systemd (Русский)

Ссылки по теме

- **Systemd/Пользователь**
- **Systemd/Таймеры**
- **systemd FAQ**
- **init**
- **Init Rosetta (Русский)**
- **Демоны#Список демонов**
- **Udev (Русский)**
- **Увеличение производительности/Процесс загрузки системы**
- **Разрешить пользователям выключение системы**

Эта страница нуждается в сопроводителе

Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. **Команда переводчиков ArchWiki**

Состояние перевода: На этой странице представлен перевод статьи **systemd**. Дата последней синхронизации: 20 сентября 2015. Вы можете **помочь** синхронизировать перевод, если в английской версии произошли **изменения**.

Цитата с **веб-страницы проекта**:

> *systemd* - менеджер системы и служб для Linux, совместимый со скриптами инициализации SysV и LSB. systemd обеспечивает возможности агрессивной параллелизации, использует сокеты и активацию **D-Bus** для запускаемых служб, предлагает запуск демонов по необходимости, отслеживает процессы при помощи **контрольных групп** Linux, поддерживает мгновенные снимки и восстановление состояния системы, монтирование и точки монтирования, а также внедряет основанную на зависимостях логику контроля процессов сложных транзакций.

Примечание: За детальным объяснением причин происходящего перехода Arch'a на *systemd* обратитесь к **сообщению на англоязычном форуме**

Contents

[hide]

Основы использования systemctl

Главная команда для отслеживания и контроля состояния *systemd* - команда *systemctl*. Некоторые из вариантов ее использования связаны с изучением состояния системы и управлением системой и службами. Обратитесь к странце руководства **`systemctl(1)`** для получения более детальной информации.

Совет:

- Вы можете использовать все приведенные ниже команды *systemctl* с ключом `-H пользователь@хост` для того, чтобы контролировать *systemd* на удаленной машине. В этом случае для соединения с удаленным процессом *systemd* будет использоваться **SSH**

Совет:

- *systemadm* - официальная графическая оболочка для *systemctl*. Она доступна в пакетах **`systemd-ui`** и **`systemd-ui-git`**AUR[**ссылка недействительна**: сохранено в **aur-mirror**]

Анализ состояния системы

Список запущенных юнитов:

```
$ systemctl
```

или:

```
$ systemctl list-units
```

Список неудач - список юнитов, попытка запуска которых не удалась:

```
$ systemctl --failed
```

Доступные файлы юнитов можно посмотреть в директориях `/usr/lib/systemd/system/` и `/etc/systemd/system/` (второй каталог имеет приоритет). Вы можете увидеть список установленных файлов юнитов командой:

```
$ systemctl list-unit-files
```

Использование юнитов

Юнитами могут быть, например, службы (*.service*), точки монтирования (*.mount*), устройства (*.device*) или сокеты (*.socket*).

При использовании *systemctl* обычно всегда необходимо указывать полное имя файла юнита, включая суффикс, например, `sshd.socket`. Однако, есть несколько сокращений для указания юнита в следующих командах *systemctl*:

- Ели вы не указали суффикс, systemctl предполагает, что это *.service*. Например, `netctl` и `netctl.service` будут трактоваться одинаково
- Точки монтирования будут автоматически преобразованы в соответствующий юнит *.mount*. Например, указание `/home` равнозначно `home.mount`
- Так же, как и точки монтирования, имена устройств автоматически преобразуются в соответствующий юнит *.device*, поэтому указание `/dev/sda2` полностью соответствует юниту `dev-sda2.device`

Для получения дополнительной информации смотрите страницу справочного руководства **`systemd.unit(5)`**.

Примечание: В некоторых именах юнитов содержится знак `@` (например, `имя@`*`строка`*`.service`). Это означает, что они являются **экземплярами** юнита-*шаблона*, в имени которого нет части *`строка`* (например, `имя@.service`). Часть *`строка`* называется *идентификатором экземпляра* и является аргументом, передаваемым юниту-шаблону при вызове команды *systemctl*: в файле юнита он заменит указание (specifier) `%i`.

Для большей точности работы *systemd* будет сперва искать юнит по полному имени файла `имя@строка.суффикс`, и *лишь затем* пытаться использовать экземпляр юнита-шаблона `имя@.суффикс`, даже несмотря на то, что подобные "конфликты" довольно редки, так как большинство файлов юнитов, содержащих знак `@`, подразумевают использование шаблонов. Также помните, что если вызвать юнит-шаблон без идентификатора экземпляра, ничего не получится, поскольку в этом случае не будет возможности передать указание `%i`

Совет:

- Большинство указанных ниже команд также работают, если указать несколько юнитов. Для получения дополнительной информации смотрите страницу справочного руководства **`systemctl(1)`**
- Начиная с версии **systemd 220**, переключатель `--now` может быть использован в сочетании с `enable`, `disable` и `mask` чтобы соответственно запустить или остановить все юниты сразу.
- Пакет может предложить юнитов для различных целей. Если вы только что установили пакет, воспользуйтесь командой `pacman -Qql` *`package`* `| grep -Fe .service -e .socket` для проверки и нахождения юнитов.

Незамедлительно **запустить** юнит:

```
# systemctl start юнит
```

Незамедлительно **остановить** юнит:

```
# systemctl stop юнит
```

Перезапустить юнит:

```
# systemctl restart юнит
```

Попросить юнита **перезагрузить** его настройки:

```
# systemctl reload юнит
```

Показать **статус** юнита, а также запущен он или нет:

```
$ systemctl status юнит
```

Проверить, включен ли юнит в автозапуск при загрузке системы:

```
$ systemctl is-enabled юнит
```

Включить юнит в автозапуск при загрузке системы:

```
# systemctl enable юнит
```

Убрать юнит из автозапуска при загрузке системы:

```
# systemctl disable юнит
```

Маскировать юнит, чтобы сделать невозможным его запуск:

```
# systemctl mask юнит
```

Снять маску юнита:

```
# systemctl unmask юнит
```

Показать страницу справочного руководства, связанного с юнитом (необходима поддержка этой функции в указанном файле юнита):

```
$ systemctl help юнит
```

Перезагрузить *systemd* для поиска **новых или измененных юнитов**:

```
# systemctl daemon-reload
```

Управление питанием

Для управления питанием от имени непривилегированного пользователя необходим **polkit**. Если вы находитесь в локальной пользовательской сессии *systemd-logind*, и нет других активных сессий, приведенные ниже команды сработают и без привилегий суперпользователя. В противном случае (например, вследствие того, что другой пользователь вошел в систему в tty), *systemd* автоматически запросит у вас пароль суперпользователя.

Завершить работу и перезагрузить систему:

```
$ systemctl reboot
```

Завершить работу и выключить компьютер (с отключением питания):

```
$ systemctl poweroff
```

Перевести систему в ждущий режим:

```
$ systemctl suspend
```

Перевести систему в спящий режим:

```
$ systemctl hibernate
```

Перевести систему в режим гибридного сна (или suspend-to-both):

```
$ systemctl hybrid-sleep
```

Написание файлов юнитов

Синтаксис **файлов юнитов** *systemd* вдохновлен файлами *.desktop* XDG Desktop Entry Specification, а они, в свою очередь - файлами *.ini* Microsoft Windows. Файлы юнитов загружаются из двух мест. Вот они по приоритету от низшего к высшему:

- `/usr/lib/systemd/system/`: юниты, предоставляемые пакетами при их установке
- `/etc/systemd/system/`: юниты, устанавливаемые системным администратором

Примечание: При запуске *systemd* в **пользовательском режиме** используются совершенно другие пути загрузки

В качестве примера, посмотрите установленные юниты вашими пакетами, а также **секцию примеров** из **`systemd.service(5)`**.

Совет: Как и обычно, вы можете добавлять комментарии, предваряемые символом `#`, но только на новых строках. Не используйте комментарии в конце строки, после параметров *systemd*, иначе юнит не будет запущен

Обработка зависимостей

В случае использования *systemd* зависимости могут быть указаны правильным построением файлов юнитов. Наиболее частый случай -- юниту *A* требуется, чтобы юнит *B* был запущен перед тем, как запустится сам юнит *A*. В этом случае добавьте строки `Requires=B` и `After=B` в секцию `[Unit]` файла службы *A*. Если подобная зависимость не является обязательной, взамен указанных выше добавьте, соответственно, строки `Wants=B` и `After=B`. Обратите внимание,

что `Wants=` и `Requires=` не подразумевают `After=`, что означает, что если `After=` не определено, два юнита будут запущены параллельно друг другу.

Обычно зависимости указываются в файлах служб, а не в целевых юнитах. Например, `network.target` потребуется любой службе, которая связана с настройкой ваших сетевых интерфейсов, поэтому в любом случае определите загрузку вашего пользовательского юнита после запуска `network.target`.

Типы служб

Существует несколько различных типов запуска служб, которые надо иметь в виду при написании пользовательского файла службы. Тип определяется параметром `Type=` в секции `[Service]`:

- `Type=simple` (по умолчанию): *systemd* предполагает, что служба будет запущена незамедлительно. Процесс при этом не должен разветвляться. Не используйте этот тип, если другие службы зависят от очередности при запуске данной службы. Исключение - активация сокета
- `Type=forking`: *systemd* предполагает, что служба запускается однократно и процесс разветвляется с завершением родительского процесса. Используйте данный тип для запуска классических демонов за исключением тех случаев, когда, как вам известно, в таком поведении процесса нет необходимости. Вам следует также определить `PIDFile=`, чтобы *systemd* могла отслеживать основной процесс
- `Type=oneshot`: полезен для скриптов, которые выполняют одно задание и завершаются. Вам может понадобиться также установить параметр `RemainAfterExit=yes`, чтобы *systemd* по-прежнему считала процесс активным, даже после его завершения
- `Type=notify`: идентичен параметру `Type=simple`, но с той оговоркой, что демон пошлет *systemd* сигнал о своей готовности. Эталонная реализация данного уведомления представлена в *libsystemd-daemon.so*
- `Type=dbus`: сервис считается находящимся в состоянии готовности, когда определенное `BusName` появляется в системной шине DBus
- `Type=idle`: *systemd* will delay execution of the service binary until all jobs are dispatched. Кроме того, поведение очень похоже на `Type=simple`.

Смотрите справочную страницу руководства **systemd.service(5)** для более детального пояснения значений `Type`.

Обратитесь к руководству **`systemd.service(5)`** для получения более детального объяснения.

Редактирование предоставленных пакетами файлов юнитов

Есть два способа редактирования файлов юнита, предоставленного пакетом: заменить весь блок файла на новый или создать фрагмент кода, который применяется в верхней части существующего блока файла. В обоих методах, чтобы применить изменения, нужно перезагрузить юнит. Это может быть сделано либо путем редактирования блока с помощью `systemctl edit` (которая автоматически загружает модуль) либо при перезагрузке всех юнитов:

```
# systemctl daemon-reload
```

Совет:

- Вы можете использовать *systemd-delta*, чтобы увидеть, какие файлы юнитов были переопределены и что конкретно было изменено. Для обслуживания системы в целом важно регулярно проверять предоставляемые файлы юнитов на полученные обновления.

- Используйте `systemctl cat *юнит*` чтобы посмотреть содержимое файла юнита и все Drop-in snippets кода.
- Подсветку синтаксиса для файлов юнитов *systemd* в редакторе **Vim** можно включить, установив пакет **`vim-systemd`** из **официальных репозиториев**.

Замена файлов юнита

Чтобы заменить файл юнита `/usr/lib/systemd/system/*юнит*`, создайте файл `/etc/systemd/system/*юнит*` и перезапустите юнит для обновления символьных ссылок:

```
# systemctl reenable юнит
```

В качестве альтернативы, можно выполнить:

```
# systemctl edit --full юнит
```

Эта команда откроет `/etc/systemd/system/*юнит*` в вашем текстовом редакторе (копирует установленную версию, если она еще не существует) и автоматически загружает её, когда вы закончите редактирование.

Примечание: Pacman не обновит заменённые файлы юнита, в отличие от оригинальных которые обновятся. Так что этот метод может сделать обслуживание системы более сложным. По этой причине рекомендуется следующий подход.

Drop-in snippets

Чтобы создать drop-in snippets для файла юнита `/usr/lib/systemd/system/*юнит*`, создайте каталог `/etc/systemd/system/*юнит*.d/` и поместите файлы *.conf* там, чтобы отменять или добавлять новые опции. *systemd* будет анализировать эти файлы *.conf* и применять их поверх оригинального юнита.

Самый простой способ чтобы выполнить это, сделайте:

```
# systemctl edit юнит
```

Эта команда откроет `/etc/systemd/system/*юнит*.d/override.conf` (создаст его если это потребуется) в вашем текстовом редакторе и автоматически перезапустит юнит, когда вы закончите редактирование.

Примеры

Например, если вы просто хотите добавить дополнительную зависимость к юниту, можно создать следующий файл:

```
/etc/systemd/system/unit.d/customdependency.conf

[Unit]
Requires=new dependency
After=new dependency
```

В качестве другого примера, для того чтобы заменить направление для юнита `ExecStart`, что не относится к типу `oneshot`, создайте следующий файл:

```
/etc/systemd/system/unit.d/customexec.conf
```

```
[Service]
ExecStart=
ExecStart=новая комманда
```

Обратите внимание `ExecStart` должна быть очищена, перед новым назначением ([1]).

Еще один пример, чтобы автоматически перезапустить службу:

```
/etc/systemd/system/unit.d/restart.conf

[Service]
Restart=always
RestartSec=30
```

Цели

systemd использует *цели* (англ. target), которые выполняют ту же задачу, что и уровни запуска (англ. runlevel), но действуют немного по-другому. Каждая *цель* поименована (т.е. имеет собственное имя, а не номер) и, как предполагается, предназначена для конкретных задач; возможно иметь в одно и то же время активными несколько таких целей. Некоторые *цели* реализованы так, что наследуют все службы других *целей*, добавляя к ним свои. В *systemd* имеются также *цели*, которые имитируют общие уровни запуска SystemVinit, поэтому вы можете переключаться между целевыми юнитами, используя привычную команду `telinit RUNLEVEL`.

Получение информации о текущих целях

При использовании *systemd* для этого предназначена следующая команда (заменяющая `runlevel`):

```
$ systemctl list-units --type=target
```

Создание пользовательской цели

Уровни запуска, по которым расписаны конкретные задачи при установке ванильной Fedora по умолчанию - 0, 1, 3, 5 и 6 - имеют соответствие 1:1 с конкретными *целями systemd*. К сожалению, не существует хорошего способа сделать то же самое для определяемых пользователем уровней, таких как 2 и 4. Их использование предполагает, что вы создаете новый именованный *целевой юнит systemd* наподобие `/etc/systemd/system/ваша цель`, который берет за основу один из существующих уровней запуска (взгляните, например, на `/usr/lib/systemd/system/graphical.target`), создаете каталог `/etc/systemd/system/ваша цель.wants`, а после этого - символические ссылки на дополнительные службы из директории `/usr/lib/systemd/system/`, которые вы хотите включить при загрузке.

Таблица целей

Уровнень запуска SysV	Цель systemd	Примечания
0	runlevel0.target,	Выключить систему

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

Изменение текущей цели

В *systemd* цели доступны посредством *целевых юнитов*. Вы можете изменить их командой:

```
# systemctl isolate graphical.target
```

Данная команда изменит только лишь текущую цель и не повлияет на следующую загрузку системы. Она соответствует командам Sysvinit вида `telinit 3` и `telinit 5`.

Изменение цели загрузки по умолчанию

Стандартная цель - `default.target`, которая по умолчанию является псевдонимом `graphical.target` (примерно соответствующего прежнему уровню запуска 5). Для изменения цели загрузки по умолчанию добавьте один из следующих **параметров ядра** в ваш загрузчик:

- `systemd.unit=multi-user.target` (что примерно соответствует прежнему уровню запуска 3)
- `systemd.unit=rescue.target` (что примерно соответствует прежнему уровню запуска 1)

Другой способ - оставить загрузчик без изменений, а изменить целевой юнит по умолчанию - *default.target*. Это делается с использованием *systemctl*:

```
# systemctl set-default multi-user.target
```

Чтобы иметь возможность перезаписать ранее установленную `default.target`, используйте опцию force:

```
# systemctl set-default -f multi-user.target
```

Эффект от применения данной команды выводится через *systemctl*. Символическая ссылка на новый целевой юнит по умолчанию создается в директории `/etc/systemd/system/default.target`.

Временные файлы

"*systemd-tmpfiles* создает, удаляет и очищает непостоянные и временные файлы и каталоги". Он читает конфигурационные файлы из `/etc/tmpfiles.d/` и `/usr/lib/tmpfiles.d/`, чтобы понять, что ему следует делать. Конфигурационные файлы в первом каталоге имеют приоритет над теми, что расположены во втором.

Конфигурационные файлы обычно предоставляются вместе с файлами служб и имеют названия вида `/usr/lib/tmpfiles.d/`*`программа`*`.conf`. Например, демон **Samba** предполагает, что существует каталог `/run/samba` с корректными правами доступа. Поэтому пакет **samba** поставляется в следующей конфигурации:

```
/usr/lib/tmpfiles.d/samba.conf
```

```
D /run/samba 0755 root root
```

Конфигурационные файлы также могут использоваться для записи значений при старте системы. Например, если вы используете `/etc/rc.local` для отключения пробуждения от устройств USB при помощи `echo USBE > /proc/acpi/wakeup`, вместо этого вы можете использовать следующий tmpfile:

```
/etc/tmpfiles.d/disable-usb-wake.conf
```

```
w /proc/acpi/wakeup - - - - USBE
```

Для получения дополнительной информации смотрите страницы справочного руководства (man) `systemd-tmpfiles(8)` и `tmpfiles.d(5)`.

Примечание: Этот способ может не сработать для установки опций в `/sys`, поскольку служба *systemd-tmpfiles-setup* может запускаться перед тем, как будут загружены соответствующие модули устройств. В этом случае при помощи команды `modinfo` *`модуль`* вы можете проверить, имеет ли модуль параметр для установки необходимой вам опции, и установить эту опцию в **конфигурационном файле /etc/modprobe.d**. В противном случае для установки верных атрибутов сразу, как только устройство появляется, вам придется написать **правило udev**

Таймеры

Таймер - это файл конфигурации юнита, имя которого заканчивается на *.timer*. Он расшифровывает информацию о таймере, контролируемом при помощи *systemd*, для активации в определенное время. Смотрите статью **systemd/Таймеры**.

Примечание: Таймеры способны в значительной степени заменить функциональность *cron*. Смотрите раздел **Замена cron**

Монтирование

Так как systemd полностью заменяет собой SysVinit, он отвечает за точки монтирования, описанные в файле `/etc/fstab`. Фактически он выходит за рамки возможностей обычного `fstab`, реализуя особые точки монтирования с префиксом `x-systemd`, например, т.н. *автомонтирование* (монтирование по запросу) использует данные расширения (см. **для более подробной информации как это реализовано**). С полным описанием всех расширений и работы с ними вы можете ознакомиться на английском в **[2]**

Журнал

systemd имеет собственную систему ведения логов, названную журналом (journal). В связи с этим больше не требуется запускать демон `syslog`. Для чтения логов используйте команду:

```
# journalctl
```

В Arch Linux каталог `/var/log/journal/` является частью пакета **`systemd`**, и по умолчанию (когда в конфигурационном файле `/etc/systemd/journald.conf` параметр `Storage=` имеет значение `auto`) журнал записывается именно в `/var/log/journal/`. Если вы или какая-то программа удалит этот каталог, *systemd* **не** пересоздаст его автоматически и вместо этого будет писать свои журналы по непостоянному пути `/run/systemd/journal`. Однако, папка будет пересоздана, когда вы установите `Storage=persistent` и выполните `systemctl restart systemd-journald` (или перезагрузитесь).

Сообщения в журнале классифицируются по приоритету и объектам. Классификация записей соответствует классическому протоколу **Syslog** (**RFC 5424**).

Фильтрация вывода

journalctl позволяет фильтровать вывод по особым полям. Помните, что, если должно быть отражено большое количество сообщений или необходима фильтрация в большом промежутке времени, вывод этой команды может быть отложен на какое-то время.

Совет: Поскольку журнал хранится в двоичном формате, содержимое его сообщений не меняется. Это означает, что их можно просматривать при помощи *strings*, например, в окружении, в котором не установлен *systemd*. Пример:

```
$ strings /mnt/arch/var/log/journal/af4967d77fba44c6b093d0e9862f6ddd/syste
m.journal | grep -i сообщение
```

Примеры:

- Показать все сообщения с момента текущей загрузки системы:

```
# journalctl -b
```

  Однако, пользователи часто интересуются сообщениями не для текущей, а для предыдущей загрузки (например, если произошел невосстановимый сбой системы). Это возможно, если задать параметр флагу `-b`: `journalctl -b -0` покажет сообщения с момента текущей загрузки, `journalctl -b -1` - предыдущей загрузки, `journalctl -b -2` - следующей за предыдущей, и т.д. Для просмотра полного описания смотрите страницу справочного руководства **`journalctl(1)`**: имеется гораздо более мощная семантика

- Показать все сообщения, начиная с какой-либо даты (и, если хотите, времени):

```
# journalctl --since="2012-10-30 18:17:16"
```

- Показать все сообщения за последние 20 минут:

```
# journalctl --since "20 min ago"
```

- Показывать новые сообщения:

```
# journalctl -f
```

- Показать все сообщения для конкретного исполняемого файла:

```
# journalctl /usr/lib/systemd/systemd
```

- Показать все сообщения для конкретного процесса:

```
# journalctl _PID=1
```

- Показать все сообщения для конкретного юнита:

```
# journalctl -u netcfg
```

- Показать кольцевой буфер ядра:

```
# journalctl -k
```

- Показать auth.log эквивалентно фильтрации syslog facility:

```
# journalctl -f -l SYSLOG_FACILITY=10
```

Для получения дополнительной информации смотрите страницы справочного руководства **`journalctl(1)`** и **`systemd.journal-fields(7)`** или **пост в блоге** Lennart'a.

Совет: По умолчанию *journalctl* отсекает части строк, которые не вписываются в экран по ширине, и, в некоторых случаях, возможно, будет лучше использовать специальную программу-обертку. Управление этой возможностью производится посредством **переменной окружения** `SYSTEMD_LESS`, в которой содержатся опции, передаваемые в **less** (программа постраничного просмотра, используемая по умолчанию). По умолчанию ей присвоены опции `FRSXMK` (для получения дополнительной информации смотрите **`less(1)`** и **`journalctl(1)`**).

Если убрать опцию `S`, будет достигнут требуемый результат. Например, запустите *journalctl*, как показано здесь:

```
$ SYSTEMD_LESS=FRXMK journalctl
```

Если вы хотите, чтобы такое поведение использовалось по умолчанию, **экспортируйте** переменную из файла `~/.bashrc` или `~/.zshrc`

Ограничение размера журнала

Если журнал сохраняется при перезагрузке, его размер по умолчанию ограничен значением в 10% от объема соответствующей файловой системы. Например, для директории `/var/log/journal`, расположенной на корневом разделе в 50 Гбайт, максимальный размер журналируемых данных составит 5 Гбайт. Максимальный объем постоянного журнала можно контролировать при помощи значения `SystemMaxUse` в конфигурационном файле `/etc/systemd/journald.conf`, поэтому для ограничения его объемом, например, в 50 Мбайт раскомментируйте и отредактируйте соответствующую строку:

```
/etc/systemd/journald.conf

SystemMaxUse=50M
```

Для получения дополнительной информации обратитесь к странице справочного руководства **journald.conf(5)**.

Очистка файлов журнала вручную

Файлы журнала находятся в `/var/log/journal`, так что `rm` будет работать. Или используйте `journalctl`,

Примеры:

- Remove archived journal files until the disk space they use falls below 100M:

```
# journalctl --vacuum-size=100M
```

- Make all journal files contain no data older than 2 weeks.

```
# journalctl --vacuum-time=2weeks
```

Для получения дополнительной информации, обратитесь к **journalctl(1)**.

Journald в связке с классическим демоном syslog

Совместимость с классической реализацией non-journald aware **syslog** можно обеспечить, заставив *systemd* направлять все сообщения через сокет `/run/systemd/journal/syslog`. Чтобы дать возможность демону syslog работать вместе с журналом *systemd*, следует привязать данный демон к указанному сокету вместо `/dev/log` (**официальное сообщение**). Пакетом **syslog-ng** из репозиториев автоматически предоставляется необходимая конфигурация.

Начиная с версии systemd 216, по умолчанию `journald.conf` для передачи данных в сокет был изменён на `ForwardToSyslog=no`, чтобы избежать нагрузки на систему, потому что **rsyslog** или **syslog-ng** (начиная с версии 3.6) тянут сообщения из журнала **самостоятельно**.

Смотрите **Syslog-ng#Overview** и **Syslog-ng#syslog-ng and systemd journal**, или соответственно **rsyslog** для подробной информации о конфигурировании.

Если взамен вы используете **rsyslog**[AUR], нет необходимости менять эту настройку, поскольку **rsyslog** забирает сообщения из журнала **самостоятельно**.

Перенаправить журнал на /dev/tty12

Создайте **drop-in каталог** `/etc/systemd/journald.conf.d` и создайте файл `fw-tty12.conf` с содержимым:

```
/etc/systemd/journald.conf.d/fw-tty12.conf

[Journal]
ForwardToConsole=yes
TTYPath=/dev/tty12
MaxLevelConsole=info
```

Затем **перезапустите** systemd-journald.

Команда просмотра другого журнала

Если появилась необходимость проверить логи другой системы, которая неисправна, загрузитесь с работоспособной системы, чтобы восстановить неисправную систему. Примонтируйте диск неисправной системы, например в `/mnt` и укажите путь журнала через `-D`/`--directory`, например так:

```
$ journalctl -D /mnt/var/log/journal -xe
```

Решение проблем

Изучение ошибок systemd

В качестве примера мы изучим ошибки службы `systemd-modules-load`:

1. Давайте найдем службы *systemd*, которые не смогли запуститься:

```
$ systemctl --failed

systemd-modules-load.service   loaded failed failed  Load Kernel Modules
```

2. Хорошо, мы обнаружили проблему в службе `systemd-modules-load` и хотим узнать больше:

```
$ systemctl status systemd-modules-load

systemd-modules-load.service - Load Kernel Modules
   Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; s
tatic)
   Active: failed (Result: exit-code) since So 2013-08-25 11:48:13 CEST; 3
2s ago
     Docs: man:systemd-modules-load.service(8).
           man:modules-load.d(5)
```

```
  Process: 15630 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exi
ted, status=1/FAILURE)
```

Если вы не увидите в списке `Process ID`, просто перезапустите службу при помощи команды `systemctl restart systemd-modules-load`

3. Теперь у нас есть id процесса (PID) для более детального изучения ошибки. Введите следующую команду с правильным `Process ID` (в данном примере это 15630):

```
$ journalctl _PID=15630

-- Logs begin at Sa 2013-05-25 10:31:12 CEST, end at So 2013-08-25 11:51:1
7 CEST. --
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'b
lacklist usblp'
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'i
nstall usblp /bin/false'
```

4. Мы видим, что некоторые конфигурационные файлы модулей ядра имеют неверные настройки. В этом случае мы взглянем на эти настройки в каталоге `/etc/modules-load.d/`:

```
$ ls -Al /etc/modules-load.d/

...
-rw-r--r--   1 root root    79  1. Dez 2012  blacklist.conf
-rw-r--r--   1 root root     1  2. Mär 14:30 encrypt.conf
-rw-r--r--   1 root root     3  5. Dez 2012  printing.conf
-rw-r--r--   1 root root     6 14. Jul 11:01 realtek.conf
-rw-r--r--   1 root root    65  2. Jun 23:01 virtualbox.conf
...
```

5. Сообщение об ошибке `Failed to find module 'blacklist usblp'` должно относиться к неправильной настройке в файле `blacklist.conf`. Давайте закомментируем настройку, вставив хэш-символ # перед каждой опцией, найденной на шаге 3:

```
/etc/modules-load.d/blacklist.conf

# blacklist usblp
# install usblp /bin/false
```

6. Теперь попробуйте запустить `systemd-modules-load`:

```
$ systemctl start systemd-modules-load
```

Если все прошло успешно, ничего не отобразится. Если же вы видите какие-либо ошибки, вернитесь к шагу 3 и используйте новый PID для устранения оставшихся ошибок.

Если все хорошо, вы можете удостовериться, что служба успешно запустилась, при помощи команды:

```
$ systemctl status systemd-modules-load

systemd-modules-load.service - Load Kernel Modules
   Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; s
tatic)
   Active: active (exited) since So 2013-08-25 12:22:31 CEST; 34s ago
     Docs: man:systemd-modules-load.service(8)
           man:modules-load.d(5)
 Process: 19005 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exit
ed, status=0/SUCCESS)
Aug 25 12:22:31 mypc systemd[1]: Started Load Kernel Modules.
```

Чаще всего подобные проблемы можно решить так, как показано выше. Для дальнейшего изучения этого вопроса взгляните на раздел **#Диагностика проблем с загрузкой системы**.

Диагностика проблем с загрузкой системы

Загрузитесь с этими параметрами ядра:

```
systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M
```

Дополнительная информация по отладке.

Диагностика проблем в работе определенной службы

The factual accuracy of this article or section is disputed.

Reason: This may not catch all errors such as missing libraries. (Discuss in **User talk:Alucryd#Plex**)

Если какая-либо служба *systemd* ведет себя не так, как ожидается, и вы хотите получить дополнительную информацию о том, что происходит, присвойте **переменной окружения** `SYSTEMD_LOG_LEVEL` значение `debug`. Например, чтобы запустить демон *systemd-networkd* в режиме отладки:

```
# systemctl stop systemd-networkd
# SYSTEMD_LOG_LEVEL=debug /lib/systemd/systemd-networkd
```

В качестве альтернативы можно временно отредактировать файл службы для получения подробного вывода. Например:

```
/usr/lib/systemd/system/systemd-networkd.service

[Service]
...
Environment=SYSTEMD_LOG_LEVEL=debug
....
```

Если вы знаете, что в дальнейшем вам по-прежнему будет нужна эта отладочная информация, добавьте переменную **обычным** способом.

Выключение/перезагрузка происходят ужасно долго

Если процесс выключения занимает очень долгое время (или, по-видимому, зависает), то, вероятно, виновата служба, которая не завершает свою работу. *systemd* ожидает некоторое время, пока каждая служба завершит свою работу самостоятельно, и только потом пытается принудительно завершить (kill) ее. Если вы столкнулись с такой проблемой, обратитесь к **данной статье (англ.)**.

По-видимому, процессы с кратким сроком жизни не оставляют записей в логах

Если команда `journalctl -u foounit` не показывает вывода для службы с коротким сроком жизни, вместо нее обратитесь к PID. Например, если загрузка службы `systemd-modules-load.service` завершилась неудачно и команда `systemctl status systemd-modules-load` показывает, что она была запущена с PID 123, то вы сможете посмотреть вывод процесса в журнале под данным PID, то есть командой `journalctl -b _PID=123`. Такие поля метаданных для журнала, как `_SYSTEMD_UNIT` и `_COMM`, собираются асинхронно и зависят от директории `/proc` в случае с действующими процессами. Исправление этой ситуации требует внесения исправлений в ядро для обеспечения предоставления этих данных через сокет, наподобие `SCM_CREDENTIALS`.

Отключение журналирования аварийных дампов памяти приложений

Добавьте в файл `/etc/systemd/coredump.conf` такую строку:

```
Storage=none
```

и выполните:

```
# systemctl daemon-reload
```

чтобы перезагрузить конфигурацию.

Сообщение об ошибке при перезагрузке или выключении

cgroup : option or name mismatch, new: 0x0 "", old: 0x4 "systemd"

Для получения объяснения смотрите **эту ветку**.

watchdog watchdog0: watchdog did not stop!

Для получения объяснения смотрите **эту ветку**.

Время загрузки системы увеличивается с течением времени

После использования `systemd-analyze` некоторое количество пользователей заметило, что их время загрузки значительно увеличилось по сравнению с тем, к чему они привыкли. После использования `systemd-analyze blame` **NetworkManager** тратил необычно большое количество времени на запуск.

Проблема некоторых пользователей была связана с тем, что `/var/log/journal` становился слишком большим. При этом также может уменьшаться скорость работы других команд, например, `systemctl status` или `journalctl`. Для решения проблемы можно удалить все файлы из каталога журнала (в идеале - сделав где-нибудь резервные копии, хотя бы временно) и затем установить предел размера файла журнала, как описано в разделе **#Ограничение размера журнала**.

systemd-tmpfiles-setup.service fails to start at boot

Начиная с версии Systemd 219, `/usr/lib/tmpfiles.d/systemd.conf` определяет атрибуты для каталогов ACL, в `/var/log/journal` и, следовательно, требует чтобы поддержка ACL была включена для файловой системы, где находится журнал.

Смотрите инструкцию **Access Control Lists (Русский)#Включение ACL** для включения ACL на файловой системе в которой `/var/log/journal`.

Смотрите также

- **Systemd для администраторов (Рус.)**
- **systemd для администраторов (PDF)** - перевод **цикла статей** Леннарта Поттеринга (Lennart Poettering)
- **Официальный веб-сайт (англ.)**
- **Статья в Википедии**
- **Страницы справочных руководств (англ.)**
- **Оптимизации systemd (англ.)**
- **FAQ (англ.)**
- **Советы и трюки (англ.)**
- **О systemd в Fedora Project (англ.)**
- **Отладка проблем systemd (англ.)**
- **часть 1** и **часть 2** вводной статьи в журнале *The H Open* (англ.)
- **Блог Lennart'a (англ.)**
- **Status update (англ.)**
- **Status update2 (англ.)**
- **Status update3 (англ.)**
- **Самые последние изменения (англ.)**
- **Шпаргалка Fedora по переходу с SysVinit на systemd**
- **Статья systemd в Gentoo Wiki (англ.)**
- **Emacs Syntax highlighting for Systemd files**

Categories:

- Daemons and system services (Русский)
- Boot process (Русский)
- Русский