

Глава 3. Процесс загрузки и уровни выполнения.

3.1. Последовательность процесса загрузки.



Последовательность процесса загрузки

- Процесс загрузки компьютера состоит из нескольких этапов
 - BIOS (UEFI)
 - Включение питания
 - POST
 - Bootloader
 - Загрузчик
 - Первая стадия (MBR)
 - Вторая стадия (основной загрузчик)
 - Поиск ядра и образа начальной загрузки
 - Запуск ядра
 - Запуск системных процессов
 - Инициализация системы демоном systemd (initd)

При включении питания компьютера автоматически запускаются программы, находящиеся в ПЗУ.

Исторически, начиная с первых IBM совместимых компьютерах, использовался BIOS (Basic Input/Output Services – Базовые службы ввода-вывода).

Микросхема ПЗУ в таких компьютерах выполняется по технологии CMOS (Complementary Metal-Oxide Semiconductor – Комплементарный метало-оксидный полупроводник).

Примечание: Особенностью этой технологии является то, что программа, сохраняемая в таком ПЗУ может быть настроена без необходимости полного перепрограммирования ПЗУ.

Программа, которая позволяет настраивать поведение BIOS называется CMOS Setup.

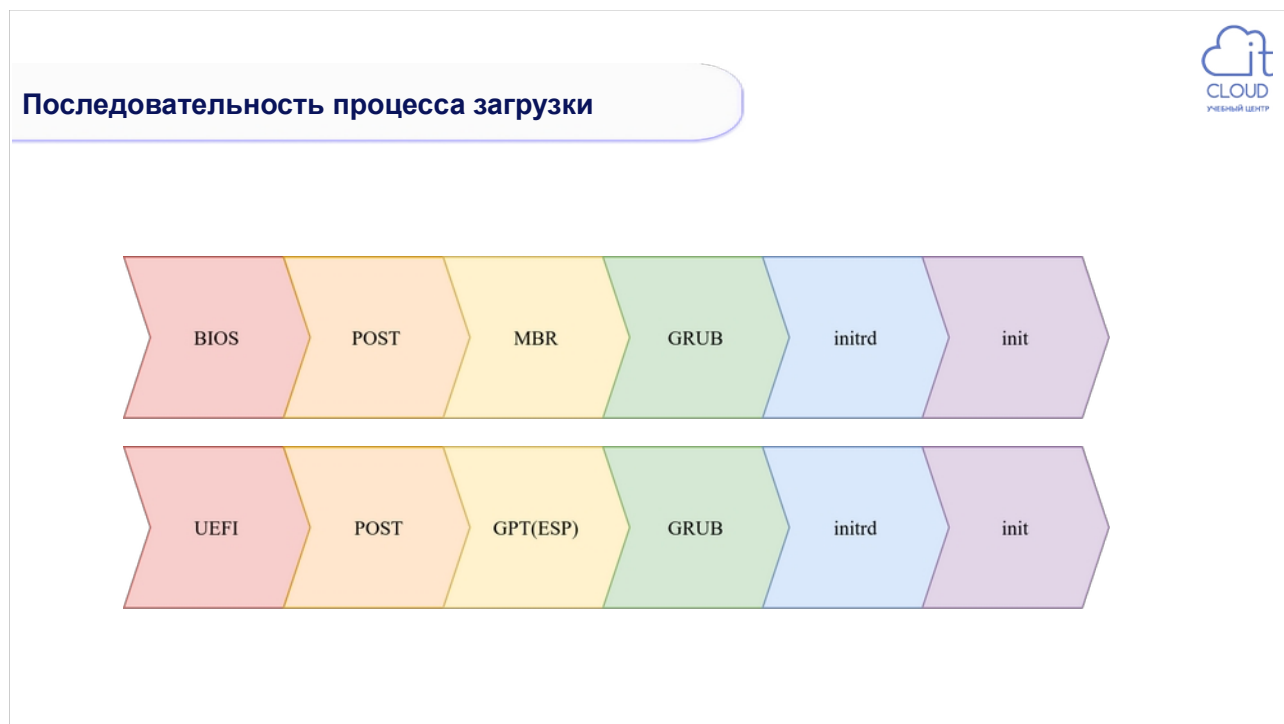
В 1990-х Intel разработал новый стандарт для компьютеров с архитектурой Itanium. Изначальное название — Intel Boot Initiative (Загрузочная инициатива Intel), позже было переименовано в EFI.

Основной мотивацией разработки EFI было преодоление ограничений BIOS: 16-битный исполняемый код, адресуемая память 1 Мбайт, проблемы с одновременной инициализацией нескольких устройств, ограничения на размер дисков.

В 2005 году Intel внесла эту спецификацию в UEFI Forum, который теперь ответственен за развитие и продвижение EFI. EFI был переименован в Unified EFI (UEFI).

Одна из программ, входящих в BIOS/UEFI называется POST (Power On Self Test – Проверка самого компьютера при включении питания). Эта программа выполняет начальную проверку конфигурации компьютера и при наличии проблем выдает специальные

сигналы (обычно звуковые).



После удачного завершения программы POST запускается другая программа в BIOS, которая называется Bootstrap Loader (аппаратный загрузчик). Она предназначена для обращения к загрузочному устройству, указанному с помощью CMOS Setup в BIOS, поиска на нем загрузочного сектора и, при удачном поиске, загрузке его.

Если загрузка осуществляется с магнитного диска, то программа загрузки находится в первом по порядку секторе. Для жестких магнитных дисков загрузочный раздел еще называют MBR (Master Boot Record – Главная загрузочная запись).

Аппаратный загрузчик опознает программу загрузки в загрузочном разделе по сигнатуре ее последних двух байт (510 и 511 байты начиная с 0 байта – в шестнадцатеричном виде, соответственно, байты 0x1FE и 0x1FF). Сигнатура 16 бит, составленная этими байтами, является числом 0xAA55. Именно по этой сигнатуре опознается загрузочный сектор, который и загружается аппаратным загрузчиком в ОЗУ.

Если программный загрузчик в MBR отсутствует, то его поиск осуществляется в первом секторе раздела, помеченного в таблице разделов жесткого диска, как активный.

Размер секторов на магнитном диске – 512 байт, поэтому программа, находящаяся в загрузочном секторе, проста и не может выполнять сложных действий.

Примечание: В некоторых загрузчиках (не в GNU/Linux!) программный загрузчик в MBR выдает на экран список разделов, с которых возможно продолжение загрузки.

Для преодоления этого ограничения загрузчики операционных систем разбивают на несколько частей.

Загрузчики, чаще всего применяемые в GNU/Linux – LILO и GRUB, состоят из двух частей.

1. Одна из них, называемая начальным загрузчиком, помещается либо в MBR, либо в первый сектор активного раздела. Задачей начального загрузчика является поиск и загрузка второй части загрузчика.

Глава 3. Процесс загрузки и уровни выполнения.

2. Вторая часть загрузчика представляют собой более сложную программу, которая может загрузить ядро Linux (или другое ядро) и выполнить необходимые предварительные действия для загрузки операционной системы (например, создание электронного диска с примитивным образом корневой файловой системы).

Загрузка с UEFI реализована несколько иначе.

1. С GPT-раздела с идентификатором EF00 (ESP, EFI system partition) и файловой системой FAT32 по умолчанию загружается и запускается файл `\efi\boot\boot[название архитектуры].efi`, например: `\efi\boot\bootx64.efi`.

2. Следующий этап либо запуск загрузчика, либо прямая загрузка ядра Linux.

Также в большинстве реализаций UEFI возможна загрузка в режиме совместимости с диска с разметкой MBR.

При использовании UEFI возможна также *Secure Boot*, смысл которой заключается в проверке цифровых подписей ядра и модулей ядра.

Secure Boot одна из причин почему может не загрузиться Linux, например когда вы скомпилируете новые модули ядра самостоятельно, но не подпишите их.

Загрузчики, используемые в GNU/Linux, способны передавать ядру Linux различные конфигурационные параметры и дополнительные команды.

Примечание: В частности, ядру Linux может быть передана команда считать из файловой системы на электронном диске специальный командный файл, который чаще всего называется `linuxrc`. В этом командном файле можно указать список дополнительных действий, которые необходимо выполнить при инициализации системы GNU/Linux.

После загрузки ядра и его инициализации, если в параметрах, переданных ядру, не указано иное, производится запуск программы `/sbin/init` (или в новых ОС GNU/Linux `systemd`).

Ее конфигурационный файл `/etc/inittab`.

Процесс, соответствующий этой программе всегда имеет PID равный 1.

Эта программа осуществляет дальнейшую инициализацию операционной системы GNU/Linux, зависящую от выбранного режима работы (например, однопользовательский или многопользовательский режим).

В Linux и UNIX подобных системах такие режимы работы называются уровнями исполнения (`runlevels`).

В зависимости от выбранного уровня исполнения команда `/sbin/init` исполняет тот или иной набор специальных командных файлов, называемых инициализационными скриптами.

Именно инициализационные скрипты определяют какие программы будут работать в фоновом режиме на том или ином уровне исполнения, то есть какова будет функциональность системы.

Пример: однопользовательский режим применяется для обслуживания системы и при работе в нем нет необходимости в запуске сетевых приложений. Подробнее вопросы инициализации системы на различных уровнях исполнения будут рассмотрены позже в этой главе.

Одна из программ, которые запускает процесс `/sbin/init` – это программа `getty` или ее разновидность.

Глава 3. Процесс загрузки и уровни выполнения.

Эта программа выводит приглашение войти в сеанс (`login:`). Если пользователь вводит что-либо в ответ на это приглашение, то эта строка интерпретируется как имя пользователя, которое передается программе `login` как аргумент.

Далее выводится приглашение ввести пароль (`password:`), который проверяется в базе данных паролей (например, в `/etc/shadow` – в зависимости от настроек системы).

Если аутентификация производится успешно, то запускается оболочка, указанная для пользователя в файле `/etc/passwd`, осуществляя, таким образом, вход в сеанс пользователя.

Перед выводом приглашения `login:` на экран печатается содержимое файла `/etc/issue` (или `/etc/issue.net`, если подключение осуществляется через сеть). А после входа в сеанс на экран выводится содержимое файла `/etc/motd` (Message Of The Day – Сообщение дня).

3.2. Загрузчик GRUB2.

Загрузчик GRUB



- Grand Unified Bootloader — универсальный загрузчик Unix подобных систем (Linux, FreeBSD, Solaris ...)
- Поддерживает цепную загрузку (chainload)
- Имеется встроенная оболочка
- Первая версия почти не используется

Загрузчик GRUB (GRand Unified Bootloader) является современной альтернативой LILO.

В отличие от загрузчика LILO, который использует данные о точном расположении нужного образа ядра на магнитном носителе, загрузчик GRUB обращается к файловой системе на этом носителе для поиска файла ядра.

Примечание: Загрузчик LILO обращается к заданному номеру сектора диска, считывая определенное количество байт, а загрузчик GRUB обращается к файлу в файловой системе. Загрузчик GRUB распознает различные типы файловых систем, например, ext2, ext3, ReiserFS, XFS, JFS. GRUB способен работать с большими дисками, имеющими более 1024 цилиндров. При этом обеспечивается загрузка ядер ОС с разделов, расположенных в любом месте диска.

GRUB способен загружать ядра разнообразных операционных систем, например, GNU/Linux, GNU/HURD, FreeBSD и других.

Для операционных систем, загрузка ядер которых не поддерживается, загрузчик GRUB обеспечивает цепную загрузку (chainload), то есть передачу управления загрузчику другой операционной системы.

GRUB поддерживает автоматическую декомпрессию загружаемых файлов, сжатых в формате zip.

GRUB обладает встроенной командной оболочкой, позволяющей гибко управлять процессом загрузки, а также автоматизировать его, задавая в конфигурационном файле загрузчика последовательности команд, которые должны быть выполнены в процессе загрузки.

Загрузчик GRUB поддерживает загрузку через локальную сеть посредством протокола TFTP (Trivial File Transfer Protocol) и поддерживает управление загрузкой с удаленного последовательного терминала.

Загрузчик GRUB2

- Основное новшество — возможность описать загрузку ОС без привязки к физическому расположению устройств
- Конфигурационный файл — `/boot/grub2/grub.cfg`
- В GRUB2 конфигурационный файл не редактируется вручную
- Вместо этого имеется специальная команда `grub2-mkconfig`
- Для смены параметров загрузки ядер Linux необходимо вносить изменения в файл `/etc/default/grub`

В современных GNU/Linux как правило применяется загрузчик GRUB2.

GRUB2 — дальнейшее развитие проекта GRUB.

Основное новшество — возможность описать загрузку ОС без привязки к физическому расположению устройств.

В GRUB2 изменился формат файла конфигурации.

Конфигурационный файл — `/boot/grub2/grub.cfg`

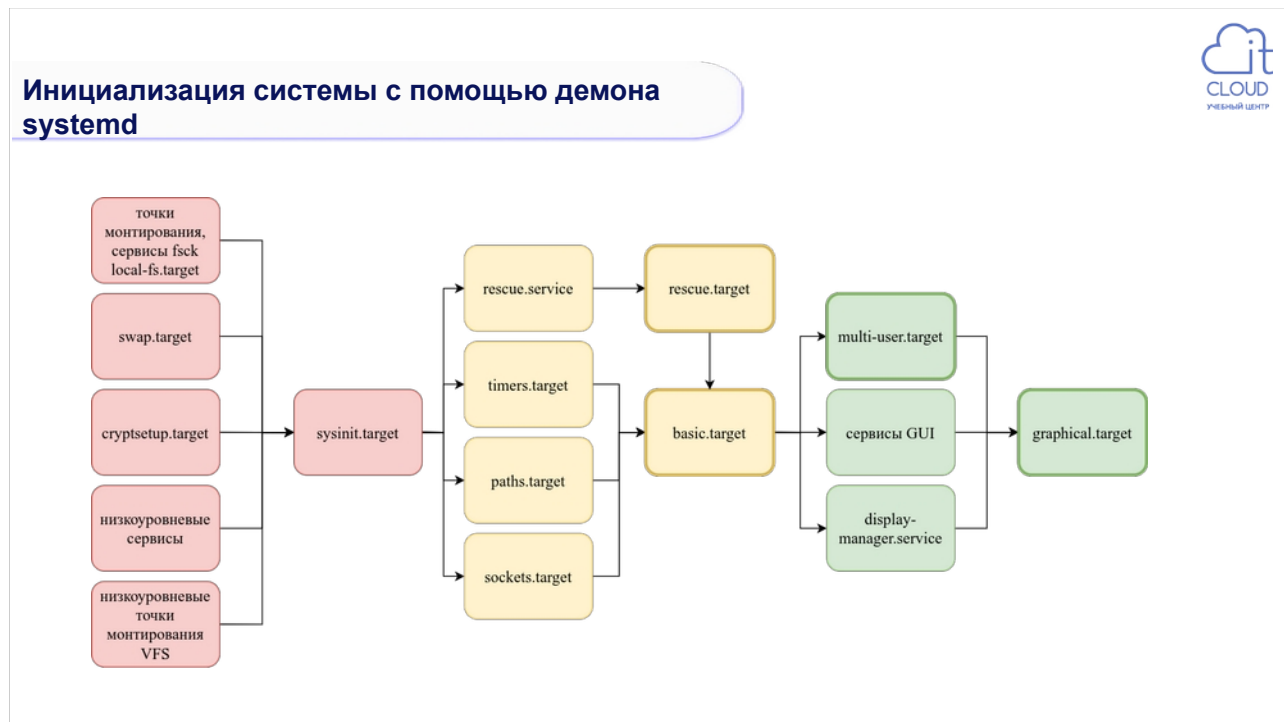
Настоятельно **не рекомендуется** изменять файл конфигурации вручную. Вместо этого имеется специальная команда `grub2-mkconfig` (иногда `grub-mkconfig`)

`grub2-mkconfig` — создает конфигурацию на основе заготовок в каталоге `/etc/grub.d/`, содержимого каталога `/boot` и `/etc/default/grub`

Если необходимо поменять параметры загрузки ядер Linux, то изменения необходимо вносить в файл `/etc/default/grub`

Дополнительные пункты меню загрузки определяются в файле `/etc/grub.d/40_custom`

3.3. Инициализация системы с помощью демона systemd



В стандартном подходе к инициализации Unix-подобных систем используется демон `init`, который запускается первым и получает `PID=1`, затем `init` запускает различные сервисы, в том числе сценарии инициализации `rc`

Такой подход имеет ряд ограничений:

1. Нет обратной связи между сервисом и демоном `init`. Демон `init` может только отслеживать работает или не работает запущенный им сервис (скрипт).
2. Нет возможности посмотреть результат загрузки сервисов после старта системы.
3. Монтирование, запуск служб, запуск сетевых сокетов, запуск виртуальных машин и т. д. в системах с `init` инициализацией это отдельные задачи, которые решаются разными средствами.
4. Не возможно параллельно запустить несколько сервисов, отложить запуск или запустить в фоновом режиме службу.

Systemd предлагает универсальное средство для инициализации, управления и настройки ОС.

Как правило программа `init` является ссылкой на `systemd` в таких системах.

Различные режимы загрузки

- В systemd отказались от понятия уровень исполнения вместо него используется понятие цель (target)
- Цель это некоторое состояние в которое должна попасть ОС.
- Одни цели могут зависеть от других.
- Цели так же зависят от других юнитов (служб, сокетов, монтирований и т.д.).

В systemd отказались от понятия уровень исполнения, вместо этого используется понятие цель (target).

Цель — некоторое состояние в которое система должна прийти после инициализации нужных сервисов.

Зависимости целей

```
systemctl list-dependencies default.target | grep target
default.target
• └─multi-user.target
•   └─basic.target
•       └─paths.target
•           └─slices.target
•               └─sockets.target
•                   └─sysinit.target
•                       └─cryptsetup.target
•                           └─local-fs.target
•                               └─swap.target
•                                   └─timers.target
• └─getty.target
• └─nfs-client.target
•     └─remote-fs-pre.target
•         └─remote-fs.target
•             └─nfs-client.target
•                 └─remote-fs-pre.target
```

Не все цели предназначены для конечной загрузки.

Основополагающее понятие systemd — **unit**.



Unit

- Основополагающее понятие systemd — unit
- Unit — отдельный элемент для описания и управления в systemd
- С каждым юнитом может быть связан конфигурационный файл, который должен находиться в одном из каталогов:
 1. /etc/systemd/system
 2. /run/systemd/system
 3. /usr/lib/systemd/system

Unit — отдельный элемент для описания и управления в systemd.

С каждым юнитом может быть связан конфигурационный файл, который должен находиться в одном из каталогов.

1. /etc/systemd/system
2. /run/systemd/system
3. /usr/lib/systemd/system

Каталоги перечислены в порядке убывания приоритета.

Локальные изменения следует проводить в каталоге /etc/systemd/system

Юниты делятся на типы:

1. `target` — цель, используется для описания некоторого состояния системы, в которое мы должны попасть;
2. `service` — описывает процесс, который контролирует systemd;
3. `socket` — сетевой сокет, для каждого сокета должен существовать соответствующий ему сервис;
4. `timer` — описывает какой сервис должен запуститься в определенное время;
5. `device` — создает файл устройства;
6. `mount` — точка монтирования, дополнительная к /etc/fstab;
7. `automount` — то же, что mount, но монтирование производится по требованию;
8. `swap` — область подкачки;
9. `slice` — описание виртуальной машины или контейнера;
10. `snapshot` — юнит без юнит-файла, позволяет создать копию конфигурации и затем

Глава 3. Процесс загрузки и уровни выполнения.

вернуться к ней;

11. `scope` — юнит без юнит-файла, создается программно для группировки процессов.



systemctl

- Команда `systemctl` управляет работой демона `systemd`

Команда `systemctl` управляет работой демона `systemd`

1. `systemctl list-dependencies` — показать от чего зависит юнит.
2. `systemctl list-units` — показывает список известных (включенных) юнитов.
3. `systemctl list-unit-files` — показывает список всех юнитов, в том числе выключенных, и их состояние.
4. `systemctl get-default` — показывает цель загрузки по умолчанию.
5. `systemctl set-default target` — устанавливает цель загрузки по умолчанию.
6. `systemctl {enable|disable} unit` — включение (разрешение на запуск) или выключение (запрет) юнита.

Примечание: некоторые сервисы все равно запускаются несмотря на запрет, т. к. это могут потребовать зависимые сервисы.

7. `systemctl {start|stop} unit` — запуск или остановка юнита.

Когда запускается `systemd` он определяет цель по умолчанию (`default.target`), заданную в настройках системы или во время загрузки ядра (параметр ядра `systemd.unit=нужная_цель.target`)

```
$ ls -l /etc/systemd/system/default.target
lrwxrwxrwx 1 root root 40 Nov 27 2016 /etc/systemd/system/default.target ->
/usr/lib/systemd/system/graphical.target
```

```
$ systemctl get-default
graphical.target
```

После определения цели для загрузки определяется последовательность запуска юнитов:

```
$ systemctl list-dependencies | grep target
default.target
• └─multi-user.target
•   └─basic.target
```

Глава 3. Процесс загрузки и уровни выполнения.

- |—paths.target
- |—slices.target
- |—sockets.target
- |—sysinit.target
- |—cryptsetup.target
- |—local-fs.target
- |—swap.target
- |—timers.target
- |—getty.target
- |—nfs-client.target
- |—remote-fs-pre.target
- |—remote-fs.target
- |—nfs-client.target
- |—remote-fs-pre.target

```
$ systemctl list-dependencies swap.target
```

```
swap.target
```

- |—dev-disk-by\x2duuid-1496f7c8\x2d3bb8\x2d4395\x2d97f3\x2d31f8a6083d9f.swap
- |—dev-disk-by\x2duuid-4cc362d5\x2db46d\x2d4e20\x2d8d75\x2dc4225eba0ea9.swap
- |—dev-zram0.swap

Подробнее это можно почитать в `man 7 bootup`.

Когда все нужные юниты будут запущены, то система считается работающей:

```
$ systemctl is-system-running
running
```

Если имеются проблемы, то мы увидим состояние `degraded`. Чтобы выяснить, что не так можно выполнить команду `systemctl --failed`:

```
$ systemctl is-system-running
```

```
degraded
```

```
$ systemctl --failed
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
• kdump.service	loaded	failed	failed	Crash recovery kernel arming

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of SUB.

SUB = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass `--all` to see loaded but inactive units, too.
To show all installed unit files use `'systemctl list-unit-files'`.

Попробуем понять в чем дело:

```
$ systemctl status kdump.service
```

```
• kdump.service - Crash recovery kernel arming
```

```
Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: enabled)
```

```
Active: failed (Result: exit-code) since Fri 2021-02-05 15:08:54 +05; 1h 57min ago
```

```
Process: 957 ExecStart=/usr/bin/kdumpctl start (code=exited, status=1/FAILURE)
Main PID: 957 (code=exited, status=1/FAILURE)
```

```
Feb 05 15:08:54 cent8-stream systemd[1]: Starting Crash recovery kernel arming...
```

```
Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: No memory reserved for crash kernel
```

Глава 3. Процесс загрузки и уровни выполнения.

```
Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: Starting kdump: [FAILED]
Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Main process exited,
code=exited, status=1/FAILURE
Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Failed with result
'exit-code'.
Feb 05 15:08:54 cent8-stream systemd[1]: Failed to start Crash recovery kernel
arming.
```

Теперь у нас есть причина: No memory reserved for crash kernel. Чтобы исправить мы можем или установить параметр ядра crashkernel=auto или просто запретить запуск службы при старте системы:

```
$ sudo systemctl disable kdump
Removed /etc/systemd/system/multi-user.target.wants/kdump.service.
$ sudo systemctl stop kdump
```

В systemd есть три состояния у юнита:

- 1.enabled – стартовать зависимость автоматически
- 2.disabled – не запускать автоматически, но вручную можно
- 3.masked – никогда не запускать юнит.

Systemd имеет свою особую систему журналов работы, как, собственно, systemd, так и всех служб им запущенных.

Просмотр журналов systemd производится командой journalctl.

Примечание: Описание команды будет в главе посвященной системным журналам.

3.4. Остановка и перезагрузка системы.



Остановка и перезагрузка системы

- Команды для остановки и перезагрузки
 - `systemctl` – основная команда управления питанием, остальные являются ссылками
 - `init`
 - `halt poweroff reboot`
 - `shutdown`
- ```
root@debian12:~# ls -l /usr/sbin/init
lrwxrwxrwx 1 root root 20 сен 20 2023 /usr/sbin/init -> /lib/systemd/systemd
root@debian12:~# ls -l /usr/sbin/shutdown
lrwxrwxrwx 1 root root 14 сен 20 2023 /usr/sbin/shutdown -> /bin/systemctl
root@debian12:~# ls -l /usr/sbin/halt
lrwxrwxrwx 1 root root 14 сен 20 2023 /usr/sbin/halt -> /bin/systemctl
```

Для немедленной остановки или перезагрузки системы можно использовать команды, соответственно, `/sbin/init 0` или `/sbin/init 6`. Однако для этого удобнее использовать команды `halt` для остановки или `reboot` для перезагрузки.

Команда `halt`

1. вносит в файл `/var/log/wtmp` запись о том, что система была остановлена в это время.

2. Далее для остановки вызывается команда `shutdown -h now`, останавливающая систему.

Опция `-f` (force) команды `halt`, заставляет систему остановиться без вызова `shutdown`.

Если команда `halt` вызвана с опцией `-n`, то перед остановом не будет произведена операция сброса содержимого кэша на диск.

При использовании команды `halt -d` запись в файл `/var/log/wtmp` произведена не будет.

Остановка системы с последующим отключением питания будет произведена в результате выполнения команд `halt -p` или `poweroff`.

*Примечание:* Обычно команды `poweroff` и `reboot` реализованы в виде символических ссылок на файл команды `/sbin/halt`.

Основной командой для безопасной остановки или перезагрузки системы является `/sbin/shutdown`.

С помощью нее можно осуществлять как немедленную, так и отложенную остановку системы.

### Глава 3. Процесс загрузки и уровни выполнения.

Эта команда посылает пользователям предупреждение о том, что система останавливается. Процессам, работающим в этот момент, посылается сигнал SIGTERM, получив который приложения могут корректно завершить свою работу.

Для останова системы команда shutdown посылает процессу init для перехода на 0-й или 6-й уровень исполнения при вызове с опцией, соответственно, -h или -r.

#### **Пример:**

```
/sbin/shutdown -h now
```

*Примечание: Данная команда осуществит немедленную остановку системы, так как в качестве времени останова системы указан параметр now. Если же необходимо остановить или перезагрузить систему в заданное время, то его следует указать в качестве аргумента:*

```
/sbin/shutdown -r 17:00 'System will be rebooted at 17:00!'
```

*В этом примере перезагрузка (-r) системы будет произведена в 17:00, причем пользователи будут оповещены об этом с помощью строки сообщения, указанной в качестве аргумента.*

Вместо использования точного указания времени можно указывать время задержки перед остановом. Если задержка измеряется секундами, то количество секунд следует указать после опции -t.

Количество минут задается опцией +n где n количество минут

#### **Пример:**

```
/sbin/shutdown -h +10
```

*Примечание: В данном случае останов будет предпринят через 10 минут. Реально между этими двумя вариантами задания задержки существует разница: после опции -t задается время задержки в секундах до того, как shutdown передаст сигнал init для перехода на другой уровень исполнения. Если же используется указание либо времени, либо задержки в минутах, то при этом реальное действие самой команды shutdown будет произведено с заданной задержкой. При этом пользователи, вошедшие в сеанс могут продолжать работать до начала останова, но новые сеансы не будут открыты.*

В таблице, приведенной ниже, указаны часто используемые опции команды shutdown.

| Опция | Назначение                                                                                     |
|-------|------------------------------------------------------------------------------------------------|
| -c    | Отменить начавшийся останов системы.                                                           |
| -f    | создает файл /fastboot, наличие которого позволяет не проверять файловую систему при загрузке. |
| -F    | создает файл /forcefsck, наличие которого вынуждает проверять файловую систему при загрузке.   |
| -h    | Остановка системы.                                                                             |
| -r    | Перезагрузить систему.                                                                         |
| -k    | Послать пользователям сообщение, но не останавливать систему.                                  |

В системах с `systemd` выключением и перезагрузкой системы управляет сам `systemd`, командой `systemctl`:

1. `systemctl reboot` — перезагрузка;
2. `systemctl halt` — остановка;
3. `systemctl poweroff` — выключение питания.