юнит-файлах. Тип каждого юнита можно определить по суффиксу в конце файла.

Для задач управления сервисами предназначены юнит-файлы с суффиксом .service. Однако в большинстве случаев суффикс .service можно опустить, так как система systemd достаточно умна, чтобы без суффикса определить, что нужно делать при использовании команд управления сервисом.

Запуск и остановка сервиса

Чтобы запустить сервис systemd, используйте команду start. Если вы работаете не в сессии пользователя root, вам нужно использовать sudo, поскольку эта команда повлияет на состояние операционной системы:

```
sudo systemctl start application.service
```

Как уже говорилось ранее, systemd знает, что для команд управления сервисами нужно искать файлы *.service, поэтому эту команду можно ввести так:

```
sudo systemctl start application
```

Вышеуказанный формат можно использовать в повседневной работе, но в мануале для ясности мы будем использовать суффикс .service.

Чтобы остановить сервис, достаточно ввести команду stop:

```
sudo systemctl stop application.service
```

Перезапуск и перезагрузка

Чтобы перезапустить сервис, используйте restart:

```
sudo systemctl restart application.service
```

Если указанное приложение может перезагрузить свои конфигурационные файлы (без перезапуска), можно использовать reload:

```
sudo systemctl reload application.service
```

Если вы не знаете, может ли сервис перезагрузить свои файлы, используйте команду reload-or-restart. Она перезагрузит сервис, а если это невозможно – перезапустит его.

```
sudo systemctl reload-or-restart application.service
```

Включение и отключение сервисов

Приведенные выше команды необходимы при работе с сервисом в текущей сессии. Чтобы добавить сервис в автозагрузку systemd, его нужно включить.

Для этого существует команда enable:

```
sudo systemctl enable application.service
```

Это создаст символическую ссылку на копию файла сервиса (обычно в /lib/systemd/system или /etc/systemd/system) в точке на диске, где systemd ищет файлы для автозапуска (обычно /etc/systemd/system/some_target.target.want, подробнее об этом — дальше в руководстве).

Чтобы убрать сервис из автозагрузки, нужно ввести:

```
sudo systemctl disable application.service
```

Это удалит символическую ссылку, после этого сервис перестанет запускаться автоматически.

Имейте в виду, что включение сервиса не запускает его в текущей сессии. Если вы хотите запустить сервис и включить его в автозагрузку, вам нужно запустить команды start и enable.

Проверка состояния сервиса

Чтобы проверить состояние сервиса, введите:

```
systemctl status application.service
```

Эта команда выведет состояниесервиса, иерархию групп и первые несколько строк лога.

Например, при проверке состояния сервера Nginx вы можете увидеть такой вывод:

```
nginx.service - A high performance web server and a reverse proxy
server
Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor
preset: disabled)
Active: active (running) since Tue 2015-01-27 19:41:23 EST; 22h ago
Main PID: 495 (nginx)
CGroup: /system.slice/nginx.service
├─495 nginx: master process /usr/bin/nginx -g pid /run/nginx.pid;
error_log stderr;
└─496 nginx: worker process
Jan 27 19:41:23 desktop systemd[1]: Starting A high performance web
```

```
server and a reverse proxy server...
Jan 27 19:41:23 desktop systemd[1]: Started A high performance web
server and a reverse proxy server.
```

Это предоставляет обзор текущего состояния приложения, уведомляет вас о любых проблемах и любых действиях, которые могут потребоваться в дальнейшем.

Существуют также методы проверки конкретных состояний. Например, чтобы проверить, активен ли данный юнит (запущен ли он), вы можете использовать команду is-active:

```
systemctl is-active application.service
```

Это отобразит текущее состояние юнита, обычно это active или inactive. Код завершения будет «0», если юнит активен, что упрощает процесс анализа.

Чтобы узнать, включен ли юнит, вы можете использовать команду is-enabled:

```
systemctl is-enabled application.service
```

Эта команда сообщит, включен ли сервис, и снова определит код завершения как «0» или «1» в зависимости от результата.

Третья команда позволяет определить, находится ли юнит в состоянии сбоя. Это указывает на то, что возникла проблема с запуском рассматриваемого юнита:

```
systemctl is-failed application.service
```

Команда вернет active, если юнит работает правильно, и failed, если случилась ошибка. Если юнит был остановлен намеренно, команда может вернуть unknown или inactive. Код завершения «0» означает, что произошел сбой, а «1» указывает на любое другое состояние.

Обзор состояния системы

Ранее мы рассмотрели команды, необходимые для управления отдельными сервисами, но они не очень полезны для изучения текущего состояния системы. Существует несколько команд systemctl, которые предоставляют эту информацию.

Просмотр списка текущих юнитов

Чтобы запросить список текущих юнитов systemd, используйте команду list-units:

```
systemctl list-units
```

Эта команда покажет список всех юнитов, которые в настоящее время существуют в системе systemd. Результат будет выглядеть примерно так:

```
UNIT                                      LOAD   ACTIVE SUB
DESCRIPTION
atd.service                               loaded active running ATD
daemon
avahi-daemon.service                      loaded active running Avahi
mDNS/DNS-SD Stack
dbus.service                              loaded active running D-Bus
System Message Bus
dcron.service                             loaded active running
Periodic Command Scheduler
dkms.service                              loaded active exited  Dynamic
Kernel Modules System
getty@tty1.service                        loaded active running Getty
on tty1
. . .
```

В выводе есть такие столбцы:

- UNIT – название юнита systemd.
- LOAD – сообщает, была ли конфигурация юнита обработана systemd. Конфигурация загруженных юнитов хранится в памяти.
- ACTIVE – сводное состояние юнита. Обычно это позволяет быстро определить, успешно ли запущен текущий юнит.
- SUB: состояние более низкого уровня, которое сообщает подробную информацию об устройстве. Это часто зависит от типа юнита, состояния и фактического метода, в котором запущен юнит.
- DESCRIPTION – краткое описание функций юнита.

Поскольку команда list-units показывает по умолчанию только активные юниты, все вышеперечисленные записи будут показывать loaded в столбце LOAD и active в столбце ACTIVE. Такой формат является поведением systemctl по умолчанию при вызове без дополнительных команд, поэтому вы увидите то же самое, если вы вызываете systemctl без аргументов:

```
systemctl
```

С помощью systemctl можно запрашивать различную информацию путем добавления флагов. Например, чтобы увидеть все юниты, которые загрузила (или попыталась загрузить) система systemd, независимо от того, активны ли они в данный момент, вы можете использовать флаг -all:

```
systemctl list-units --all
```

Эта команда сообщит о юнитах, которые загрузила или попыталась загрузить система systemd, независимо от их текущего состояния. После запуска некоторые юниты становятся неактивными, а юниты, которые пыталась загрузить systemd, не были найдены на диске.

Вы можете использовать другие флаги для фильтрации результатов. Например, флаг —state= можно использовать для определения состояний LOAD, ACTIVE или SUB. Флаг —all нужно оставить, чтобы система отображала неактивные юниты:

```
systemctl list-units --all --state=inactive
```

Еще один популярный фильтр – это —type=. Он позволяет отфильтровать юниты по типу. К примеру, чтобы запросить только активные юниты, можно ввести:

```
systemctl list-units --type=service
```

Список юнит-файлов

Команда list-units отображает только юниты, которые система systemd попыталась обработать и загрузить в память. Поскольку systemd избирательно читает только те юнит-файлы, которые кажутся ей необходимыми, список не будет включать все доступные юнит-файлы. Чтобы просмотреть список всех доступных юнит-файлов (включая те, что systemd не пыталась загрузить), используйте команду list-unit-files.

```
systemctl list-unit-files
```

Юниты являются представлениями ресурсов, о которых знает systemd. Поскольку systemd не обязательно читает все определения юнитов, она представляет только информацию о самих файлах. Вывод состоит из двух столбцов: UNIT FILE и STATE.

```
UNIT FILE                                   STATE
proc-sys-fs-binfmt_misc.automount           static
dev-hugepages.mount                         static
dev-mqueue.mount                            static
```

```
proc-fs-nfsd.mount                          static
proc-sys-fs-binfmt_misc.mount               static
sys-fs-fuse-connections.mount               static
sys-kernel-config.mount                     static
sys-kernel-debug.mount                      static
tmp.mount                                   static
var-lib-nfs-rpc_pipefs.mount                static
org.cups.cupsd.path                         enabled
. . .
```

Обычно столбец STATE содержит значения enabled, disabled, static или masked. В этом контексте static означает, что в юнит-файле нет раздела install, который используется для включения юнита. Таким образом, эти юниты невозможно включить. Обычно это означает, что юнит выполняет одноразовое действие или используется только как зависимость другого юнита и не должен запускаться сам по себе.

Подробнее о значении masked вы узнаете далее.

Управление юнитами

Теперь вы знаете, как работать с сервисами и отображать информацию о юнитах и юнит-файлах, о которых знает systemd. Получить более конкретную информацию о юнитах можно с помощью некоторых дополнительных команд.

Отображение юнит-файла

Чтобы отобразить юнит-файл, который загрузила systemd, вы можете использовать команду cat (была добавлена в версии systemd 209). Например, чтобы увидеть юнит-файл демона планирования atd, можно ввести:

```
systemctl cat atd.service
[Unit]
Description=ATD daemon
[Service]
Type=forking
ExecStart=/usr/bin/atd
[Install]
WantedBy=multi-user.target
```

На экране вы увидите юнит-файл, который известен текущему запущенному процессу systemd. Это может быть важно, если вы недавно изменяли файлы модулей или если вы переопределяете некоторые параметры в фрагменте юнит-файла (об этом поговорим позже).

Отображение зависимостей

Чтобы просмотреть дерево зависимостей юнита, используйте команду:

```
systemctl list-dependencies sshd.service
```

Она отобразит иерархию зависимостей, с которыми системе необходимо иметь дело, чтобы запустить этот юнит. Зависимости в этом контексте – это те юниты, которые требуются для работы других юнитов, которые находятся выше в иерархии.

```
sshd.service
├─system.slice
└─basic.target
├─microcode.service
├─rhel-autorelabel-mark.service
├─rhel-autorelabel.service
├─rhel-configure.service
├─rhel-dmesg.service
├─rhel-loadmodules.service
├─paths.target
├─slices.target
. . .
```

Рекурсивные зависимости отображаются только для юнитов .target, которые указывают состояния системы. Чтобы рекурсивно перечислить все зависимости, добавьте флаг —all.

Чтобы показать обратные зависимости (юниты, зависящие от указанного элемента), вы можете добавить в команду флаг —reverse. Также полезными являются флаги —before и —after, они отображают юниты, которые зависят от указанного юнита и запускаются до или после него.

Проверка свойств юнита

Чтобы увидеть низкоуровневые свойства юнита, вы можете использовать команду show. Это отобразит список свойств указанного юнита в формате ключ=значение.

```
systemctl show sshd.service
Id=sshd.service
Names=sshd.service
Requires=basic.target
Wants=system.slice
WantedBy=multi-user.target
Conflicts=shutdown.target
Before=shutdown.target multi-user.target
After=syslog.target network.target auditd.service systemd-
```

```
journald.socket basic.target system.slice
Description=OpenSSH server daemon
. . .
```

Чтобы отобразить одно из свойтсв, передайте флаг –p и укажите имя свойства. К примеру, чтобы увидеть конфликты юнита sshd.service, нужно ввести:

```
systemctl show sshd.service -p Conflicts
Conflicts=shutdown.target
```

Маскировка юнитов

Systemd может блокировать юнит (автоматически или вручную), создавая симлинк на /dev/null. Это называется маскировкой юнитов и выполняется командой mask:

```
sudo systemctl mask nginx.service
```

Теперь сервис Nginx не будет запускаться автоматически или вручную до тех пор, пока включена маскировка.

Если вы проверите list-unit-files, вы увидите, что сервис отмечен как masked:

```
systemctl list-unit-files
. . .
kmod-static-nodes.service              static
ldconfig.service                       static
mandb.service                          static
messagebus.service                     static
nginx.service                          masked
quotaon.service                        static
rc-local.service                       static
rdisc.service                          disabled
rescue.service                         static
. . .
```

Если вы попробуете запустить сервис, вы получите такое сообщение:

```
sudo systemctl start nginx.service
Failed to start nginx.service: Unit nginx.service is masked.
```

Чтобы раскрыть (разблокировать) юнит и сделать его доступным, используйте unmask:

```
sudo systemctl unmask nginx.service
```

Это вернет сервис в его прежнее состояние.

Редактирование юнит-файлов

Хотя конкретный формат юнит-файлов не рассматривается в данном руководстве, systemctl предоставляет встроенные механизмы для редактирования и изменения юнит-файлов. Эта функциональность была добавлена в systemd версии 218.

Команда edit по умолчанию открывает сниппет юнит-файла:

```
sudo systemctl edit nginx.service
```

Это будет пустой файл, который можно использовать для переопределения или добавления директив в определение юнита. В каталоге /etc/systemd/system будет создан каталог, который содержит имя устройства с суффиксом .d. Например, для nginx.service будет создан каталог nginx.service.d.

Внутри этого каталога будет создан сниппет с именем override.conf. Когда юнит загружается, systemd объединит в памяти сниппет для переопределения с остальным юнит-файлом. Директивы сниппета будут иметь приоритет над теми, что указаны в исходном юнит-файле.

Если вы хотите отредактировать весь юнит-файл вместо создания сниппета, вы можете передать флаг —full:

```
sudo systemctl edit --full nginx.service
```

Это загрузит текущий юнит-файл в редактор, где его можно будет изменить. Когда редактор закроется, измененный файл будет записан в /etc/systemd/system и будет иметь приоритет над определением юнита системы (обычно он находится где-то в /lib/systemd/system).

Чтобы удалить все сделанные вами дополнения, удалите каталог конфигурации .d или измененный файл сервиса из /etc/systemd/system. Например, чтобы удалить сниппет, можно ввести:

```
sudo rm -r /etc/systemd/system/nginx.service.d
```

Чтобы удалить полный отредактированный файл, введите:

```
sudo rm /etc/systemd/system/nginx.service
```

После удаления файла или каталога нужно перезагрузить процесс systemd, чтобы система больше не пыталась ссылаться на эти файлы и вернулась к использованию системных копий. Для этого введите:

```
sudo systemctl daemon-reload
```

Изменение уровней запуска

Цели – это специальные юнит-файлы, которые описывают уровни системы или точки синхронизации. Как и другие юниты, файлы целей можно определить по суффиксу. В данном случае используется суффикс .target. Сами по себе цели ничего не делают, вместо этого они используются для группировки других юнитов.

Цели можно использовать, чтобы привести систему в определенное состояние. Подобным образом другие системы инициализации используют уровни запуска. Они используются как ссылки, когда доступны определенные функции, что позволяет указать желаемое состояние вместо настройки отдельных юнитов, необходимых для создания этого состояния.

Например, есть цель swap.target, которая используется для того, что чтобы сообщить, что swap готов к использованию. Юниты, которые являются частью этого процесса, могут синхронизироваться с этой целью при помощи директив WantedBy= или RequiredBy=. Юниты, которым нужен своп, могут указывать это условие через спецификации Wants=, Requires= или After=.

Проверка и настройка целей по умолчанию

Процесс systemd имеет цель по умолчанию, которую он использует при загрузке системы. Обеспечение ряда зависимостей этой единственной цели приводит систему в желаемое состояние. Чтобы найти цель по умолчанию, введите:

```
systemctl get-default
multi-user.target
```

Если вы хотите установить другую цель по умолчанию, вы можете использовать set-default. Например, если у вас установлен графический рабочий стол и вы хотите,

чтобы система загружала его по умолчанию, вы можете соответствующим образом изменить цель:

```
sudo systemctl set-default graphical.target
```

Список доступных целей

Просмотреть список доступных целей можно с помощью команды:

```
systemctl list-unit-files --type=target
```

В отличие от уровней запуска, одновременно можно включать несколько целей. Активная цель указывает, что systemd будет пытаться запустить все юниты, привязанные к этой цели, и не попытается их отключить. Чтобы увидеть все активные цели, введите:

```
systemctl list-units --type=target
```

Изоляция целей

Можно запустить все юниты, связанные с целью, и остановить все юниты, которые не являются частью дерева зависимостей. Для этого используется команда isolate. Она похожа на изменение уровня запуска в других системах инициализации.

Например, если вы работаете в графической среде, где активна цель graphical.target, вы можете отключить графическую систему и перевести систему в состояние многопользовательской командной строки, изолировав multi-user.target. Поскольку graphical.target зависит от multi-user.target, но не наоборот, все графические юниты будут остановлены.

Вы можете взглянуть на зависимости изолируемой цели, прежде чем выполнять эту процедуру, чтобы убедиться, что вы не останавливаете жизненно важные сервисы:

```
systemctl list-dependencies multi-user.target
```

Если вас все устраивает, можете изолировать цель:

```
sudo systemctl isolate multi-user.target
```

Сокращения

Существуют цели, определенные для важных событий, таких как выключение или перезагрузка. systemctl также предлагает несколько сокращений для быстрого вызова.

К примеру, чтобы перевести систему в режим отладки, можно ввести просто rescue вместо isolate rescue.target:

```
sudo systemctl rescue
```

Это обеспечит дополнительную функциональность – оповестит всех пользователей системы о событии.

Чтобы остановить систему, вы можете использовать команду halt:

```
sudo systemctl halt
```

Чтобы начать полное завершение работы, вы можете использовать команду poweroff:

```
sudo systemctl poweroff
```

Перезапуск можно начать с помощью команды reboot:

```
sudo systemctl reboot
```

Эти команды сообщат пользователям системы о событиях, чего не сделает стандартная команда. Обратите внимание, большинство машин используют более короткие команды для этих операций, чтобы они работали правильно с systemd.

Например, чтобы перезагрузить систему, вы можете ввести просто:

```
sudo reboot
```

Заключение

Теперь вы знакомы с основными механизмами systemctl и знаете, как управлять системой инициализации с помощью этого инструмента.

Хотя systemctl работает в основном с процессом systemd, в системе systemd есть другие компоненты, которые контролируются другими утилитами. К примеру, для управления логированием и пользовательскими сеансами используются отдельные демоны и утилиты (journald/journalctl и logind/loginctl соответственно).