systemd (Русский)

9 languages

- Page
- Discussion
- Read
- View source
- View history

Tools

Ссылки по теме

- **Systemd/Пользователь**
- **Systemd/Таймеры**
- **Systemd/Журнал**

- systemd/FAQ
- init
- Демоны
- udev (Русский)
- Improving performance/Boot process
- Разрешить пользователям выключение системы

Состояние перевода: На этой странице представлен перевод статьи **systemd**. Дата последней синхронизации: 12 июля 2021. Вы можете **помочь** синхронизировать перевод, если в английской версии произошли **изменения**.

Цитата с **веб-страницы проекта**:

> *systemd* — набор базовых компонентов Linux-системы. Представляет собой менеджер системы и служб, который выполняется как процесс с PID 1 и запускает остальную часть системы. systemd обеспечивает возможности агрессивной параллелизации, сокетную и **D-Bus** активацию для запуска служб, запуск демонов по запросу, отслеживание процессов с помощью **контрольных групп** Linux, обслуживание точек (авто)монтирования, а также предлагает развитую транзакционную логику управления службами на основе зависимостей. *systemd* поддерживает сценарии инициализации SysV и LSB и работает как замена sysvinit. Среди прочих элементов и функций — демон журнала, утилиты управления базовой конфигурацией системы (имя хоста, дата, языковой стандарт), ведение списков вошедших в систему пользователей, запущенных контейнеров, виртуальных машин, системных учётных записей, каталогов и настроек времени выполнения, а также демоны для управления несложными сетевыми конфигурациями, синхронизации времени по сети, пересылки журналов и разрешения имён.

Примечание: Причины перехода Arch на systemd подробно объяснены **на форуме**.

Основы использования systemctl

Главная команда для работы с systemd — *systemctl*. Она позволяет (среди прочего) отлеживать состояние системы и управлять системой и службами. Подробнее см. **`systemctl(1)`**.

Совет:

- Для управления systemd на удалённой машине команды необходимо выполнять с ключом `-H пользователь@хост`. Соединение с удалённым процессом systemd будет установлено через **SSH**.
- В **Plasma** для systemctl разработан графический интерфейс **`systemd-kcm`**AUR[ссылка недействительна: package not found]. После установки соответствующий модуль появится в разделе *System administration*.

Использование юнитов

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

При работе с systemctl обычно необходимо указывать полное имя юнита с суффиксом, например, `sshd.socket`. Существует несколько возможных сокращений:

- Если суффикс не указан, systemctl предполагает, что это *.service*. Например, `netctl` равнозначно `netctl.service`.
- Точки монтирования автоматически преобразуются в юнит *.mount*. Например, `/home` равнозначно `home.mount`.
- Аналогично точкам монтрования, имена устройств автоматически преобразуются в юнит *.device*. Например, `/dev/sda2` равнозначно `dev-sda2.device`.

Подробнее см. **`systemd.unit(5)`**.

Примечание: Некоторые юниты содержат в названии символ `@` (вида `название@`*`строка`*`.service`). Это т.н. **экземпляры** юнита-шаблона, настоящее имя которого не содержит части *`строка`* (т.е. имеет вид `название@.service`). *`строка`* называется идентификатором экземпляра и передаётся юниту-шаблону в качестве аргумента при вызове systemctl: в файле юнита идентификатор заменит спецификатор `%i`.

Если говорить точнее, systemd сначала попытается найти юнит, название которого полностью совпадёт с `название@`*`строка`*`.суффикс`, и лишь в случае неудачи создаст экземпляр шаблона `название@.суффикс`. Тем не менее, такие "конфликты" довольно редки, так как по соглашению символ `@` должен использоваться только в названиях юнитов-шаблонов. Также помните, что вызвать юнит-шаблон без идентификатора экземпляра не получится, поскольку в этом случае нечего будет подставить вместо спецификатора `%i`.

Совет:

- Большинство команд ниже также будут работать, если указать несколько юнитов; подробнее см. **`systemctl(1)`**.
- Опция `--now` в командах `enable`, `disable` и `mask` соответственно запускает, останавливает или маскирует указанный юнит сразу при выполнении команды, а не после перезагрузки.
- Пакеты могут содержать собственные юниты для различных целей. Если вы только что установили пакет, выполните `pacman -Qql` *`название_пакета`* `| grep -Fe .service -e .socket`, чтобы их найти.

Действие	Команда	Примечание

Анализ состояния системы		
Состояние системы	`$ systemctl status`	
Список запущенных юнитов	`$ systemctl` или `$ systemctl list-units`	
Список юнитов, запустить которые не удалось	`$ systemctl --failed`	
Список установленных файлов юнитов[1]	`$ systemctl list-unit-files`	
Информация о процессе по его PID	`$ systemctl status` *`pid`*	**cgroup slice**, занимаемая память и родительский процесс
Состояние юнита		
Страница руководства юнита	`$ systemctl help` *`юнит`*	если юнит её предоставляет
Состояние юнита	`$ systemctl status` *`юнит`*	в т. ч. работает ли он в данный момент
Проверить, добавлен ли юнит в автозапуск	`$ systemctl is-enabled` *`юнит`*	

Запуск, перезапуск, перезагрузка юнита		
Незамедлительно **запустить** юнит	`# systemctl start` *`юнит`*	
Незамедлительно **остановить** юнит	`# systemctl stop` *`юнит`*	
Перезапустить юнит	`# systemctl restart` *`юнит`*	
Перезагрузить юнит с новыми настройками	`# systemctl reload` *`юнит`*	
Перезагрузить настройки systemd[2]	`# systemctl daemon-reload`	сканировать систему на наличие новых или изменённых юнитов
Включение юнита (автозапуск)		
Включить юнит, добавив его в автозапуск	`# systemctl enable` *`юнит`*	
Включить юнит и сразу **запустить**	`# systemctl enable --now` *`юнит`*	
Отключить запуск юнита при загрузке	`# systemctl disable` *`юнит`*	

Включить юнит **заново**[3]	`# systemctl reenable` *`юнит`*	т.е. отключить и снова включить
Маскировка юнита		
Замаскировать юнит, сделав невозможным его запуск[4]	`# systemctl mask` *`юнит`*	
Снять маскировку юнита	`# systemctl unmask` *`юнит`*	

1. В руководстве **`systemd.unit(5) §UNIT FILE LOAD PATH`** приведён перечень каталогов, в которых могут храниться файлы юнитов.
2. Перезагружаются только настройки systemd, но не юнитов. Для юнитов необходимо использовать команду *reload*.
3. Например, если раздел `[Install]` изменился с момента последнего включения.
4. Как вручную, так и по зависимости, что делает маскировку несколько опасной.

Управление питанием

Для управления питанием от имени непривилегированного пользователя необходим **polkit**. Если вы находитесь в локальном пользовательском сеансе *systemd-logind* и нет других активных сеансов, приведенные ниже команды сработают, даже если будут выполнены не от root. В противном случае (например, другой пользователь вошел в систему через tty) systemd автоматически запросит у вас пароль суперпользователя.

Действие	Команда
Завершить работу и перезагрузить систему	`$ systemctl reboot`

Завершить работу и выключить компьютер	`$ systemctl poweroff`
Перевести систему в ждущий режим	`$ systemctl suspend`
Перевести систему в спящий режим	`$ systemctl hibernate`
Перевести систему в режим гибридного сна (suspend-to-both)	`$ systemctl hybrid-sleep`

Написание файлов юнитов

Синтаксис файлов юнитов systemd (см. **`systemd.unit(5)`**) вдохновлён **desktop-файлами** XDG Desktop Entry Specification, а они, в свою очередь, основаны на синтаксисе **файлов .ini** Microsoft Windows. Файлы юнитов загружаются из целого ряда мест (команда `systemctl show --property=UnitPath` выведет полный список), ключевыми из которых являются следующие (в порядке увеличения приоритета):

- `/usr/lib/systemd/system/`: юниты, добавленные пакетами при установке;
- `/etc/systemd/system/`: юниты, созданные системным администратором.

Примечание:

- При запуске systemd в **пользовательском режиме** пути загрузки будут отличаться.
- Названия юнитов могут содержать только буквы и цифры ASCII-набора, подчёркивания и точки. Другие символы должны быть экранированы в C-стиле ("\x2d") или использоваться исключительно в рамках определённой семантики ('@', '-'). Подробнее см. **`systemd.unit(5)`** и **`systemd-escape(1)`**.

При создании собственных юнитов за образец можно взять юниты установленных пакетов или примеры из **`systemd.service(5)` §`EXAMPLES`**.

Совет: Комментарии в файлах юнитов должны начинаться с символа `#` и размещаться на отдельной строке. Не используйте комментарии в конце строки, после параметров *systemd*, иначе юнит не будет работать.

Обработка зависимостей

В *systemd* зависимости определяются правильным построением файлов юнитов. Простой пример — юниту *A* требуется, чтобы юнит *B* был запущен перед запуском самого юнита *A*. Для этого добавьте строки `Requires=B` и `After=B` в раздел `[Unit]` юнит-файла *A*. Если зависимость является необязательной, укажите `Wants=B` и `After=B` соответственно. Обратите внимание, что `Wants=` и `Requires=` не подразумевают `After=`. Если `After=` не указать, то юниты будут запущены параллельно.

Зависимости обычно указываются для служб, но не для **целей**. Так, цель `network.target` будет "подтянута" ещё на этапе настройки сетевых интерфейсов одной из соответствующих служб, и можно спокойно указывать эту цель как зависимость в пользовательской службе, поскольку `network.target` будет запущена в любом случае.

Типы служб

Службы различаются по типу запуска, и это следует учитывать при написании юнитов. Тип определяется параметром `Type=` в разделе `[Service]`:

- `Type=simple` (по умолчанию): *systemd* запустит эту службу незамедлительно. Процесс при этом не должен разветвляться (fork). Если после данной службы должны запускаться другие, то этот тип использовать не стоит (исключение — служба использует сокетную активацию).
- `Type=forking`: *systemd* считает службу запущенной после того, как процесс разветвляется с завершением родительского процесса. Используется для запуска классических демонов за исключением тех случаев, когда в таком поведении процесса нет необходимости. Укажите параметр `PIDFile=`, чтобы *systemd* мог отслеживать основной процесс.
- `Type=oneshot`: удобен для сценариев, которые выполняют одно задание и завершаются. Если задать параметр `RemainAfterExit=yes`, то *systemd* будет считать процесс активным даже после его завершения.
- `Type=notify`: идентичен параметру `Type=simple`, но с уточнением, что демон пошлет *systemd* сигнал готовности. Реализация уведомления находится в библиотеке *libsystemd-daemon.so*.
- `Type=dbus`: служба считается находящейся в состоянии готовности после появления указанного `BusName` в системной шине DBus.
- `Type=idle`: *systemd* отложит выполнение двоичного файла службы до окончания запуска остальных ("более срочных") задач. В остальном поведение аналогично `Type=simple`.

Подробнее о параметре `Type` см. **`systemd.service(5) §OPTIONS`**.

Редактирование файлов юнитов

Не стоит редактировать юнит-файлы пакетов напрямую, так как это приведёт к конфликтам с pacman. Есть два безопасных способа редактирования: создать новый файл, который полностью **заменит** оригинальный, или создать **drop-in файл**, который будет применяться поверх оригинального юнита. В обоих случаях после редактирования необходимо перезагрузить юнит, чтобы изменения вступили в силу. Это выполняется либо путем редактирования блока с помощью команды `systemctl edit`, которая автоматически перезагружает юнит, либо перезагрузкой всех юнитов командой:

```
# systemctl daemon-reload
```

Совет:

- С помощью *systemd-delta* можно узнать, какие файлы юнитов были переопределены и что конкретно было изменено.
- Команда `systemctl cat` *`юнит`* позволит просмотреть содержимое файла юнита и связанных с ним drop-in сниппетов.

Замещение файла юнита

Чтобы полностью заместить файл юнита `/usr/lib/systemd/system/`*`юнит`*, создайте файл с таким же именем `/etc/systemd/system/`*`юнит`* и **включите заново** юнит для обновления символических ссылок.

Альтернативный способ:

```
# systemctl edit --full юнит
```

Эта команда откроет файл `/etc/systemd/system/`*`юнит`* в текстовом редакторе (если файл ещё не существует, будет скопирован оригинал) и автоматически перезагрузит юнит после завершения редактирования.
Примечание: Новые изменённые юниты продолжат работать даже после того, как pacman обновит оригинальные юниты в будущем. Это может усложнить обслуживание системы, поэтому предпочтительнее использовать подход, описанный в следующем разделе.

Drop-in файлы

Чтобы создать drop-in файл для `/usr/lib/systemd/system/`*`юнит`*, создайте каталог `/etc/systemd/system/`*`юнит`*`.d/` и поместите в него файлы *.conf* с добавленными или изменёнными опциями. systemd будет анализировать эти файлы и применять их поверх оригинального юнита.

Самый простой способ — использовать команду:

```
# systemctl edit юнит
```

Команда откроет `/etc/systemd/system/юнит.d/override.conf` в текстовом редакторе (файл будет создан, если его ещё нет) и автоматически перезапустит юнит после завершения редактирования.
Примечание: Не все опции могут быть заменены в drop-in файле. Например, для изменения опции `Conflicts=` **придётся создать** полную замену файла юнита (см. предыдущий раздел).

Откат изменений

Отменить все изменения, сделанные с помощью `systemctl edit`, можно командой:

```
# systemctl revert юнит
```

Примеры

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

```
/etc/systemd/system/юнит.d/customdependency.conf
[Unit]
Requires=новая зависимость
After=новая зависимость
```

Другой пример: для замены `ExecStart` в юните (кроме типа `oneshot`) создайте следующий файл:

```
/etc/systemd/system/юнит.d/customexec.conf
[Service]
ExecStart=
ExecStart=новая команда
```

Обратите внимание, что `ExecStart` необходимо очистить перед присвоением нового значения [1]. Это относится ко всем параметрам, которые позволяют прописать несколько значений, вроде `OnCalendar` в таймерах.

Пример настройки автоматического перезапуска службы:

```
/etc/systemd/system/юнит.d/restart.conf
[Service]
Restart=always
RestartSec=30
```

Цели

Systemd использует юнит типа *цель* (target) для группировки юнитов по зависимостям и в качестве стандартизированных точек синхронизации. Они выполняют ту же задачу, что и **уровни запуска**, но действуют немного по-другому. Каждая цель имеет имя, а не номер, и предназначена для конкретных задач; несколько целей могут быть активны одновременно. Некоторые цели реализованы путём наследования служб из других целей с добавлением собственных. В systemd также имеются цели, имитирующие общие уровни запуска SystemVinit, поэтому вы можете переключаться между целями, используя привычную команду `telinit RUNLEVEL`.

Получение информации о текущих целях

В systemd для этого предназначена следующая команда (заменяющая `runlevel`):

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

Создание пользовательской цели

Уровни запуска, имеющие определённое значение в sysvinit (0, 1, 3, 5 и 6), один в один соответствуют конкретным целям systemd. К сожалению, не существует хорошего способа сделать то же самое для пользовательских уровней 2 и 4. Их использование предполагает, что вы создаёте новый юнит-цель с названием `/etc/systemd/system/цель`, который берет за основу один из существующих уровней запуска (взгляните, например, на `/usr/lib/systemd/system/graphical.target`), создаёте каталог `/etc/systemd/system/цель.wants`, а после этого — символические ссылки на те службы из каталога `/usr/lib/systemd/system/`, которые вы хотите включить при загрузке.

Соответствие уровней SysV целям systemd

Уровнень запуска SysV	Цель systemd	Примечания
0	runlevel0.target, poweroff.target	Выключение системы
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 переключать их такой командой:

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

Данная команда только изменит 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 ссылается на `graphical.target` (примерно соответствующего прежнему уровню запуска 5).

Узнать текущую цель можно так:

```
$ systemctl get-default
```

Для установки новой цели загрузки по умолчанию измените ссылку `default.target`. С помощью команды *systemctl* это делается так:

```
# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target.
```

Альтернативный способ — добавить один из следующих **параметров ядра** в загрузчик:

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

Порядок выбора цели по умолчанию

systemd выбирает `default.target` в следующем порядке :

1. Параметр ядра, описанный выше.
2. Символическая ссылка `/etc/systemd/system/default.target`.
3. Символическая ссылка `/usr/lib/systemd/system/default.target`.

Компоненты systemd

Некоторые (не все) составные части systemd:

- **systemd-boot** — простой **менеджер загрузки** для UEFI;
- **systemd-firstboot** — инициализация системных настроек при первой загрузке;
- **systemd-homed** — переносимые **аккаунты пользователей**;
- **systemd-logind** — управление сеансами;
- **systemd-networkd** — управление **сетевыми настройками**;
- **systemd-nspawn** — приложение для контейнеризации процессов;
- **systemd-resolved** — **разрешение** сетевых имён;
- **`systemd-sysusers(8)`** — создание системных пользователей/групп и добавление пользователей в группы при установке пакетов и загрузке системы;
- **systemd-timesyncd** — синхронизация **системных часов** по сети;
- **systemd/Журнал** — системные логи;
- **systemd/Таймеры** — таймеры для управления событиями и службами, альтернатива **cron**.

systemd.mount — монтирование

systemd полностью отвечает за монтирование разделов и файловых систем, описанных в файле `/etc/fstab`. **`systemd-fstab-generator(8)`** преобразует записи из `/etc/fstab` в юниты systemd; это выполняется при каждой загрузке системы, а также при перезагрузке конфигурации системного менеджера.

systemd расширяет возможности **fstab** и предлагает дополнительные опции монтирования. Они могут влиять на зависимости юнита монтирования: например, могут гарантировать, что монтирование выполняется только после подключения к сети или после монтирования другого раздела. Полный список опций монтирования systemd (обычно они имеют префикс `x-systemd`) описан в **`systemd.mount(5) § FSTAB`**.

Примером этих опций может быть т.н. *автомонтирование* (здесь имеется в виду не автоматическое монтирование во время загрузки, а монтирование при появлении запроса от устройства). Подробнее смотрите **fstab#Автоматическое монтирование с systemd**.

Автомонтирование GPT-раздела

На UEFI-системах **`systemd-gpt-auto-generator(8)`** автоматически монтирует **GPT**-разделы в соответствии с **Discoverable Partitions Specification**, поэтому их можно не указывать в файле **fstab**.

Требования:

- Загрузчик должен установить EFI-переменную **LoaderDevicePartUUID**, по которой можно будет определить системный раздел EFI. Эта возможность поддерживается в **systemd-boot**, а также в **rEFInd** (по умолчанию отключена). Это проверяется наличием строки `Boot loader sets ESP partition information` в выводе команды `bootctl`.
- Корневой раздел должен быть на одном физическом диске с системным разделом EFI. Автомонтируемые разделы должны быть на одном физическом диске с корневым разделом. Очевидно, монтируемые разделы должны оказаться на одном диске и с ESP.

Совет: Автомонтирование разделов отключается изменением его **GUID-типа** или установкой атрибута раздела "do not mount" (бит 63), см. **gdisk#Prevent GPT partition automounting**.

/var

Для автомонтирования раздела `/var` его PARTUUID должен совпадать с хэш-суммой SHA256 HMAC, вычисленной на основании UUID типа раздела. В качестве ключа хэша используется machine ID. Необходимый PARTUUID можно получить командой:

```
$ systemd-id128 -u --app-specific=4d21b016-b534-45c2-a9fb-5c16e091fd2d machine-id
```

Примечание: Утилита **systemd-id128(1)** считывает machine ID из файла `/etc/machine-id`, поэтому вычислить PARTUUID до установки системы невозможно.

systemd-sysvcompat

Пакет **systemd-sysvcompat** (зависимость пакета **base**) содержит традиционный бинарный файл **init**. В системах под управлением systemd `init` — символическая ссылка на исполняемый файл `systemd`.

Кроме того, в этом пакете находятся 4 команды **SysVinit** — **halt(8)**, **poweroff(8)**, **reboot(8)** и **shutdown(8)**. Это символические ссылки на `systemctl`, и их работа обусловлена логикой systemd. Подробнее см. **#Управление питанием**.

В systemd-системах отказаться от совместимости с System V можно либо задав **параметр загрузки** `init=` (см. **BBS#233387**), либо с помощью собственных аргументов команды `systemctl`.

systemd-tmpfiles — временные файлы

Утилита *systemd-tmpfiles* создает, удаляет и очищает непостоянные и временные файлы и каталоги. Она читает конфигурационные файлы из `/etc/tmpfiles.d/` и `/usr/lib/tmpfiles.d/`, чтобы понять, что необходимо 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

#    Path                 Mode UID  GID  Age Argument
w    /proc/acpi/wakeup    -    -    -    -   USBE
```

Подробнее смотрите **systemd-tmpfiles(8)** и **tmpfiles.d(5)**.

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

Советы и рекомендации

Программы настройки с графическим интерфейсом

- **systemadm** — Графический поисковик юнитов systemd. Выводит список юнитов, возможна фильтрация по типу.

https://cgit.freedesktop.org/systemd/systemd-ui/ || `systemd-ui`

- **SystemdGenie** — Утилита управления systemd на основе инструментов KDE.

https://invent.kde.org/system/systemdgenie || `systemdgenie`

Запуск сервисов после подключения к сети

Чтобы запустить сервис только после подключения к сети, добавьте такие зависимости в *.service* файле:

```
/etc/systemd/system/foo.service

[Unit]
...
Wants=network-online.target
After=network-online.target
...
```

Также должна быть включена служба ожидания сети того приложения, которое управляет сетью; только тогда `network-online.target` будет соответствовать состоянию сети.

- В **NetworkManager** служба `NetworkManager-wait-online.service` включается вместе с `NetworkManager.service`. Проверить состояние службы можно командой `systemctl is-enabled NetworkManager-wait-online.service`. Если служба не включена, то **включите заново** `NetworkManager.service` ещё раз.
- В случае **netctl** **включите** службу `netctl-wait-online.service`.
- Для пользователей **systemd-networkd** юнит `systemd-networkd-wait-online.service` включается вместе со службой `systemd-networkd.service`; проверьте это командой `systemctl is-enabled systemd-networkd-wait-online.service`. Если нет, то **включите заново** `systemd-networkd.service`.

Подробнее можно почитать: **Network configuration synchronization points**.

Если служба отправляет DNS-запросы, она должна запускаться также после `nss-lookup.target`:

```
/etc/systemd/system/foo.service
```

```
[Unit]
...
Wants=network-online.target
After=network-online.target nss-lookup.target
...
```

Подробнее см. **systemd.special(7)**.

Чтобы цель `nss-lookup.target` работала как положено, должна быть служба, которая запускает её параметром `Wants=nss-lookup.target` и размещает себя *перед* ней (`Before=nss-lookup.target`). Обычно это выполняет локальный **DNS-распознаватель**.

Чтобы узнать, какие службы зависят от `nss-lookup.target`, выполните:

```
$ systemctl list-dependencies --reverse nss-lookup.target
```

Включение установленных юнитов по умолчанию

This article or section needs expansion.
Reason: Как это работает для юнитов-экземпляров? (Discuss in **Talk:Systemd (Русский)**)

Arch Linux поставляется с файлом `/usr/lib/systemd/system-preset/99-default.preset`, в котором указан параметр `disable *`. Это означает, что *systemctl preset* отключает по умолчанию юниты и пользователь должен сам их включать после установки пакетов.

Если такое поведение не устраивает, создайте символическую ссылку `/etc/systemd/system-preset/99-default.preset` на `/dev/null` для переопределения файла конфигурации. Это заставит systemctl preset включать юниты новых пакетов — вне зависимости от типа — кроме указанных в других файлах из каталога настроек systemctl preset. Пользовательских юнитов это не касается. Подробнее смотрите **systemd.preset(5)**.

Примечание: Политика включения всех юнитов по умолчанию может привести к проблемам, если в установленном пакете находится несколько взаимоисключающих юнитов. В этом случае в файле preset-настроек придётся явно указать, какие юниты включаться не должны. Подробнее смотрите **systemd.preset(5)**.

Песочница для приложений

Юнит может быть использован в качестве песочницы для изоляции приложений и их процессов в виртуальном окружении. Systemd использует механизм **namespaces**, белые и чёрные списки **capabilities**, а также **control groups** для контейнеризации процессов при помощи настраиваемых окружений — см. **`systemd.exec(5)`**.

Добавление к существующему юниту systemd функциональности песочницы обычно происходит методом проб и ошибок вкупе с использованием различных инструментов логирования — **`strace`**, **stderr** и **`journalctl(1)`**. В таких случаях имеет смысл предварительно поискать соответствующую документацию от разработчиков. В качестве отправной точки для поиска путей повышения безопасности изучите вывод команды:

```
$ systemd-analyze security юнит
```

Рекомендации по созданию песочницы с помощью systemd:

- Параметр `CapabilityBoundingSet` определяет список разрешённых capabilities, но с его помощью можно также и запрещать некоторые capabilities для определённого юнита.
  - Например, можно задать capability `CAP_SYS_ADM`, **необходимую** для создания безопасной песочницы: `CapabilityBoundingSet=~ CAP_SYS_ADM`

Уведомление о неработающих службах

Для уведомления о неудачном запуске службы используется директива `OnFailure=` в соответствующем файле службы или **drop-in файле**. Чтобы эта директива возымела эффект для всех служб одновременно, её необходимо добавть в drop-in файл верхнего уровня, см. **`systemd.unit(5)`**.

Создайте drop-in верхнего уровня:

```
/etc/systemd/system/service.d/toplevel-override.conf
```

```
[Unit]
OnFailure=failure-notification@%n
```

Это добавит строку `OnFailure=failure-notification@%n` в файл каждой службы. Если *какой-то_юнит* завершится с ошибкой, запустится экземпляр службы `failure-notification@`*`какой-то_юнит`* для создания уведомления (или любой другой задачи, которая была назначена).

Создайте юнит-шаблон `failure-notification@`:

```
/etc/systemd/system/failure-notification@.service
```

```
[Unit]
Description=Send a notification about a failed systemd unit
After=network.target

[Service]
Type=simple
ExecStart=/путь/к/failure-notification.sh %i
```

После этого создайте сценарий `failure-notification.sh`, в котором определите, каким именно способом будет создаваться уведомление (mail, gotify, xmpp). Параметр `%i` будет заменён на название неудачно завершившегося юнита и будет передан сценарию в качестве аргумента.

Чтобы предотвратить регрессию экземпляров `failure-notification@.service`, создайте пустой файл drop-in настроек с именем, совпадающим с названием drop-in файла верхнего уровня (пустой файл "уровня служб" будет иметь приоритет над файлом "верхнего уровня"):

```
# mkdir -p /etc/systemd/system/failure-notification@.service.d
# touch /etc/systemd/system/failure-notification@.service.d/toplevel-override.conf
```

Решение проблем

Неудачно запущенные службы

Следующая команда найдёт все службы, которые не смогли выполнить запуск:

```
$ systemctl --state=failed
```

Чтобы определить причину, по которой служба не запустилась, необходимо изучить записи её логов. Подробнее см. **systemd/Журнал#Фильтрация вывода**.

Диагностика загрузки системы

В systemd есть несколько опций для диагностики проблем процесса загрузки. В статье об **отладке загрузки** описано, как получить доступ к сообщениям, выданным **процессом загрузки** до того, как systemd перехватил управление. Также смотрите **документацию по отладке systemd**.

Диагностика службы

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:

Добавьте **drop-in файл** для службы:

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

Или, как вариант, пропишите переменную окружения вручную:

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

После этого **перезапустите** systemd-networkd и следите за **журналом** службы с помощью опции `-f`/`--follow`.

Выключение/перезагрузка происходят ужасно долго

Если процесс выключения занимает очень долгое время (или выглядит зависшим), то, вероятно, виновата служба, которая не может завершить свою работу. Systemd ожидает некоторое время, пока каждая служба прекратит работу самостоятельно, и только потом пробует завершить её принудительно. Если вы столкнулись с такой проблемой, обратитесь к **Shutdown completes eventually** в systemd-вики.

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

Если команда `journalctl -u foounit` не даёт вывода для службы с коротким сроком жизни, вместо названия службы используйте её 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`. Поля метаданных для журнала вроде `_SYSTEMD_UNIT` и `_COMM` собираются асинхронно и полагаются на каталог `/proc` в случае с действующими процессами. Для решения проблемы требуется внести исправления в ядро, чтобы эти данные можно было собирать через сокет, наподобие `SCM_CREDENTIALS`. В общем, **это баг**. Имейте в виду, что быстро падающие службы могут не успеть оставить сообщения в журнале из-за особенностей systemd.

Время загрузки системы увеличивается с течением времени

После использования `systemd-analyze` некоторые пользователи заметили, что время загрузки системы значительно увеличилось. После использования `systemd-analyze blame` **NetworkManager** запускался необычно долго.

Проблема связана с тем, что файл `/var/log/journal` стал 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), а затем **ограничить** размер журнала.

systemd-tmpfiles-setup.service не запускается во время загрузки

Начиная с версии Systemd 219, `/usr/lib/tmpfiles.d/systemd.conf` определяет ACL-атрибуты для каталогов в `/var/log/journal` и, следовательно, требует **включённой** поддержки ACL для той файловой системы, в которой находится журнал.

Отключение emergency mode на удалённой машине

Вам может понадобиться отключить emergency mode на удалённой машине, например на виртуальных машинах Azure или Google Cloud. Это связано с тем, что в случае ухода системы в emergency mode она отключится от сети и лишит вас возможности подключения к ней.

Для отключения этого режима **замаскируйте** `emergency.service` и `emergency.target`.

Смотрите также

- **Wikipedia:ru:systemd**
- **Официальный веб-сайт (англ.)**
  - **Оптимизации systemd**

    - **systemd FAQ**
    - **systemd Советы и трюки**
- **`systemd(1)`**
- Другие дистрибутивы
    - **Gentoo:Systemd**
    - **Fedora:Systemd**
    - **Fedora:How to debug Systemd problems** — отладка systemd.
    - **Fedora:SysVinit to Systemd Cheatsheet** — памятка по переходу с SysVinit на systemd.
    - **Debian:systemd**
- **Блог Lennart'a (англ.)**, **update 1**, **update 2**, **update 3**, **Why systemd?**
- **Debug Systemd Services** — отладка юнитов systemd.
- **systemd для администраторов (PDF)** - перевод **цикла статей** Леннарта Поттеринга (Lennart Poettering)
- **How To Use Systemctl to Manage Systemd Services and Units**
- **Session management with systemd-logind**
- **Emacs Syntax highlighting for Systemd files (англ.)**
- **часть 1** и **часть 2** вводной статьи в журнале *The H Open* (англ.)

Category:

- Init (Русский)