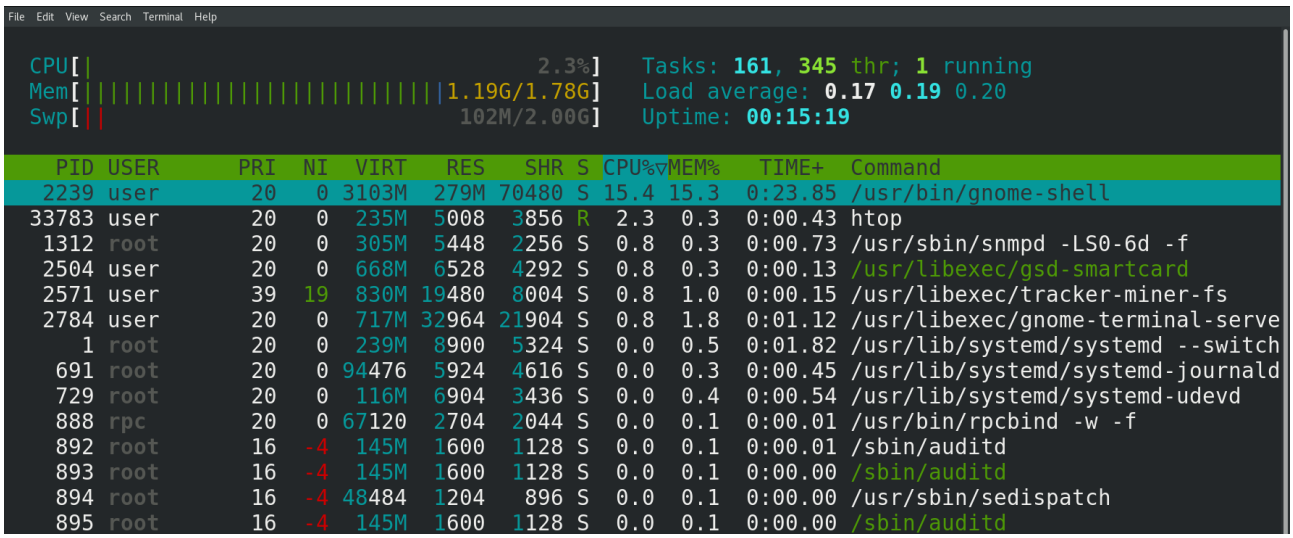


Мы с вами помним: когда запускается программа - появляется соответствующий процесс. Но если посмотреть список процессов на свежезапущенной операционной системе, мы увидим более сотни процессов, хотя я всего лишь запустил эмулятор терминала, bash и htop. Значит, всё остальное было запущено другими программами.

То окно, в котором я залогинился - это ведь тоже какая-то программа, благодаря которой я и могу войти в систему. И вот этот графический интерфейс - тоже какая-то программа. Если я открою браузер и зайду на какой-то сайт - я может и не думаю об этом, но чтобы я смог зайти на этот сайт, сработала ещё одна программа, которая отвечает за работу с сетью. Т.е. чтобы я мог выполнить какие-то простые действия, должны работать какие-то программы.

Де́мон (ст.-слав. де́монъ[3] от др.-греч. δαίμων [даймон] «дух», «божество»[4][5]) — собирательное название сверхъестественных существ или духов, занимающих низшее по сравнению с богами положение[5], которые могут играть как положительную, так и отрицательную роль[5].

У древних греков существовало философское понятие Даймоний. Сократ и его последователи — Платон, стоики и другие, отождествляли с даймонием «внутренний голос» человека, совесть. В римской мифологии им соответствует гений[5], в христианстве — ангел-хранитель[6][7][8].

Они всегда работают на фоне и помогают пользователям удобно пользоваться операционной системой. Все эти программы, которые работают на фоне, называются демонами, или даймонами - и тут отсылка не к библейским демонам, а к древнегреческой мифологии. Что-то типа ангелов хранителей.

Т.е., чтобы я мог нормально работать, должны работать демоны. Но чтобы они работали, их должен кто-то запустить. Этим занимается специальная программа - система инициализации. Есть различные реализации этой программы, мы с вами рассмотрим systemd, которая используется во многих популярных дистрибутивах, в том числе Centos и RHEL.

```
[user@centos8 ~]$ ls -l /sbin/init
lrwxrwxrwx. 1 root root 22 Jul 21  2020 /sbin/init -> ../lib/systemd/systemd
[user@centos8 ~]$
```

В прошлый раз мы остановились на том, что ядро вместе с initramfs запускает программу /sbin/init - ls -l /sbin/init. /sbin/init олицетворяет систему инициализации, и, как видите, сейчас она показывает, что системой инициализации является systemd.

```
SYSTEMD(1)                                systemd                                SYSTEMD(1)

NAME
       systemd, init - systemd system and service manager

SYNOPSIS
       /usr/lib/systemd/systemd [OPTIONS...]

       init [OPTIONS...] {COMMAND}

DESCRIPTION
       systemd is a system and service manager for Linux operating systems. When run
       as first process on boot (as PID 1), it acts as init system that brings up and
       maintains userspace services.

       For compatibility with SysV, if systemd is called as init and a PID that is
       not 1, it will execute telinit and pass all command line arguments unmodified.
       That means init and telinit are mostly equivalent when invoked from normal
       login sessions. See telinit(8) for more information.
```

Система инициализации - man systemd - это первый процесс, запускаемый в пользовательском пространстве. У него PID - 1. Мы упоминали про процесс с pid 1, когда говорили о процессах - если убить родительский процесс, у дочернего процесса родителем станет процесс с номером 1. Так вот, после запуска система инициализации должна запустить демоны. Но это не просто список программ, которые нужно запустить - какие-то программы нужно запускать раньше, чем другие, какие-то программы могут конфликтовать и всё такое. А что делать, если какая-то из программ не запустилась? Нужно же ещё дать возможность пользователям при желании решать, какие программы запускать при включении, какие не запускать. Все эти задачи и стоят перед системой инициализации.

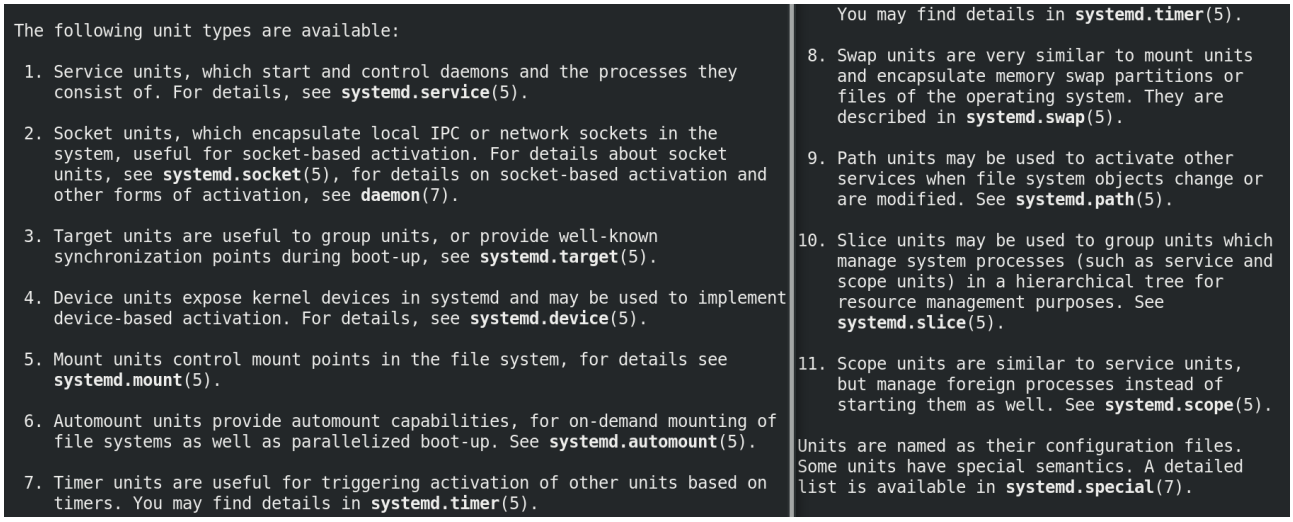
systemd, кроме того, что является системой инициализации, также отвечает за многое другое, например, за управление сервисами, логами, сетью и т.п. Нужно понимать, что роль системы инициализации, как и говорит название - запуск операционной системы, чтобы всё начало работать как надо. На этом полномочия системы инициализации заканчиваются. Но systemd позиционирует себя как системный и сервисный менеджер, он связывает и заменяет многие компоненты операционной системы. Из-за чего, с одной стороны, многое друг с другом интегрировано, легче разрабатывать и администрировать, а с другой - сложнее заменить один компонент на другой и появляется зависимость дистрибутивов от systemd.

> Дуг Макилрой, изобретатель каналов Unix и один из основателей традиции Unix, обобщил философию следующим образом:
>
> > «Философия Unix гласит:
> >
> > Пишите программы, которые делают что-то одно и делают это хорошо.
> >
> > Пишите программы, которые бы работали вместе.
> >
> > Пишите программы, которые бы поддерживали текстовые потоки, поскольку это универсальный интерфейс».

Очень сложно заменить systemd на что-то другое - придётся многое переделывать. Это противоречит философии Unix-а - пишите программы, которые делают одну вещь и делают её хорошо. В итоге сообщество пользователей разделилось на сторонников и противников использования systemd. Я всё это к тому, что пусть вас не пугает большой функционал systemd, он нацелен на большее количество задач, чем просто инициализировать систему. Поэтому что-то мы рассмотрим сегодня, а что-то останется на потом.

Демонов много, они по разному запускаются, по разному останавливаются, что-то нужно перезапускать в случае ошибки, что-то не нужно, что-то нужно раньше запустить, что-

то позже. Т.е. к ним нужен как массовый подход, чтобы удобно было ими управлять, так и индивидуальный. Поэтому демонов оборачивают в так называемые сервисы. Сервис - что-то типа инструкции по эксплуатации демона. В сервисе указано, как запускать демон, как его останавливать, с какими другими демонами он связан и т.п. Какие-то сервисы приходят вместе с операционной системой, другие сервисы появляются при установке программ, а что-то вы можете и сами написать. Например, если вы работаете в компании, где программисты написали какую-то программу, а вам нужно обеспечить работу этой программы на серверах, то это, скорее всего, ваша задача - написать сервис, чтобы программа нормально стартовала при запуске операционной системы, нормально завершалась и всё такое.



The following unit types are available:

1. Service units, which start and control daemons and the processes they consist of. For details, see **systemd.service**(5).

2. Socket units, which encapsulate local IPC or network sockets in the system, useful for socket-based activation. For details about socket units, see **systemd.socket**(5), for details on socket-based activation and other forms of activation, see **daemon**(7).

3. Target units are useful to group units, or provide well-known synchronization points during boot-up, see **systemd.target**(5).

4. Device units expose kernel devices in systemd and may be used to implement device-based activation. For details, see **systemd.device**(5).

5. Mount units control mount points in the file system, for details see **systemd.mount**(5).

6. Automount units provide automount capabilities, for on-demand mounting of file systems as well as parallelized boot-up. See **systemd.automount**(5).

7. Timer units are useful for triggering activation of other units based on timers. You may find details in **systemd.timer**(5).

8. Swap units are very similar to mount units and encapsulate memory swap partitions or files of the operating system. They are described in **systemd.swap**(5).

9. Path units may be used to activate other services when file system objects change or are modified. See **systemd.path**(5).

10. Slice units may be used to group units which manage system processes (such as service and scope units) in a hierarchical tree for resource management purposes. See **systemd.slice**(5).

11. Scope units are similar to service units, but manage foreign processes instead of starting them as well. See **systemd.scope**(5).

Units are named as their configuration files. Some units have special semantics. A detailed list is available in **systemd.special**(7).

И так, мы разобрали, что такое сервис. systemd работает с unit-ами, а сервисы - один из типов таких unit-ов. Есть ещё другие типы юнитов - например, юниты устройств, юниты монтирования и т.д. Но нас сейчас интересуют сервисные юниты и таргеты. Таргеты - это тоже юниты, представляющие из себя группу юнитов. target - от слова цель - говорит о конечном результате, достигаемом с помощью группы юнитов. Допустим, чтобы у нас был графический интерфейс, чтобы я мог открыть браузер и зайти на сайт, послушать музыку и т.п. - одного сервиса недостаточно. А вот если я возьму группу сервисов, отвечающих за графический интерфейс, сеть, звук и т.п. - то это будет готовый результат, который я хочу - т.е. это графический target.

```
[user@centos8 ~]$ systemctl get-default
graphical.target
[user@centos8 ~]$ systemctl list-dependencies graphical.target
graphical.target
● ├─accounts-daemon.service
● ├─gdm.service
● ├─rtkit-daemon.service
● ├─systemd-update-utmp-runlevel.service
● ├─udisks2.service
● └─multi-user.target
●   ├─atd.service
●   ├─auditd.service
●   ├─avahi-daemon.service
●   ├─chronyd.service
●   ├─crond.service
```

И если мы посмотрим вывод команды systemctl get-default, то мы как раз увидим, что systemd по умолчанию грузит таргет с названием graphical.target. А с помощью команды systemctl list-dependencies graphical.target мы как раз увидим, какие юниты нужны для этого таргета. Как видите, графический таргет предполагает использование multi-user таргета, а внутри него огромное количество других юнитов. На серверах, по умолчанию, вместо графического таргета используется multi-user - примерно тоже самое, но нет графического интерфейса, что позволяет сэкономить ресурсы.

```
[user@centos8 ~]$ sudo systemctl set-default multi-user.target
[sudo] password for user:
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target.
[user@centos8 ~]$ systemctl get-default
multi-user.target
[user@centos8 ~]$ sudo systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
[user@centos8 ~]$
```

И если мы захотим, чтобы у нас тоже было как на серверах, чтобы всё работало, но без графического интерфейса, мы можем поменять таргет по умолчанию, с помощью команды sudo systemctl set-default multi-user.target. Ну и заметим мы это только при запуске. А пока вернём как было - sudo systemctl set-default graphical.target.

```
[user@centos8 ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@centos8 ~]$
```

Как видите, при выполнении этих команд у нас создаются и удаляются символические ссылки. И тут участвуют 2 директории - /etc/systemd и /usr/lib/systemd. Когда вы устанавливаете какую-то программу или в целом операционную систему, то все файлы юнитов попадают в директорию /usr/lib/systemd. Допустим, посмотрим файл graphical таргета - systemctl cat graphical.target. Тут у нас описание юнита, путь к документации. Ну и как видите, тут написано Requires=multi-user.target, т.е. для загрузки графического таргета требуется загрузка multi-user таргета. Также тут написано Wants=display-manager.service. Display manager - это та программа, которая у нас спрашивает логин при входе в систему, ну и она же грузит рабочее окружение. Wants означает, что если этот сервис есть, то нужно его загрузить, а вот если он не грузится, или нету такого сервиса, то ничего страшного, таргет всё равно прогрузится. Если проблема с Requires то таргет перестанет грузиться. Также у нас тут Conflicts - то есть этот таргет не может одновременно работать с этими сервисами и таргетами, и что запуск этого таргета остановит работу указанных здесь сервиса и таргета. After означает порядок, после каких сервисов и таргетов грузится этот юнит.

Ну и AllowIsolate означает, можно ли использовать этот таргет как состояние, к которому можно перейти с помощью команды systemctl isolate. Т.е. таргеты - это группа юнитов, и, в случае с graphical.target или multi-user.target, их можно использовать как состояние загрузки операционной системы - т.е. с графическим интерфейсом или без, но всё же есть все нужное для работы. Но есть, например, таргет сети - network target - опять же, группа юнитов, нужная для работы сети, но на него переключаться бессмысленно, так как это не готовое состояние операционной системы, в котором можно работать, а просто группа юнитов. Для примера, есть rescue.target - это состояние системы, при котором большинство демонов не работает, и это нужно для решения каких-то проблем, допустим, когда система не грузится.

```
File  Edit  View  Search  Terminal  Help
[user@centos8 ~]$ sudo systemctl isolate rescue.target
```

```
[  OK  ] Started Session c1 of user gdm.
[  OK  ] Started User Manager for UID 42.
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):
```

```
CPU[|                                  1.3%]   Tasks: 11, 3 thr; 1 running
Mem[||||||||||||                     124M/1.78G]   Load average: 0.29 0.71 0.35
Swp[|                               8.75M/2.00G]   Uptime: 00:03:52

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1 root       20   0  511M 11820  8712 S  0.0  0.6  0:01.81 /usr/lib/systemd/systemd --switched
  692 root       20   0 94352 10208  8584 S  0.0  0.5  0:00.57  ├─ /usr/lib/systemd/systemd-journal
  727 root       20   0  116M 11212  7556 S  0.0  0.6  0:00.46  ├─ /usr/lib/systemd/systemd-udevd
  953 rngd       20   0  156M  6536  5828 S  0.0  0.3  0:05.28  ├─ /sbin/rngd -f --fill-watermark=0
  979 rngd       20   0  156M  6536  5828 S  0.0  0.3  0:03.91  │  └─ /sbin/rngd -f --fill-watermar
 1520 dnsmasq    20   0 71888  2360  1928 S  0.0  0.1  0:00.00  ├─ /usr/sbin/dnsmasq --conf-file=/
 1521 root       20   0 71860   428     0 S  0.0  0.0  0:00.00  │  └─ /usr/sbin/dnsmasq --conf-file
 2648 user       20   0  247M   340     0 S  0.0  0.0  0:00.00  ├─ /usr/bin/VBoxClient --vmsvga
 3022 user       20   0  583M 10096  8572 S  0.0  0.5  0:00.02  ├─ ibus-daemon --xim --panel disabl
 3075 user       20   0  583M 10096  8572 S  0.0  0.5  0:00.00  │  ├─ ibus-daemon --xim --panel dis
 3123 user       20   0  583M 10096  8572 S  0.0  0.5  0:00.00  │  └─ ibus-daemon --xim --panel dis
 3265 root       20   0 83436  4496  3696 S  0.0  0.2  0:00.00  └─ /usr/lib/systemd/systemd-sulogin
 3266 root       20   0  230M  5416  3504 S  0.0  0.3  0:00.10     └─ bash
 3297 root       20   0  234M  4328  3528 R  0.7  0.2  0:00.10        └─ htop
```

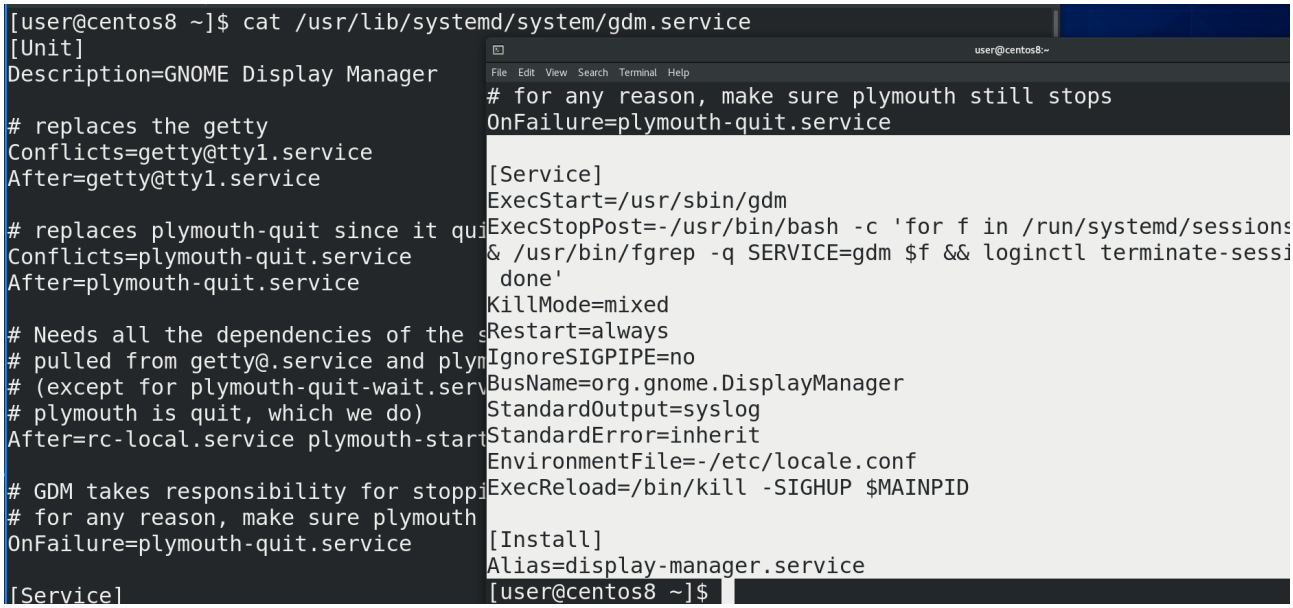
Сделаем sudo systemctl isolate rescue.target и, как видите, теперь у меня система просит root пароль для перехода в режим восстановления. Вводим пароль и смотрим htop - как видите, процессов очень мало.

```
root@centos8 ~]# systemctl isolate graphical.target _
```

Ну и чтобы вернуться обратно, я делаю systemctl isolate graphical.target.

```
[user@centos8 ~]$ ls /etc/rc
rc0.d/    rc1.d/    rc2.d/    rc3.d/    rc4.d/    rc5.d/    rc6.d/    rc.d/     rc.local
[user@centos8 ~]$ ls /etc/rc
```

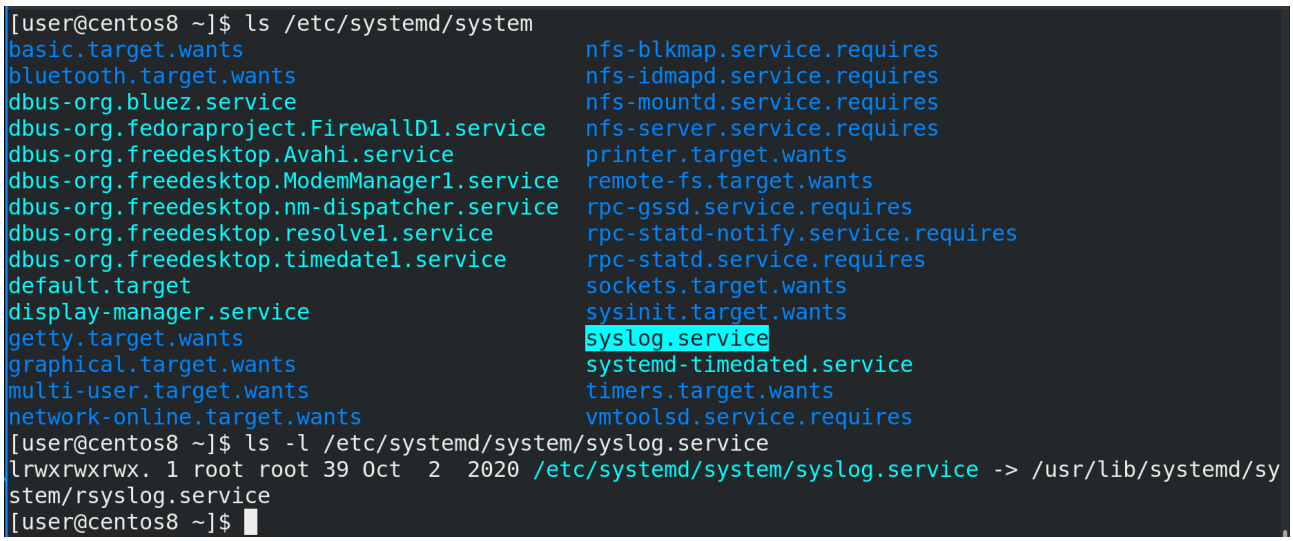
В старых системах инициализации эти состояния, к которым можно было переходить, назывались runlevel-ами. Они были пронумерованы - от 0 до 6. Среди которых 0 это shutdown, 6 это restart, 1 это аналог rescue таргета, 3 это аналог multi-user таргета, 5 - аналог графического таргета, ну и оставшееся это промежуточные состояния. Были директории с соответствующими номерами, в которых лежали скрипты, которые выполнялись при переходе на определённый runlevel. Собственно, это и было основой системы инициализации. Это просто полезно знать, потому что некоторые люди до сих пор говорят о ранлвлах, могут на собеседовании спросить, ну и вы вполне можете наткнуться на другую систему инициализации. Я не буду сравнивать системы инициализации, но, если вам интересно, у Семаева есть ролики про различные системы инициализации, плюс можете почитать по ссылке.



Мы поговорили о таргетах - группах юнитов. Теперь же давайте посмотрим на какой-нибудь определённый сервисный unit - допустим, gdm - тот самый дисплейный менеджер — cat /usr/lib/systemd/system/gdm.service. Тут у нас кроме секции Unit появились ещё две секции - Service и Install. В секции [Service] у нас есть информация о том, как сервис запускает демон - ExecStart, т.е. просто запускает команду /usr/sbin/gdm. А вот ExecStopPost запускает команду после остановки сервиса. Как видите, это какой-то скрипт, и вы можете в нём разобраться. Все опции я разбирать не буду, по большей части у разных сервисов могут быть свои опции, знать всё наизусть не надо, всегда можно обратиться к документации. Что может быть интересно так это секция [Install]. Как видите, тут написан Alias=display-manager.service.

```
[user@centos8 ~]$ sudo systemctl enable gdm
[user@centos8 ~]$ sudo systemctl disable gdm
Removed /etc/systemd/system/display-manager.service.
[user@centos8 ~]$ sudo systemctl enable gdm
Created symlink /etc/systemd/system/display-manager.service → /usr/lib/systemd/system/gdm.service
.
[user@centos8 ~]$ sudo systemctl is-enabled gdm
enabled
[user@centos8 ~]$
```

У нас есть две команды - systemctl enable и systemctl disable. Если мы хотим, чтобы какой-то сервис запускался при включении операционной системы, мы запускаем команду systemctl enable и имя сервиса - sudo systemctl enable gdm. Как видите, никакого вывода не было, потому что этот сервис уже был включён. Если мы хотим убрать из автозапуска этот сервис, делаем - sudo systemctl disable gdm. В выводе написано, что удалён файл /etc/systemd/system/display-manager.service. Если опять сделаем enable - sudo systemctl enable gdm - то увидим, что создалась символическая ссылка display-manager.service, которая ведёт на файл gdm.service. Т.е. такое вот название символической ссылки, на основе того, что было написано в Alias в секции Install. Ну и можно посмотреть, включён ли сервис, с помощью команды systemctl is-enabled gdm.



Что ещё интересно - символическая ссылка создаётся в директории /etc/systemd/system. Если в /usr/lib/systemd у нас файлы этих сервисов, то в /etc/systemd преимущественно символические ссылки, означающие, что данный сервис включён. Ну и если у нас какие-то свои сервисы, написанные нами вручную, то правильнее всего считается класть их именно в /etc/systemd.

Подводя итоги. Для нормальной работы операционной системы нужны программы, работающие в фоне - демоны. Для запуска демонов при включении компьютера нужна система инициализации, одной из которых является systemd. systemd много чего умеет, помимо запуска демонов. Для правильной работы с демонами используются сервисы. systemd для этого использует service unit-ы и target unit-ы - т.е. группы юнитов. Чтобы сервисы запускались при запуске операционной системы, они должны быть enabled, что мы делали с помощью команды systemctl enable, ну или наоборот — чтобы убрать из автозапупска — systemctl disable. Таким образом операционная система запускает все нужные программы при включении.

Теперь, объединяя эту и предыдущую тему, вы имеете представление, что именно происходит при запуске компьютера и операционной системы.