Загрузка Linux

Загрузка Linux. GRUB, консоль grub, grub-rescue. Init. Представления о SysV Init, Upstart, Systemd. Работа с systemd

[Введение](#_c713xnuw3xem)

[1. BIOS или UEFI](#_o9iw37xs0vpf)

[2. MBR или GPT](#_usw1xwytymab)

[MBR](#_a72xcjyc59v3)

[GPT](#_jy7ef8shkr1y)

[3. Загрузчик GRUB](#_vqnwkm1ttzvn)

[Консоль GRUB](#_d9kiyrt9v095)

[Работа в grub rescue](#_dzhuv5jti6r7)

[4. Ядро Linux](#_je176xh2wo4p)

[5. Init](#_xo2gd0anfqwj)

[SysV Init](#_otrw59lzkxi)

[Upstart](#_38qgz3449cbt)

[Systemd](#_smz65c8e2c2c)

[Управление сервисами](#_x04p1kyqt72f)

[Пример 1. Изменить число терминалов tty](#_5zelxib9w9m8)

[Пример 2. Простейший сервис](#_hx63wj7rmc5s)

[Пример 3. Активация по сокету](#_gvdiyy5lxf4u)

[Работа с логами](#_8j9wg2pijz9i)

[6. Уровни выполнения](#_m4vorllb75tx)

Практическое [задание](#_3tfrjxxltv85)

[Дополнительные материалы](#_jeeoh8oz8lfe)

[Используемая литература](#_uvp6qax5r1ok)

# 

# 

# Введение

От нажатия кнопки Включить до возможности выполнить первую команду или запустить первое приложение на любой машине при использовании любой операционной системы происходит несколько этапов. GNU/Linux — не исключение.

Традиционная ситуация, которая имела место быть долгое время, представляла следующую картину.

1. BIOS — базовая система ввода-вывода. При нажатии кнопки питания подаётся питание на материнскую плату либо при управлении питанием ACPI на отдельные узлы. Процессор начинает читать код, прошитый в ПЗУ, выполняет программу первичной диагностики POST, инициализирует базовую систему ввода-вывода, после чего читает настройки из энергонезависимой памяти (питаемой от батарейки, установленной на системной плате). Выяснив порядок загрузки, BIOS считывает начальную загрузочную запись диска в память и передает ей управление.
2. MBR — начальная загрузочная запись. Содержит данные о физическом носителе, файловой системе и программу загрузки. Её задача — передать управление загрузчику. (В редких случаях загрузчик не требовался. Так в DOS управление передавалось файлу io.sys размещавшемуся в начале раздела.)
3. Загрузчик — его задача обнаружить ядро операционной системы и передать ему управление. ntldr — для Windows, lilo, grub — для linux.
4. Ядро операционной системы — самый важный компонент, работающий в привилегированном режиме процессора, имеющий монопольный доступ к оборудованию и ко всем компонентам ЭВМ. Ядро запускает процессы пользовательского окружения и уровня ядра. Для linux первым процессом пользовательского пространства будет init, а первым процессом ядра — [kthreadd]. Каждый из них является родителем для всех остальных процессов, соответственно init — для процессов пользовательского пространства, [kthreadd] — процессов ядра.
5. В linux процесс Init является родителем для всех остальных процессов. Также Init управляет процессами пользовательского ядра, в особенности это касается процессов-демонов, постоянно выполняющихся в фоновом режиме.
6. В зависимости от уровней выполнения запускаются те или иные скрипты, конфигурирующие, запускающие или останавливающие соответствующие демоны.

Так выглядела классическая схема запуска операционной системы. Разберём теперь, как это работает сейчас.

# 1. BIOS или UEFI

BIOS – Basic Input Output System. Базовая система ввода-вывода.

Изначально при поступлении питания на процессор выполняются команды из ROM. Это программа самотестирования POST, инициализация прерываний BIOS, считывание параметров из энергонезависимой памяти BIOS (фактически разновидность оперативной памяти, питаемая постоянно от батарейки, используется системным таймером, также хранит ряд настроек системы, в том числе порядок загрузки и указание, с какого диска грузится).

Несмотря на современные 32-битные и даже 64-битные процессоры BIOS (так же как и MS DOS) выглядело довольно архаично: 16-битный код, адресация 1 Мб памяти, исполнение в реальном режиме 8086. Потребность в использовании ресурсов в полной мере привела к созданию EFI – Extensible Firmware Interface (расширяемый интерфейс прошивки).

В настоящее время стандарт развивается как UEFI – Unified Extensible Firmware Interface.

Так же, как BIOS предоставляло системные вызовы для базовых операций с вводом/выводом, UEFI предоставляет платформонезависимую среду драйверов, которую операционная система может использовать для использования текста и графики до загрузки платформозависимых драйверов.

UEFI поддерживает механизмы загрузки операционных систем, в том числе и с GPT-дисков (если в MBR – Main Boot Record в первом секторе диска должен был находиться загрузчик, в UEFI уже содержится загрузчик, кроме того, таблица разделов GTP позволяет адресовать более 2,2 Тб).

Помимо этого, UEFI поддерживает механизмы расширений и оболочку, наподобие COMMAND.COM с возможностью выполнения скриптов.

Как часто бывает, новые технологии могут порождать и новые проблемы. Так на компьютерах, поставляемых с ОС Windows, в UEFI включен по умолчанию режим Secure Boot. Этот режим предотвращает загрузку операционных систем, если они не подписаны ключом, загруженным в UEFI. В таком случае можно отключить режима Secure Boot либо добавить в прошивку UEFI подписанный ключ или выбрать дистрибутив Linux, в котором поддерживается режим Secure Boot, например, Ubuntu.

Как можно понять из вышесказанного, UEFI представляет собой специфическую операционную систему (впрочем, ничего удивительного, традиция начинается ещё с прошивки в ROM BASIC в эпоху микрокомпьютеров).

# 2. MBR или GPT

## MBR

MBR – Main Boot Record (главная загрузочная запись). Содержит исполняемый код, необходимый для передачи управления загрузчик и таблицу разделов (partition table). Может содержать только 4 первичных (primary) раздела, а при необходимости большего количества разделов вместо одного из первичных разделов можно создавать расширенный раздел (extended partition). Такой раздел будет содержать внутри себя ещё несколько (до 16) разделов, называемых логическими (logical). Есть определённые ограничения, связанные с созданием разделов и установленных на нём ОС. Например, Windows необходимо обязательно устанавливать в первичный раздел, помеченный как активный (загрузочный). Для разделов Linux таких ограничений сейчас нет.

Также MBR имеет существенное ограничение, что поддерживается размер дисков только до 2Тб. В GPT нет такого ограничения.

## GPT

GUID Partition Table. Больше не содержит загружаемый код, эти функции отнесены к UEFI. Более того, GPT – является частью стандарта UEFI. (Тем не менее блок MBR все равно присутствует в начале для совместимости и защиты от повреждения утилитами, не умеющими работать с GPT, но понимающими MBR. В совместимом MBR указан один раздел, охватывающий весь диск.)

GPT не накладывает ограничений на разделы, поэтому понятия как расширенные и первичные разделы не используются.

fdisk не будет работать с gpt, вместо него следует использовать gdisk.

# 3. Загрузчик GRUB

Как правило, BIOS передаёт управление загрузочной записи MBR, а та — загрузчику операционной системы (или даже непосредственно ядру операционной системы, как в MS DOS, почему файлы io.sys и msdos.sys должны были располагаться в начале диска). Для windows NT/2000/XP и т.д. это ntldr, для Windows 3.11,95,98 роль загрузчика играл MS DOS. Для Linux применяются загрузчики GRUB и Lilo. Существует также загрузчик Loadlin, позволяющий загрузить Linux из MS DOS (не как приложение MS DOS, перед загрузкой Линукса MS DOS будет выгружен из памяти).

Но наибольшую популярность получил загрузчик GRUB — GRand Unified Bootloader.

## Консоль GRUB

В случае, если меню grub скрыто, то нужно при запуске системы зажать Shift. После этого появляется скрытое меню grub, возможность запустить memtest, нажав e отредактировать и нажав c перейти в консоль.



GRUB2 имеет несколько конфигурационных файлов, позволяет загружать несколько операционных систем, настраивать цвет текста и шрифта, показывать при загрузке splash-экран, отображать или отключать меню выбора при загрузке и т.д.

В меню по умолчанию доступны загрузка ОС, а также Memtest (уже упоминалось, программа, не требующая операционной системы, для тестирования RAM). Здесь же могут присутствовать новые пункты, например, загрузка Windows.

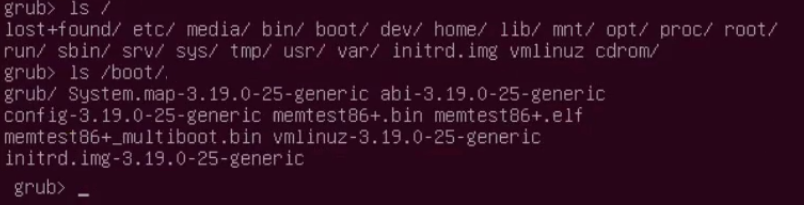
Можно нажать «e» для редактирования команд перед загрузкой или перейти в режим консоли «c».

Нажмем «c».



GRUB так и сообщит, что это bash-подобная система. Работает автодополнение по TAB, вызов ранее введённых команд клавишей «вверх».

Можно набрать help, чтобы получить список команд. Среди прочего есть команды ls, cat. GRUB понимает файловые системы ext2 и ext4 (но не понимает LVM, в этом случае обычно создаётся отдельный раздел на ext2 вне томов LVM и система загружается с него).



Обратите внимание на файлы, содержащиеся в разделе /boot, сравните со стартовым меню. Попробуйте самостоятельно с помощью cat посмотреть содержимое одного из системных файлов (например, /etc/shadow), сделайте выводы.

Особым образом работает команда ls без параметров.



Обратите внимание, что система именования дисков привычна от имеющейся в Linux (/dev/sda, /dev/sda1, /dev/sda2, /dev/sdb1, /dev/sdb2).

(hd0) соответствует диску /dev/sda.

(hd0,msdos5) означает 5 раздел на диске hd0 (/dev/sda), соответствует /dev/sda5. Это логический раздел, очевидно для swap.

(hd0,msdos1) означает 1 раздел на диске hd0 (/dev/sda), соответствует /dev/sda1. Это первичный раздел, очевидно, он будет монтироваться как корень (/) в Linux.

В консоли GRUB мы тоже имеем некий аналог монтирования диска в корень. Только по большей части это не столько монтирование, сколько указание значения переменной.

Добавьте в виртуальной машине второй диск (/dev/sdb), создайте раздел (/dev/sdb1) и отформатируйте его в ext2 или ext4. Создайте несколько файлов.

В Linux.

Создаём раздел.

|  |
| --- |
| # fdisk /dev/sdb  Команда (m для справки) : m  Команда (m для справки) : p  Команда (m для справки) : n  Select (default p): p  Команда (m для справки) : p  Команда (m для справки) : w |

Некоторые команды fdisk.

* m – справка.
* p – вывести перечень разделов.
* n – создать новый раздел (требуется указать, p – первичный, e – расширенный).
* w – сохранить изменения.
* q – выход без сохранений.

Форматируем раздел.

|  |
| --- |
| # mkfs -t ext4 /dev/sdb1  # mkdir /mnt/diskb  # mount /dev/sdb1 /mnt/diskb |

Самостоятельно создайте в /mnt/diskb один или несколько каталогов и файлов (или скопируйте).

В GRUB.

|  |
| --- |
| grub> set root=(hd1,msdos1)  grub> ls  grub> set root=(hd0,msdos1)  grub> ls |

Сравните результаты вывода ls.

Процесс загрузки.

Обратите внимание, что root должен быть установлен на тот раздел, с которого вы будете грузиться. Если вы не меняли, дополнительно его устанавливать нет нужды, он указан по умолчанию.

Для загрузки понадобится следующее.

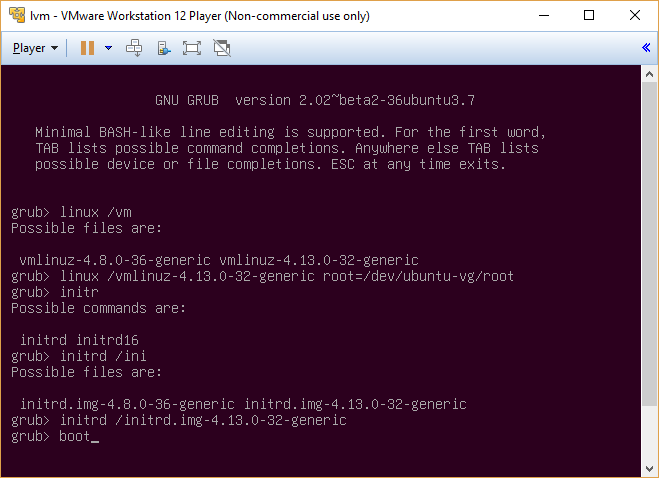
* Загрузить в память ядро командой linux.
* Загрузить в память образ виртуальной файловой системы командой initrd.
* Стартовать загрузку командой boot.

|  |
| --- |
| grub> linux /boot/vmlinuz-3.19.0-25-generic root=/dev/sda1  grub> initrd /boot/initrd.img-3.19.0-25-generic  grub> boot |

Важно: версии ядра и образа initrd должны совпадать.

Для запуска в однопользовательском режиме для linux также необходимо указать параметр single.

Обратите внимание, если вы не правильно укажите root, то система полностью не загрузится, так как не сможет смонтировать корень. Если вы используете lvm, то это также надо учитывать, см. рисунок.



В случае, если передумали загружаться, можно остановить компьютер командой halt или вызвать перезагрузку командой reboot.

## Работа в grub rescue

Если grub не смог загрузиться, будет запущена упрощенная оболочка grub resque. В ней нет истории команд и автодополнения, тем не менее, многие команды работают.

Простейший пример: переименовать директорию /boot/grub в /boot/grub2. В таком случае после сообщения об ошибке появится grub rescue, после чего можно попробовать загрузиться следующими командами.

|  |
| --- |
| set prefix=(hd1,msdos1)/grub2  insmod normal  normal |

Иногда имеет смысл загрузиться вручную с Live-CD и переконфигурировать grub.

|  |
| --- |
| # grub-mkdevicemap  # update-grub2 |

# 4. Ядро Linux

Когда образ ядра и образ ядра initrd загружен в память командой linux, после команды boot управление передаётся ядру. Ядро стартует, распаковывает образ initrd в память, монтирует, после чего загружает необходимые модули ядра. Можно не использовать initrd, если самостоятельно скомпилировать ядро с драйверами для нужного оборудования. Иногда так делают для увеличения скорости загрузки. В случае ошибки (например, если не указать файловую систему для монтирования в корень) будет запущена initframes. В этом случае останется доступным RAM-диск.

Init RAM disk – образ файловой системы, который распаковывается в RAM-диск и монтируется в корень. Необходим для загрузки модулей и выполнения простейших скриптов. В настоящее время образ initrd представляет собой архив cpio сжатый gzip.

Причина, по которой используется initrd, напоминает проблему курицы и яйца. Для того, чтобы linux могло примонтировать файловую систему и начать работу, ей необходимы модули ядра, а чтобы загрузить модули ядра, ей необходима файловая система. Именно поэтому и используется образ файловой системы с необходимыми модулями ядра и другими полезными вещами, распаковываемую в память. Чтобы лучше понять, как это работает, можно распаковать образ initrd и изучить его содержимое.

Распаковать и изучить образ можно следующим способом.

|  |
| --- |
| mkdir /tmp/initrd&cd /tmp/initrd  cp /boot/initrd.img-3.19.0-25-generic /tmp/initrd/img.cpio.gz  gzip -d img.cpio.gz  cpio -i < img.cpio  ls  mc |

# 5. Init

После того, как драйвера (модули ядра) загружены и корень диска будет смонтирован, запускаются процессы init и поток ядра kthreadd. При этом kthreadd служит родителем для потоков ядра, а init запускает все программы, работающие в пользовательском окружении.

Стоит отметить, что Линус Торвальдс писал linux, в большей степени основываясь на UNIX System V, но с некоторыми оглядками на BSD (например, реализация сокетов Беркли). Таким образом, первой системой инициализации была SysV Init. Она обладала рядом недостатков, в частности, последовательной работой и отсутствием обратной связи от процессов. В результате появились новые системы такие, как Upstart и Systemd. Последняя не unix-way, но практически доминирует в новых дистрибутивах GNU/Linux.

SysV Init/Upstart/Systemd образуют определенный аспект различий между разными линуксами. Одни и те же вещи делаются по-разному и задаются разными конфигурационными файлами. В целом такие различия могут всплывать и между разными версиями одной ОС. В целом различия в конфигурационных сервисах и файлах могут меняться от Линукса к Линуксу. Это необходимо иметь ввиду при переходе между основанных на Red Hat системах (Fedora, Centos и т.д.) и основанных на Debian (в т.ч. Ubuntu и её клоны).

При этом вещи выполняются одни и те же. Стартуют сервисы (демоны) такие, как cron, syslog, запускаются службы терминалов и т.д.

## SysV Init

Классическая, но ныне устаревшая система инициализации использует концепцию семи уровней исполнения.

Узнать, на каком уровне вы находитесь, можно выполнив одну из нижеперечисленных команд.

|  |
| --- |
| $ runlevel  $ who -r |

Значительную роль в системе играл файл /etc/inittab, чем-то схожий с crontab.

Строки этого файла управляли запуском сервисов.

|  |
| --- |
| id:5:initdefault: si::sysinit:/etc/rc.d/rc.sysinit l0:0:wait:/etc/rc.d/rc 0 l1:1:wait:/etc/rc.d/rc 1 l2:2:wait:/etc/rc.d/rc 2 l3:3:wait:/etc/rc.d/rc 3 l4:4:wait:/etc/rc.d/rc 4 l5:5:wait:/etc/rc.d/rc 5 l6:6:wait:/etc/rc.d/rc 6 1:2345:respawn:/sbin/mingetty tty1 2:2345:respawn:/sbin/mingetty tty2 3:2345:respawn:/sbin/mingetty tty3 4:2345:respawn:/sbin/mingetty tty4 5:2345:respawn:/sbin/mingetty tty5 6:2345:respawn:/sbin/mingetty tty6 x:5:respawn:/etc/X11/prefdm -nodaemon |

Обратите внимание на строки, начинающиеся с l1..l6, — это инициализация уровней выполнения.

Строки с 1 по 6 — инициализация терминалов, с 1 по 6, 2345 — перечисляет, на каких уровнях выполнения стартуют. x — запуск X-Server.

respawn — означает перезапустить сервис, если он будет остановлен (например, в случае ошибки или в результате принудительного kill -9).

Управление сервисами в SysV Init (будет работать также и в Upstart и Systemd для совместимости).

## Upstart

По прежнему поддерживаются механизмы SysV Init, но inittab уже нет. Вместо него присутствуют специфические скрипты /etc/rcX.d и конфигурационные файлы с вкраплениями скриптов в /etc/init.d

Для управления сервисами помимо service есть встроенные средства.

Запустить сервис.

|  |
| --- |
| # start ssh |

Остановить сервис.

|  |
| --- |
| #stop ssh |

Перезапустить сервис.

|  |
| --- |
| #restart ssh |

Узнать статус сервиса.

|  |
| --- |
| #status ssh |

Пример 1. Изменить число терминалов tty.

Нажимаем Alt-F8 (или Ctrl-Alt-F8, если работаем в X-Server), убеждаемся, что tty8 не доступен.

В директории /etc/init копируем tty1.conf в tty8.conf.

|  |
| --- |
| # cp tty1.conf tty8.conf |

Правим tty8.conf.

|  |
| --- |
| mcedit tty8.conf |

|  |
| --- |
| # **tty8** - getty  #  # This service maintains a getty on tty1 from the point the system is  # started until it is shut down again.  start on stopped rc RUNLEVEL=[2345] and (  not-container or  container CONTAINER=lxc or  container CONTAINER=lxc-libvirt)  stop on runlevel [!2345]  respawn  exec /sbin/getty -8 38400 **tty8** |

Выполним команды.

|  |
| --- |
| # start tty8 |

Нажимаем Alt-F8 (или Ctrl-Alt-F8, если работаем в X-Server), убеждаемся, что tty8 доступен.

## Systemd

Systemd — новая реализация init, системный менеджер пользовательского пространства, пришедший на смену системам SysV init и Upstart и в последнее время потеснивший конкурентов во многих реализациях GNU/Linux. Systemd используется в Centos 7 и пришел на смену Upstart в Ubuntu 16.0 LTS.

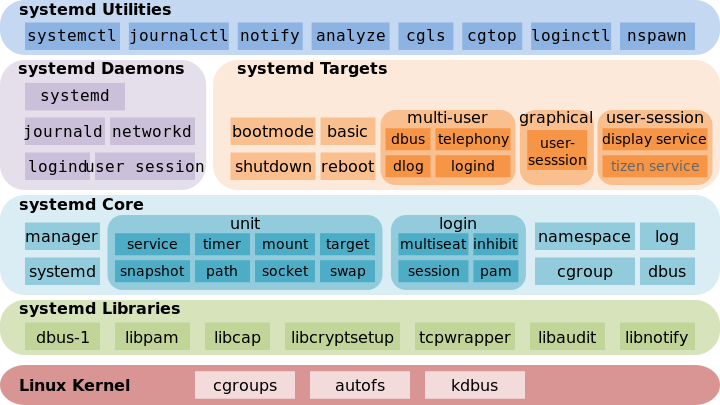
Systemd изначально разработан для Linux, так как опирается на возможности ядра Linux такие, как cgroups (используется для отслеживания демонов вместо PID, поэтому связь с демоном не будет утрачена после ветвления процессов). Systemd не вполне соответствует концепции unix-way, тем не менее обладает широкими возможностями, а кроме того, довольно прозрачным и понятным механизмом управления сервисами, значительно превосходящим аналогичные у upstart.

По сравнению с SysV Init, запускающим процессы строго последовательно, служба systemd осуществляет распараллеливание процессов при запуске, а также умеет запускать сервисы по требованию и некоторыми другими интересными возможностями.

В отличие от upstart с его конфигами с вкраплениями скриптов в Systemd используется прозрачная и лаконичная концепция юнитов. Юнит — это конфигурационный файл, по структуре напоминающий ini-файлы. При этом юниты существуют разных типов, например.

* service — служит для запуска сервиса.
* target — возможность группировать юниты, в частности такая концепция в systemd используется вместо концепции уровней выполнения.
* socket — активация сервиса по сокету.
* device — активация сервиса при подключении устройства.
* path — активация сервиса по изменению (созданию) файла или директории.
* mount — механизм для монтирования/размонтирования.
* automount — позволяет автоматически использовать механизм юнитов mount при обращении к указанному пути. Пока обращений к выбранной директории нет, устройство остается размонтированным, при попытке перейти в директорию, прочитать оглавление и т.д. директория монтируется автоматически.

Systemd обладает многими интересными возможностями, позволяющими обойтись даже без mount, /etc/fstab, cron, использовать journald вместо (или совместно с) rsyslog (хотя далеко не всегда используются все возможности systemd, и по-прежнему применяются те или иные средства такие, как cron). Имеются возможности запуска сервисов по таймеру, возможность управления сgroup-контейнерами, есть механизм создания снэпшотов (состояний юнитов и возможность их восстановления).



Архитектура Systemd.

Автор: Shmuel Csaba Otto Traian, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=28698339

### Управление сервисами

По прежнему работает механизм из SysV Init, но на самом деле это просто редирект. Сравните.

|  |
| --- |
| $ service ssh status  $ systemctl status ssh |

А вот с Upstart совместимости нет. Команда status работать не будет.

Вывести список запущенных сервисов.

|  |
| --- |
| $ systemctl list-units --type service |

Вывести список сервисов, которые завершились с ошибкой.

|  |
| --- |
| $ systemctl list-units --type service --state failed |

Запустить сервис.

|  |
| --- |
| # systemctl start ssh.service |

Или.

|  |
| --- |
| # systemctl start ssh |

Остановить сервис.

|  |
| --- |
| # systemctl stop ssh.service |

Или.

|  |
| --- |
| # systemctl stop ssh |

Перезапустить сервис.

|  |
| --- |
| # systemctl restart ssh |

Перечитать конфигурацию сервиса.

|  |
| --- |
| # systemctl reload ssh |

Деактивировать сервис.

|  |
| --- |
| # systemctl disable ssh |

Активировать сервис.

|  |
| --- |
| # systemctl enable ssh |

Сделать это одновременно.

|  |
| --- |
| # systemctl reenable ssh |

### Пример 1. Изменить число терминалов tty

Нажимаем Alt-F8 (или Ctrl-Alt-F8, если работаем в X-Server), убеждаемся, что tty8 не доступен.

В директории /etc/systemd правим файл logind.conf.

|  |
| --- |
| # mcedit /etc/systemd/logind.conf |

Изменяем число терминалов. Если строчка закомментирована, раскомментируйте ее или добавьте новую.

|  |
| --- |
| NAutoVTS=8 |

Выполним команды.

|  |
| --- |
| # systemctl daemon-reload  # systemctl restart systemd-logind.service |

Нажимаем Alt-F8 (или Ctrl-Alt-F8, если работаем в X-Server), убеждаемся, что tty8 доступен.

### Пример 2. Простейший сервис

/opt/geektest.sh.

|  |
| --- |
| #!/bin/bash  exec &> /tmp/geektest.log  echo Starting geektest  cd /nonexisting  echo Stopping geektest |

/etc/systemd/system/geektest.service.

|  |
| --- |
| [Unit]  Description=geek test service  [Sevice]  ExecStart=/opt/geektest.sh  Type=oneshot |

### Пример 3. Активация по сокету

/etc/systemd/system/geeksock.socket.

|  |
| --- |
| [Unit]  Description=activation on socket test  [Socket]  ListenStream=6666  Accept=yes  [Install]  WantedBy=sockets.target |

/etc/systemd/system/geeksock@.service.

|  |
| --- |
| [Unit]  Description=socket activation service  [Sevice]  ExecStart=/bin/cat  StandardInput=socket |

### Работа с логами

journalctl — удобный механизм для просмотра логов. Нам уже знакомы less /var/log/syslog или tail -f /var/log/syslog.

Сравните со следующим.

|  |
| --- |
| $ journalctl -f |

Просмотр логов с момента текущей загрузки.

|  |
| --- |
| $ journalctl -f |

Просмотр логов за заданный период времени.

|  |
| --- |
| $ journalctl --since yesterday  $ journalctl --since 20:00 --until now  $ journalctl --since 19:00 --until "1 hour ago"  $ journalctl --since "2017-04-07 20:30:00"  $ journalctl --since "2017-04-07 20:30:00" --until "2017-04-08 15:25:00" |

Фильтрация логов по юниту.

|  |
| --- |
| $ journalctl -u ssh.service  $ journalctl -u ssh.service --since yesterday |

С помощью \_PID, \_UID, \_GID можно фильтровать логи по PID, идентификатору пользователя или группы.

|  |
| --- |
| $ ps ax|grep firefox  $ journalctl \_PID=3840  $ id -u developer  $ journalctl \_UID=1003 |

Можно фильтровать логи по пути.

|  |
| --- |
| $ journalctl /var/www |

# 6. Уровни выполнения

Традиционно выделяют 7 уровней выполнения (по число линий в коммутаторах телефонной линии), но фактически их может быть до 9.

* 0 – останов системы. Все сервисы завершаются.
* 1 – однопользовательский режим.
* S – псевдоним для 1.
* 2 – 3 многопользовательский режим. Иногда выделяют без поддержки сети, с поддержкой сети. Какой именно уровень выполнения – зависит от операционной системы.
* 4 – 5 графический режим. Обычно 5. В этом режиме стартует X-Server.

Идея уровней выполнения в том, чтобы в зависимости от того, на каком уровне выполнения система находится, одни сервисы запускать, другие останавливать. В SysV Init эта информация берется из inittab, в Upstart из /etc/rcX.d где X — номер уровня выполнения.

Перейти на уровень выполнения, например 1, можно с помощью команды.

|  |
| --- |
| # init 1 |

В Systemd уровни выполнения поддерживаются, но фактически это псевдонимы для набора зависимостей.

* poweroff.target (runlevel0.target) — завершение работы и отключение системы.
* rescue.target (runlevel1.target) — настройка консоли восстановления (sinlgle — однопользовательский режим).
* multi-user.target (runlevel2.target, runlevel3.target, runlevel4.target) — настройка неграфической (консольной) многопользовательской системы.
* graphical.target (runlevel5.target) — настройка графической многопользовательской системы (X Server).
* reboot.target (runlevel6.target) — перезагрузка системы.

## Системные журналы Linux. Демон rsyslog

Функция системного журналирования (т.н. "логи" или логирование) — ­это основной источник информации о работе системы и ошибках. Журналирование может осуществляться на локальной системе, а также сообщения журналирования могут пересылаться на удаленную систему, кроме того, в конфигурационном файле /etc/rsyslog.conf возможна тонкая настройка уровня (подробности) журналирования. Журналирование осуществляется при помощи демона rsyslogd, который обычно получает входную информацию при помощи UNIX-сокета /dev/log (локально) или с udp­-порта 514 (с удаленных машин).

В случае локального журналирования главным файлом ­ хранителем информации, обычно является /var/log/syslog (или /var/log/messages, например, в CentOS), но в большинстве инсталляций используются и многие другие файлы, которые настроены с помощью вышеуказанного конфигурационного файла.

## Конфигурационный файл syslog.conf

Файл rsyslog.conf является главным конфигурационным файлом для демона rsyslogd.

Синтаксис /etc/rsyslog.conf.

$переменная значение:

В файле мы можем увидеть такие строки.

|  |
| --- |
| ModLoad imuxsock # provides support for local system logging $ModLoad imklog *# provides kernel logging support* #$ModLoad immark *# provides --MARK-- message capability* # provides UDP syslog reception #$ModLoad imudp #$UDPServerRun 514 # provides TCP syslog reception #$ModLoad imtcp #$InputTCPServerRun 514 |

Здесь мы видим подключение модулей. Перечислим типы модулей.

* Модули ввода — предназначены для сбора информации из различных источников, название модулей начинаются с префикса im.
* Модули вывода — позволяют отправлять сообщения в файлы, или по сети, или в базу данных, название модулей начинаются с префикса om.
* Модули фильтрации — позволяют фильтровать сообщения по разным параметрам, название модулей начинаются с префикса fm.
* Модули парсинга — предоставляют расширенные возможности для синтаксического анализа сообщения, название модулей начинаются с префикса pm.

Затем следуют глобальные директивы.

|  |
| --- |
| ActionFileDefaultTemplate RSYSLOG\_TraditionalFileFormat |

Здесь мы указываем, что нужно использовать стандартный формат хранения времени в секундах с 1970 года.

Затем идет набор прав разрешений для файлов журналов, которые будут созданы в системе.

|  |
| --- |
| FileOwner root $FileGroup adm $FileCreateMode 0640 $DirCreateMode 0755 $Umask 0022 |

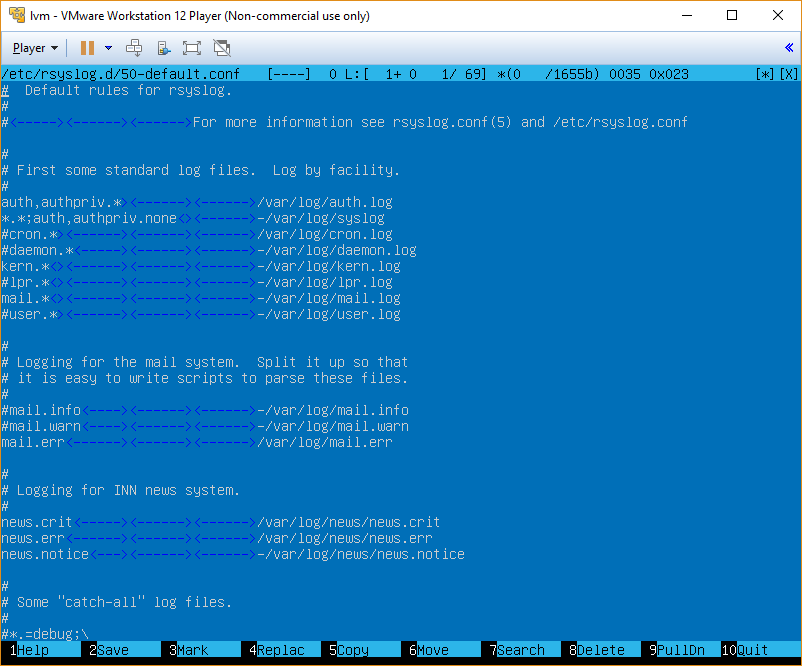
Далее подключаются все файлы из папки rsyslog.d с помощью директивы IncludeConfig.

|  |
| --- |
| IncludeConfig /etc/rsyslog.d/\*.conf |

В этих файлах могут содержаться дополнительные настройки

Включаемые файлы следуют общему формату файла rsyslog.conf и содержат набор правил. Каждое правило есть ­ строка, состоящая из селектора и действия, разделенных пробелом или табуляцией. Селектор представляет собой запись в виде источник.приоритет (источник иногда именуют ­ категорией) Селектор может состоять из нескольких записей источник.приоритет, разделенных символом ";" . Можно указывать несколько источников в одном селекторе (через запятую). Поле действие ­ устанавливает действие, выполняемое для журналирования селектора.

Получив сообщение для записи в журнал, syslogd для каждого правила проверяет не подходит ли сообщение под шаблон, определяемый селектором. Если подходит, то выполняется указанное в правиле действие. Для одного сообщения может быть выполнено произвольное количество действий (т.е. обработка сообщения не прекращается при первом успехе).



Пример /etc/rsyslog.d/50-default.conf, который инклудится в /etc/rsyslog.conf.

Сообщения с уровнем, равным или выше указанного в селекторе, и источником, равным указанному в селекторе, считается подходящим. Звездочка перед точкой соответствует любому источнику после точки ­ любому уровню. Слово none после точки ­ никакому уровню для данного источника. Можно указывать несколько источников в одном селекторе (через запятую).

Источник может быть следующим.

* 0­ kern ­ - сообщения ядра.
* 1­ user ­ - сообщения пользовательских программ.
* 2 ­ mail ­ - сообщения от почтовой системы.
* 3 ­ daemon ­ - сообщения от тех системных демонов, которые в отличие от FTP или LPR не имеют выделенных специально для них категорий.
* 4 ­ auth ­ - все, что связано с авторизацией пользователей, вроде login и su (безопасность/права доступа).
* 5 ­ syslog ­ - система протоколирования может протоколировать сообщения от самой себя.
* 6 ­ lpr ­ - сообщения от системы печати.
* 7 ­ news ­ - сообщения от сервера новостей(в настоящее время не используется).
* 8 ­ uucp ­ - сообщения от UNIX-­to­-UNIX Copy Protocol. Это часть истории UNIX и, вероятнее всего, она вам никогда не понадобится (хотя до сих пор определенная часть почтовых сообщений доставляется через UUCP).
* 9 ­ cron ­ - сообщения от системного планировщика.
* 10 ­ authpriv ­ - то же самое, что и auth, однако сообщения этой категории записываются в файл, который могут читать лишь некоторые пользователи (возможно, эта категория выделена, потому что принадлежащие ей сообщения могут содержать открытые пароли пользователей, которые не должны попадать на глаза посторонним людям, и следовательно файлы протоколов должны иметь соответствующие права доступа).
* 11 ­ ftp ­ - при помощи этой категории вы сможете сконфигурировать ваш FTP сервер, чтобы он записывал свои действия.
* 12 ­ NTP ­ - сообщения сервера времени
* 13 ­ log audit.
* 14 ­ log alert.
* 15 ­ clock daemon ­ - сообщения демона времени.
* с 16 по 23 local0 ­ local7 - зарезервированные категории для использования администратором системы. Категория local7 обычно используется для сообщений, генерируемых на этапе загрузки системы.
* mark (не имеющая цифрового эквивалента) ­ - присваивается отдельным сообщениям, формируемым самим демоном syslogd.

Под приоритет (степени важности) сообщений заданы 8 уровней важности, которые кодируются числами от 0 до 7.

* 0 ­ emerg (старое название PANIC) ­ Чрезвычайная ситуация. Система неработоспособна.
* 1 ­ alert ­ Тревога! Требуется немедленное вмешательство.
* 2 ­ crit ­ Критическая ошибка (критическое состояние).
* 3 ­ err (старое название ERROR) ­ Сообщение об ошибке.
* 4 ­ warning (старое название WARN) ­ Предупреждение.
* 5 ­ notice ­ Информация о каком-­то нормальном, но важном событии.
* 6 ­ info ­ Информационное сообщение.
* 7 ­ debug ­ Сообщения, формируемые в процессе отладки.

Согласно действию, указанному в правиле, сообщение может быть записано в следующие назначения.

* Обычный файл — задается полный путь. Поставьте перед ним дефис (-­), чтобы отменить синхронизацию файла после каждой записи. Это может привести к потере информации, но повысить производительность.
* Именованные каналы — размещение перед именем файла символа канала (|) позволит использовать fifo (first in — first out, первый пришел — первый вышел) или именованный канал (named pipe) в качестве приемника для сообщений. Прежде чем запускать (или перезапускать) syslogd, необходимо создать fifo при помощи команды mkfifo. Иногда fifo используются для отладки.
* Терминал и консоль Терминал такой, как /dev/console.
* Удаленная машина — чтобы сообщения пересылались на другой хост, поместите перед именем хоста символ (@). Обратите внимание, что сообщения не пересылаются с принимающего хоста. (для работы данного назначения на клиенте и сервере в файле /etc/services должна быть прописана строчка syslog 514/udp, и открыт UDP-­порт 514).
* Список пользователей — разделенный запятыми список пользователей, получающих сообщения (если пользователь зарегистрирован в системе). Сюда часто включается пользователь root.
* Все зарегистрированные пользователи — чтобы известить всех зарегистрированных пользователей при помощи команды wall, используйте символ звездочки (\*).

Как и во многих конфигурационных файлах, синтаксис следующий.

* Строки, начинающиеся с #, и пустые строки игнорируются.
* Символ \* может использоваться для указания всех категорий или всех приоритетов.
* Специальное ключевое слово none указывает, что журналирование для этой категории не должно быть выполнено для этого действия.
* Дефис перед именем файла указывает, что после каждой записи журнал не должен синхронизироваться. В случае аварии системы вы можете потерять информацию, но отключение синхронизации позволит повысить производительность.

В синтаксисе конфигурационного файла можно поставить перед приоритетом знак !, чтобы показать, что действие не должно применяться, начиная с этого уровня и выше. Подобным образом, перед приоритетом можно поставить знак =, чтобы показать, что правило применяется только к этому уровню, или !=, чтобы показать, что правило применяется ко всем уровням, кроме этого.

Для фильтрации логов могут использоваться не только источник и приоритет, но и более сложные выражения на основе условий и сравнений.

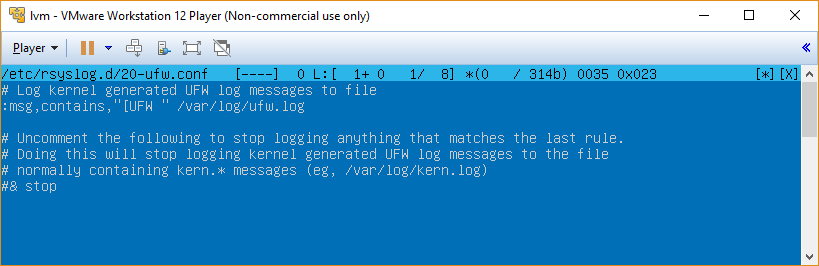
|  |
| --- |
| :поле, сравнение, "значение" путь\_к\_файлу |

В качестве операции сравнения вы можете использовать такие варианты.

* contains — поле содержит указанное значение.
* isequal — поле должно быть идентичным значению.
* startswith — поле должно начинаться со значения.
* regex — сравнивает поле с регулярным выражением.

Например, отфильтруем сообщения только от определенной программы.

|  |
| --- |
| :syslogtag, isequal, "someprogram:" /var/log/someprogram.log & stop |



Пример фильтрации логов в файле /etc/rsyslog.d/20-uvf.conf.

Также можно использовать более простой синтаксис в виде выражения if.

|  |
| --- |
| if $поле сравнение 'значение' then файл |

Пример.

|  |
| --- |
| if $syslogtag == 'someprogram' then /var/log/someprogram.log |

## Демон klogd

Демон klogd отвечает за журналирование событий, происходящих в ядре системы. Необходимость в отдельном демоне klogd объясняется тем, что ядро не может использовать стандартную функцию syslog. Дело в том, что стандартные библиотеки С (включая ту библиотеку, в которой находится функция syslog) предназначены для использования только обычными приложениями. Поскольку ядро тоже нуждается в функциях журналирования, в него включены свои библиотеки, недоступные приложениям. Поэтому ядро использует свой собственный механизм генерации сообщений.

Демон klogd предназначен для организации обработки этих сообщений. В принципе он может производить такую обработку полностью самостоятельно и независимо от syslogd, например, записывая эти сообщения в файл, но в большинстве случаев используется принятая по умолчанию настройка klogd, при которой все сообщения от ядра пересылаются тому же демону syslogd.

## logrotate — ротация логов

Так как логи постоянно растут и потребность в старых записях может быть ниже, чем в недавних, нужно решить две задачи. С одной стороны, оставить в файле лога только свежие записи, с другой стороны, иметь возможность поднять данные старых логов в случае необходимости. Это обеспечивается с помощью архивирования файлов журнала. В целом данный процесс называется ротацией логов и служит для контроля размера дискового пространства, занимаемого журналами. Утилиту logrotate, которая это делает, обычно запускают с помощью планировщика заданий crond.

Со временем файл журнала имеет свойство увеличиваться, особенно при интенсивной работе какого-­либо сервиса. Соответственно, необходимо иметь возможность контролировать размер журналов. Это делается при помощи команды logrotate, которая обычно выполняется демоном cron. Главная цель команды logrotate состоит в том, чтобы периодически создавать резервные копии журналов и создавать новые чистые журналы. Сохраняется несколько поколений журналов и, когда завершается срок жизни журнала последнего поколения, он может быть заархивирован (сжат). Результат может быть отправлен по почте, например, ответственному за ведение архивов.  
  
Для определения порядка ротации и архивирования журналов используется конфигурационный файл /etc/logrotate.conf. Для разных журналов можно задать разную периодичность, например, ежедневно, еженедельно или ежемесячно, кроме того, можно регулировать количество накапливаемых поколений, а также указать, будут ли копии архивов отправляться ответственному за ведение архивов и, если будут, когда. Ниже показан пример файла /etc/logrotate.conf.

|  |
| --- |
| # сначала заданы параметры "по­умолчанию" (глобальные опции)  # обновлять файлы журнала еженедельно  weekly  # хранить архив логов за 4 последние недели rotate 4  # создавать новый (пустой) файл после ротации (обновления) create  # раскомментируйте, если желаете, чтобы сохраненные файлы сжимались  #compress  # включить настройки ротации из указанного каталога  include /etc/logrotate.d  # не хранить wtmp, или btmp ­­ настройки ротации данных журналов следующие:  /var/log/wtmp { missingok monthly  create 0664 root utmp rotate 1  }  /var/log/btmp {   missingok monthly  create 0664 root utmp rotate 1  } |

Глобальные опции размещаются в начале файла logrotate.conf. Они используются по умолчанию, если где­-то в другом месте не задано ничего более определенного. В примере ротация журналов происходит еженедельно, и резервные копии сохраняются в течение четырех недель. Как только производится ротация журнала, на месте старого журнала автоматически создается новый. Файл logrotate.conf может содержать спецификации из других файлов. Так в него включаются все файлы из каталога /etc/logrotate.d.  
  
В этом примере также содержатся специальные правила для /var/log/wtmp и /var/log/btmp (хранящие информацию о удачных и неудачных попытках входа в систему), ротация которых происходит ежемесячно. Если файлы отсутствуют, сообщение об ошибке не выдается. Создается новый файл и сохраняется только одна резервная копия.  
В этом примере по достижении резервной копией последнего поколения она удаляется, поскольку не определено, что следует с ней делать.  
Резервные копии журналов могут также создаваться, когда журналы достигают определенного размера, и могут быть созданы скрипты из наборов команд для выполнения до или после операции резервного копирования.

В результате работы logrotate остается один активный файл журнала (в который пишутся текущие логи) и несколько архивных файлов, сжатых специальным упаковщиком. Обычно это выглядит так.

|  |
| --- |
| access.log, access.log.1.gz, access.log.2.gz, ... access.log.10.gzt] |

Здесь access.log - текущий, access.log.1.gz - самый новый архив,access.log.10.gz - самый старый архив.  
  
Для расчет объема файлов необходимо исходя из объемов диска и объема логов определить количество дней для ротации.  
Сжатие осуществляется с помощью gzip, если нужно быстрее, то применяется pigz. Иногда применяют lzop, в таком случае его придется поставить дополнительно.

# Практическое задание

1. Загрузить систему вручную через grub в single режиме. После перезагрузки перейти в multiuser режим.
2. Разобрать конфигурационный файл какого-нибудь юнита. Деактивировать и активировать обратно сервис.
3. Сравнить конфигурационные файлы в /etc/systemd/system и в /etc/init для одного и того же сервиса. (несмотря на то, что в Ubuntu 16.0 используется Systemd, конфигурационные файлы Upstart все же присутствуют).
4. \*Если вы знакомы с TCP/IP сделать так, чтобы ssh сервис активировался при подключении к TCP-порту 22.

Примечание. *Задание 4 дано для тех, кому упражнений 1 — 3 показалось недостаточно.*

# Дополнительные материалы

1. Управление службами Linux <https://losst.ru/upravlenie-sluzhbami-linux>
2. Управление логированием в systemd <https://habrahabr.ru/company/selectel/blog/264731/>
3. Уорд Б. Внутреннее устройство Linux. — СПб.: Питер, 2016. — 384 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-496-01952-1 Глава 5 Как происходит загрузка ядра Linux
4. Немет, Эви, Снайдер, Гарт, Хейн, Трент, Уэйли, Бэн. Unix и Linux: руководство системного администратора, 4-е изд. : Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2012. — 1312 с.: ил. — Парал. тит. англ. ISBN 978-5-8459-1740-9 (рус.) Глава 3 Запуск и останов системы
5. <https://losst.ru/nastrojka-rsyslog-v-linux>

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы.

1. https://ru.wikipedia.org/wiki/Init
2. <https://losst.ru/upravlenie-sluzhbami-linux>
3. http://lexpr.ru/node/498
4. <https://www.freedesktop.org/software/systemd/man/systemd.service.html>’
5. <https://losst.ru/nastrojka-rsyslog-v-linux>
6. <http://www.oberhumer.com/opensource/lzop/>
7. <http://support.highway.ru/how/logrotate.html>