Модуль systemd

Модуль *systemd* — это набор базовых модулей для обеспечения работы системы *Linux*. Он предоставляет диспетчер системы и служб, который запускает остальную часть операционной системы[1].

Основные функции *systemd* в *CentOS* 7:

- работа с сокетами операционной системы (связи типа ``точка-точка" между процессами операционной системы);
- работа с сокетами приложений - *D-Bus*. (связи типа ``точка-точка" между приложениями);
- работа с внешней периферией (устройствами ввода-вывода, девайсами, могут быть запущены, когда определенный тип оборудования подключается или становится доступным);
- контроль путей до основных папок операционной системы;
- снимки системных состояний (фактически снимок юнитов *systemd*) и восстанавливать предыдущее состояние системы и отслеживание ошибок;
- управление точками монтирования и автомонтирования (при разбиение файловой системы);
- параллелизация запуска системных сервисов;
- работа с транзакциями (обмена запросами и ответами) между юнитами. Модуль *systemd* ищет зависимости юнитов и создает временную транзакцию для проверки целостности. Если транзакция прошла с ошибкой, *systemd* автоматически пытается исправить ее и удалить не требующиеся задания из нее до формирования сообщения об ошибке;
- выполнение скриптов *SysV* (предшественника *systemd*).

В основе работы *systemd* лежит концепция модулей (так называемых юнитов), описывающих отдельные его аспекты работы. Вот основные типы модулей *system (даннные типы модулей могут быть получены при помощи запроса systemctl -t help)*:

- *service* — системный сервис
- *target* — группа юнитов systemd
- *automount* — точка автомонтирования файловой системы
- *device* — файл устройства, распознанного ядром
- *mount* — точка монтирования файловой системы
- *path* — файл или директория в файловой системе

[1] Перевод документации по sytemd можно найти тут http://www2.kangran.su/~nnz/pub/s4a/s4a_latest.pdf Оригинал тут https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd

- *scope* — процесс, созданный извне
- *slice* — группа иерархически организованных юнитов, управляющая системными процессами
- *snapshot* — сохраненное состояние менеджера *systemd*
- *socket* — сокет межпроцессного взаимодействия
- *swap* — Свап-устройство или свап-файл (файл подкачки)
- *timer* — таймер *systemd*

Файлы модуйлей находятся в каталогах:

- */usr/lib/systemd/system/* (модули, предоставляемые пакетами при их установке и установленные пакеты *rpm*, имеют наименьший приоритет)
- */run/systemd/system/* (модули, созданные в рантайме, средний приоретет)
- */etc/systemd/system/* (модули, устанавливаемые системным администратором, имеет наибольший приоритет).

Каждый модуль имеет название состоящие из двух частей (само название и суфикс, соответствующий типу модуля). Так например например *sshd.socket* – представляет собой сокет *ssh*.

В системе *centOs 7* команды *systemd* не требуют указания полного имени модуля, достаточно только его названия например *sshd.socket* и *sshd* будут интерпретированы одинаково. Однако в некоторых случаях полное название указывать рекомендуется. Также следует отметить, что при запросах к точкам монтирования */home* – название преобразуется в в юнит *.mount. (/home* равнозначно *home.mount*), а указание */dev/sda2* соответствует юниту *dev-sda2.device*.

Типичная структура модуля *system* выглядит так, как это показано ниже

```
[Unit]
Description=Daemon to detect crashing apps
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrtd
Type=forking

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

В приведенном примере секция *Unit* содержит общую информацию о сервисе. Такая секция есть не только в сервис-юнитах, но и в других юнитах (например, при управлении устройствами, точками монтирования и т.д.). В примере мы даем описание сервиса и указываем на то, что демон должен быть запущен после *syslog*. В секции *Service* непосредственно содержится информация о сервисе. Используемый

параметр ExecStart указывает на исполняемый файл сервиса. В *Type* указывается, как сервис уведомляет *systemd* об окончании запуска. Секция *Install* содержит информацию о группе (так называемой цели), в которой сервис должен стартовать. В данном случае сервис должен быть запущен, когда будет активирована группа *multi–user.target.*

Модули *systemd* могут быть заменены, а также можно создавать и свои модули). Чтобы полностью заменить файл юнита */usr/lib/systemd/system/юнит*, создайте файл с таким же именем */etc/systemd/system/юнит* и перезапустите юнит для обновления символических ссылок: *# systemctl reenable юнит*

Задание 1

1. написать свой файл сервиса systemd. Для тестирования скопировать его в */etc/systemd/system/имя_сервиса.service*. Для того, чтобы systemd узнал о сервисе выполните команду *systemctl daemon-reload*. Получите информацию о созданном вами сервисе.

Набор команд *systemctl*

Главная команда для мониторинга и управления *systemd — systemctl*. Фактически это набор команд, предназначенных для решения таких задач, как управление отдельными сервисами и группами сервисов (целями, включая их взаимосвязи), управление энергопитанием, и предоставление функционала для удаленной работы с машиной.

Так, например запрос текущего состояние системы:
systemctl status,

запрос списка запущенных модулей (юнитов):
systemctl list-units

Также список установленных модулей можно получить при помощи запроса
systemctl list-unit-files

Состояние каждого модуля может быть получено при помощи запроса вида
systemctl status имя модуля
(например, *systemctl status rsyslog* или *systemctl status dev/sda)*

При помощи команды *systemctl* можно запускать, перезапускать и останавливать каждый из модулей (проверка при помощи команды *is-active*):
systemctl start, stop, restart и reload (перезапуск конфигурации)

также можно разрешать и запрещать автозапуск каждого из модулей при помощи команд
systemctl enable, diable, и проверка при помощи команды *is-enable*

запрещать запуск/остановку модулей (максировать) при помощи команд
systemctl mask и unmask

Управление питанием реализуется при помощи одного из следующих запросов:
systemctl halt, reboot, poweroff, suspend, hibernate, hybrid-sleep.

При задействовании таких команд как *enable, disable* и *mask,* чтобы соответственно запустить, остановить или маскировать указанный юнит сразу при выполнении команды, а

не после перезагрузки следует использовать опцию *--now,* например *systemctl enable --now* юнит.

Следует отметить, что после внесения изменений в *systemd*– для того, чтобы они были видны, требуется перезагрузка сервиса при помощи запроса *systemctl daemon-reload*.

Для управления удаленной машиной по SSH используйте команду
systemctl — host user_name@host_name command,

где user_name — имя пользователя, host_name — имя хоста, которым осуществляется удаленное управление, а command — выполняемая команда systemd.

Задание 2:

1. получите информацию о сервисе rsyslog попробуйте его перезагрузить, перезагрузить его конфигурацию, выключить и включить – обратите внимание на изменения в статусе сервиса – сделайте выводы.

Группировка сервисов по целям

Каждый модуль *systemd* поддерживает объединение группы модулей вместе по их взаимным зависимостям и в качестве стандартизированных точек синхронизации. Такое объединение называется целью (*target*). Каждая цель имеет имя и предназначена для выполнения конкретных задач; при этом несколько целей могут быть активны одновременно. Целями могут быть, например *poweroff.target (runlevel0.target)* — завершение работы и отключение системы или *multi–user.target (runlevel2.target, runlevel3.target, runlevel4.target)* — настройка неграфической многопользовательской системы.

Информация о текущих целях может быть получена при помощи запроса
systemctl list-units –type target

для просмотра вообще всех целевых юнитов используется команада
systemctl list-units — type target — all.

Поменять цель можно при помощи запроса
 # systemctl isolate (например *systemctl isolate graphical.target*)

Узнать текущую цель по умолчанию можно так:
systemctl get-default

Для установки новой цели загрузки по умолчанию
 # systemctl set-default multi-user.target

Взаимосвязи внутри каждой цели нужны для обеспечения их последовательного запуска и установления путей транзакций. Например модуль *graphical.target*, использующийся для старта графической сессии, запускает системные сервисы *GNOME Display Manager (gdm.service*) и *Accounts Service (accounts–daemon.service)* и активирует *multi–user.target*. В свою очередь *multi–user.target* запускает другие системные сервисы, такие как *Network*

Manager (NetworkManager.service) или *D-Bus (dbus.service)* и активирует другие целевые юниты *basic.target*. Запрос поиска всех сервисов запускаемых до/после интересующего*:*

systemctl list-dependencies --after auditd.service или before auditd.service

Для запуска различных целей по расписанию используется системный сервис cron. Он позволяет задавать расписание для запуска пользовательских и системных программ. Конфигурационный файл хранится в */var/spool/cron/*. Файл представляет собой набор команд с указанием времени их периодического запуска. Каждая строка имеет вид

*0 5 * * 1 cd /var/tmp/updater;./run_update.sh*

В начале строки указывается действующее расписание для запуска скрипта, во второй части - строка запуска команды. Расписание имеет вид минуты, часы, день, месяц, день недели. Приведенный пример интерпретируется как запуск скрипта обновления в 05:00 каждый понедельник.

Каждой цели соответствует приоритет уровня его запуска. В таблице 1 приведены такие уровни.

Таблица 1 – уровни запуска целей *systemd*

Уровень запуска	**Цель systemd**	**Примечания**
0	runlevel0.target, poweroff.target	Выключение системы
1, s, single	runlevel1.target, rescue.target	Однопользовательский уровень запуска
2, 4	runlevel2.target, runlevel4.target, multi-user.target	Уровни запуска, определенные пользователем/специфичные для узла. По умолчанию соответствует уровню запуска 3
3	runlevel3.target, multi-user.target	Многопользовательский режим без графики. Пользователи, как правило, входят в систему при помощи множества консолей или через сеть
5	runlevel5.target, graphical.target	Многопользовательский режим с графикой. Обычно эквивалентен запуску всех служб на уровне 3 и графического менеджера входа в систему
6	runlevel6.target, reboot.target	Перезагрузка
emergency	emergency.target	Аварийная оболочка

При создании пользовательских целей предполагается, что создается новый модуль-цель с названием */etc/systemd/system/цель*, который берет за основу один из существующих уровней запуска (например, */usr/lib/systemd/system/graphical.target*, создаёте каталог */etc/systemd/system/ graphical.target.wants*, а после этого — символические ссылки на те службы из каталога */usr/lib/systemd/system/*, которые вы хотите включить при загрузке.

Задание 3:

1 Получите значение цели по умолчанию

2 Вывидите ссылку на цель по умолчанию

ls -l /etc/systemd/system/default.target

3 Поменяйте цель по умолчанию *на multi-user.target* (команда *isolate*)

4 Перейдите в режим восстановления (*rescue.target*)

Проверить работу режима можно запросом *systemctl --no-wall rescue*

5 Верните обратно первоначальную цель по умолчанию.

Задание 2:

1. создайте новую цель и добавьте ее в systemd.