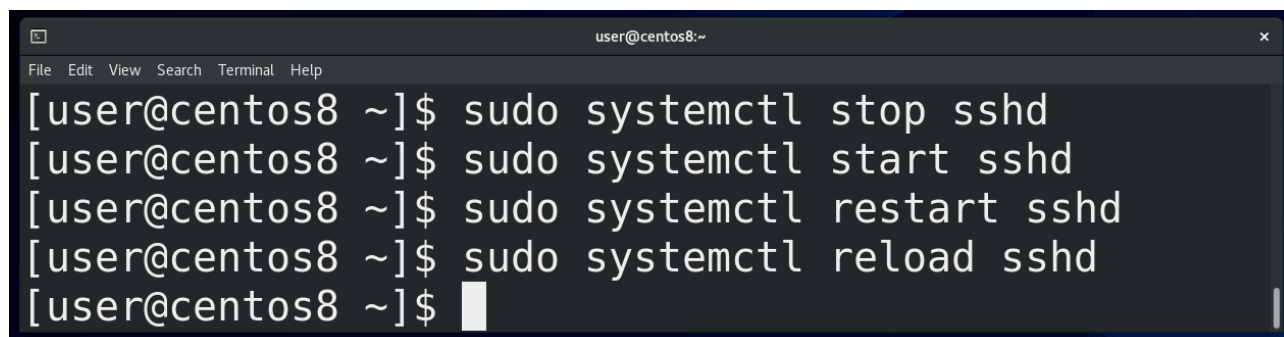
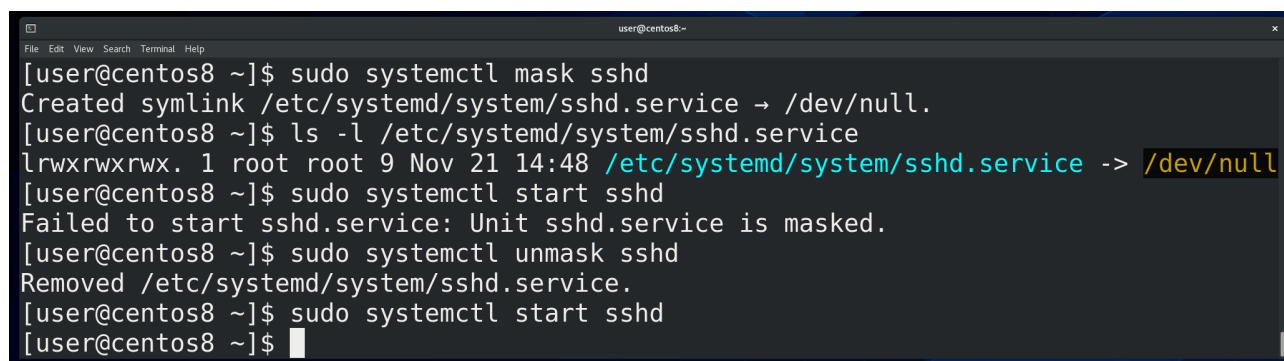


Как мы разобрались в прошлый раз, при запуске компьютера система инициализации запускает всякие сервисы, которые работают в фоне. По большей части работа системного администратора заключается в том, чтобы следить за состоянием операционной системы, устанавливать и настраивать всякие сервисы. На серверах крутятся всякие сервисы, допустим, тот же веб сервер, где крутятся сайты.

A terminal window titled 'user@centos8:~' with a menu bar (File, Edit, View, Search, Terminal, Help). It shows a series of commands to manage the sshd service: stop, start, restart, and reload. The prompt is [user@centos8 ~]\$ for each line.

```
[user@centos8 ~]$ sudo systemctl stop sshd
[user@centos8 ~]$ sudo systemctl start sshd
[user@centos8 ~]$ sudo systemctl restart sshd
[user@centos8 ~]$ sudo systemctl reload sshd
[user@centos8 ~]$
```

systemd, как сервисный менеджер, позволяет управлять этими сервисами. Для примера возьмём сервис под названием sshd. Вы часто будете встречать сервисы с буквой d в конце - она означает daemon, то есть демон программы ssh. Базовые операции - это остановить, запустить и перезапустить сервис - то есть `sudo systemctl stop sshd`, `sudo systemctl start sshd` и `sudo systemctl restart sshd` соответственно. Как понятно из названия, restart останавливает и запускает сервис заново. Обычно перезапускают сервис, когда изменяются какие-то настройки демона. Нужно его заново запустить, чтобы он перечитал свои настройки. Для каких-то сервисов это не проблема, но бывают важные сервисы, в которых даже секундная остановка вызывает проблемы. Для таких сервисов существует возможность перезапустить основной демон, при этом не затронув текущие процессы. Для этого используется опция reload - `sudo systemctl reload sshd`.

A terminal window titled 'user@centos8:~' with a menu bar (File, Edit, View, Search, Terminal, Help). It shows commands to mask and unmask the sshd service. The output shows the creation of a symlink to /dev/null and the failure to start the service when masked. The prompt is [user@centos8 ~]\$ for each line.

```
[user@centos8 ~]$ sudo systemctl mask sshd
Created symlink /etc/systemd/system/ssh.service → /dev/null.
[user@centos8 ~]$ ls -l /etc/systemd/system/ssh.service
lrwxrwxrwx. 1 root root 9 Nov 21 14:48 /etc/systemd/system/ssh.service -> /dev/null
[user@centos8 ~]$ sudo systemctl start sshd
Failed to start sshd.service: Unit sshd.service is masked.
[user@centos8 ~]$ sudo systemctl unmask sshd
Removed /etc/systemd/system/ssh.service.
[user@centos8 ~]$ sudo systemctl start sshd
[user@centos8 ~]$
```

Как мы помним, `systemctl enable sshd` даёт возможность запускать сервис автоматом при включении компьютера. Мы можем сделать `systemctl disable sshd`, если не хотим, чтобы сервис стартовал при включении. И тем не менее, другие программы и пользователи могут запустить этот сервис при необходимости. Если же мы хотим, чтобы этот сервис нельзя было запустить, мы можем его замаскировать, с помощью команды `systemctl mask sshd`. Как видите, создаётся символическая ссылка, ведущая на `/dev/null` - то есть, при попытке запустить сервис ничего не произойдёт. Но это касается только сервиса, если мы вручную запустим программу, она будет работать. Ну и с помощью `unmask` - `sudo systemctl unmask sshd` - мы можем сервис размаскировать.

```
user@centos8 ~$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-11-21 14:59:29 +04; 1s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 10328 (sshd)
    Tasks: 1 (limit: 11474)
   Memory: 1.2M
    CGroup: /system.slice/sshd.service
            └─10328 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@ope

Nov 21 14:59:29 centos8 systemd[1]: Starting OpenSSH server daemon...
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on 0.0.0.0 port 22.
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on :: port 22.
Nov 21 14:59:29 centos8 systemd[1]: Started OpenSSH server daemon.
lines 1-15/15 (END)
```

Также мы можем посмотреть статус этого сервиса с помощью команды `status - systemctl status sshd`. В строчке `Loaded` мы видим путь до основного файла сервиса. Дальше 2 раза видим слово `enabled`. Первый `enabled` говорит о том, что сервис включён, т.е., он будет запускаться при включении компьютера. Перед вторым `enabled` написано `vendor preset`. То есть, имеется в виду, что этот сервис был включён по умолчанию ещё при установке программы или операционной системы.

```
user@centos8 ~$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; ven
   Active: inactive (dead) since Sat 2020-11-21 15:16:37 +04; 8s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 12081 ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
 Main PID: 12081 (code=exited, status=0/SUCCESS)
```

В строчке `Active` мы видим, что сервис работает. Если остановить сервис, здесь будет писаться `inactive`.

```
user@centos8:~  
File Edit View Search Terminal Help  
● vboxadd.service  
  Loaded: loaded (/opt/VBoxGuestAdditions-6.1.4/init/vbox  
  Active: active (exited) since Sat 2020-11-21 14:11:47 +  
Main PID: 1009 (code=exited, status=0/SUCCESS)  
  Tasks: 2 (limit: 11474)  
Memory: 15.9M  
CGroup: /system.slice/vboxadd.service  
└─2168 VBoxClient --vmsvga  
└─2173 VBoxClient --vmsvga
```

Иногда, вместо active (running), можно увидеть active (exited) - `systemctl status vboxadd`. Собственно, ничего страшного в этом нет. Если это какой-то демон, который постоянно работает на фоне, `systemd` может отслеживать состояние процесса и говорить, что он работает, то есть running. Но иногда вместо полноценного демона за сервисом стоит какой-то скрипт, который запускается, выполняет свою работу, запускает какие-то программы и завершается. И вроде скрипт больше не работает, но свою работу он сделал. Поэтому так и выходит - вроде это и не демон, отслеживать нечего, но при этом то что нужно работает.

```
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ sudo systemctl status sshd  
● sshd.service - OpenSSH server daemon  
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)  
  Active: active (running) since Sat 2020-11-21 14:59:29 +04; 1s ago  
    Docs: man:sshd(8)  
          man:sshd_config(5)  
Main PID: 10328 (sshd)  
  Tasks: 1 (limit: 11474)  
Memory: 1.2M  
CGroup: /system.slice/sshd.service  
└─10328 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@ope  
Nov 21 14:59:29 centos8 systemd[1]: Starting OpenSSH server daemon...  
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on 0.0.0.0 port 22.  
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on :: port 22.  
Nov 21 14:59:29 centos8 systemd[1]: Started OpenSSH server daemon.  
lines 1-15/15 (END)
```

Вернёмся к нашему sshd. Дальше у нас ссылки на документацию и PID основного процесса, который запустился при старте сервиса. Дальше Tasks - общее количество процессов - включая основной и его дочерние. Memory - сколько оперативки использует этот сервис. Дальше CGroup - контрольная группа, в которую входят процессы этого сервиса. С помощью контрольных групп на уровне ядра можно изолировать группу процессов и выделять им ограниченные ресурсы - сколько-то оперативки, сколько-то процессора и т.п. Эдакая виртуализация на уровне самой операционной системы. Чуть ниже в статусе мы видим логи. К ним мы ещё вернёмся.

Я не хочу ударяться в объяснение каждой из команд `systemctl`, типа `systemctl cat sshd`, `show`, `edit` и всё такое. Не все команды используются часто, многие специфичны. На что-то мы наткнёмся и разберём, что-то вы по работе разберёте, а с чем-то и в вовсе не столкнётесь.

```
run-user-42.mount          loaded active mounted /run/user/42
sys-fs-fuse-connections.mount loaded active mounted FUSE Control File System
sys-kernel-config.mount    loaded active mounted Kernel Configuration File
sys-kernel-debug.mount     loaded active mounted Kernel Debug File System
var-lib-nfs-rpc_pipefs.mount loaded active mounted RPC Pipe File System
cups.path                  loaded active running CUPS Scheduler
systemd-ask-password-plymouth.path loaded active waiting Forward Password Requests
systemd-ask-password-wall.path loaded active waiting Forward Password Requests
init.scope                 loaded active running System and Service Manager
session-2.scope            loaded active running Session 2 of user user
lines 38-47
```

Как мы упоминали в прошлый раз, systemd отвечает не только за сервисы. Если запустить команду `systemctl --all`, мы увидим информацию о всех unit-ax - а тут и сервисы, и таргеты, и устройства, и mount-ы, и всякое другое. С сервисами и таргетами мы разобрались, немного поговорим про другие юниты.

```
SysFSPath=/sys/devices/pci0000:00/0000:00:01.1/ata2/host1/target1:0:0/1:0:0:0
Id=dev-cdrom.device
Names=dev-cdrom.device
Following=sys-devices-pci0000:00-0000:00:01.1-ata2-host1-target1:0:0-1:0:0:0
Description=VBOX_CD-ROM VBox_GAs_6.1.4
LoadState=loaded
ActiveState=active
SubState=plugged
StateChangeTimestamp=Sat 2020-11-21 14:11:35 +04
StateChangeTimestampMonotonic=5712119
InactiveExitTimestamp=Sat 2020-11-21 14:11:35 +04
lines 1-11
```

Помните, когда мы говорили про ядро, мы упоминали udev? Программа, которая делает какие-то действия при виде устройств - даёт названия устройствам, создаёт ссылки в директории `/dev`, может передать ядру какие-то параметры для устройства, запустить какие-то программы и т.п. Так вот, udev - тоже демон, но, к тому же, он является частью systemd, генерирует юнит файлы для устройств, называемые device unit-ами. Это позволяет сделать связь между устройствами и другими сервисами.

```
Where=/mydata
What=/dev/mapper/myraidvg-myraidlv
Options=rw,relatime,seclabel
Type=ext4
TimeoutUSec=1min 30s
ControlPID=0
DirectoryMode=0755
SloppyOptions=no
```

Или, допустим, mount unit-ы. Для примера возьмём `mydata.mount` - `systemctl show mydata.unit`. При включении компьютера кто-то же должен примонтировать все файловые системы, указанные в `fstab`? Так вот, systemd генерирует специальные unit-ы на основе

записей `fstab` и монтирует их. При этом он смотрит зависимости, скажем, отличает локальные файловые системы от сетевых и на основе этого формирует зависимости при создании юнитов. Есть ещё `swapon` юнит-ы - примерно тоже самое, но для `swapon` разделов.

Про остальные юнит-ы мы поговорим, когда будем разбирать темы, связанные с ними. А сегодня мы разобрали как работать с сервисами, как смотреть информацию о них, и в целом стали лучше понимать, чем же занимается `systemd`.