Содержание

Глава 6: Запуск пользовательского пространства

Пользовательское пространство запускается в следующем порядке:

1. Процесс init.
2. Основные службы низшего уровня, такие как udevd и syslogd.
3. Конфигурация сети.
4. Службы среднего и высокого уровня (cron, print и т.д.).
5. Приглашения для входа в систему, графические интерфейсы и приложения высокого уровня, такие как веб-серверы.

6.1 Основные сведения об init

init (инициализация) - программа пользовательского пространства, расположенная в /sbin. Её основная цель - запускать и останавливать основные служебные процессы системы.

Во всех актуальных версиях основных дистрибутивов Linux стандартной реализацией init является **systemd**.

Альтернативные системы init

В старых системах встречаются другие реализации:

Система	Описание	Использование
System V init	Традиционная последовательная инициализация (Sys V из Unix System V)	RHEL до 7.0, Debian 8
Upstart	Реализация для дистрибутивов Ubuntu	Ubuntu до 15.04
runit	Облегченная система инициализации	Встроенные платформы и облегченные системы
Android init	Собственная система инициализации	Android
BSD init	Версия для BSD	Редко встречается в современных Linux системах

Проблемы System V init

Система инициализации System V работает как **последовательная серия сценариев**, где каждый сценарий запускает одну службу или настраивает отдельную часть системы. Обычно зависимости разбираются просто, и система достаточно гибка для необычных задач.

Однако существуют существенные ограничения:

Производительность:

- Две части последовательной загрузки не могут выполняться одновременно

Управление системой:

- Сложность поиска PID демона службы (требуется ps или механизм, специфичный для службы)
- Полустандартная система записи PID (/var/run/myservice.pid)
- Много стандартного шаблонного кода в сценариях

Конфигурация:

- Мало информации о службах и конфигурации по требованию
- Большинство служб запускается только при загрузке
- Конфигурация устанавливается в основном при загрузке
- Традиционный демон inetd (для сетевых служб по требованию) уже не используется

Современные системы инициализации (такие как systemd) решают эти проблемы путём изменения способа запуска служб, порядка их контроля и настройки зависимостей.

6.2 Определение системы инициализации

Определить версию инициализации можно, проверив наличие определённых файлов и каталогов:

systemd:

- Наличие каталогов /usr/lib/systemd и /etc/systemd

Upstart:

- Наличие каталога /etc/init с несколько файлами .conf
- (Исключение: Debian 7 и старше используют System V init, несмотря на /etc/init)

System V init:

- Отсутствие вышеуказанных каталогов
- Наличие файла /etc/inittab

Информацию также можно получить из страницы руководства init(1).

6.3 systemd

systemd - современная реализация init в Linux, интегрирующая функциональность Unix-служб (cron, inetd). Вдохновлена launchd от Apple.

Основные отличия от традиционной init:

- Расширенное управление службами
- Отслеживание демонов после запуска
- Группировка процессов службы
- Ориентирована на выполнение задач и целей

Система использует юниты (units) для задач, содержит инструкции запуска и зависимости. Активирует юниты по мере готовности, не придерживаясь жесткой последовательности. Реагирует на системные события (uevents).

6.3.1 Юниты и типы юнитов

systemd управляет не только процессами/службами, но и монтированием ФС, запросами подключения сети, таймерами.

Тип юнита	**Описание**
Service units	Управление служебными демонами Unix
Target units	Управление другими юнитами, группировка
Socket units	Местоположение запросов входящих сетевых подключений
Mount units	Контроль присоединения файловых систем

Примечание: полный список типов см. systemd(1)

6.3.2 Графики загрузки и зависимостей юнитов

При загрузке активируется целевой юнит **default.target**, объединяющий юниты служб и монтирования как зависимости. Зависимости образуют граф (не дерево). Юниты могут зависеть от нескольких предыдущих.

Визуализировать граф: `systemd-analyze dot` (требует фильтрации для сложных систем).

Примечание: В большинстве систем default.target → целевой юнит высокого уровня (графический интерфейс и т.д.)

6.3.3 Конфигурация systemd

Основные каталоги:

Каталог	**Назначение**
/lib/systemd/system или /usr/lib/systemd/system	Системные юниты (глобальная конфигурация)
/etc/systemd/system	Локальная конфигурация

Правило: не изменяйте системные юниты (сохраняет дистрибутив), вносите изменения в /etc/systemd/system.

Проверка пути поиска:

```
systemctl -p UnitPath show
pkg-config systemd --variable=systemdsystemunitdir
pkg-config systemd --variable=systemdsystemconfdir
```

Юнит-файлы

Формат из спецификации XDG Desktop Entries (.ini подобный) с секциями в квадратных скобках и переменными.

Пример dbus-daemon.service:

```
[Unit]
Description=D-Bus System Message Bus
Documentation=man:dbus-daemon(1)
Requires=dbus.socket
RefuseManualStart=yes

[Service]
ExecStart=/usr/bin/dbus-daemon --system --address=systemd: --nofork --
nopidfile --systemd-activation --syslog-only
ExecReload=/usr/bin/dbus-send --print-reply --system --type=method_call --
dest=org.freedesktop.DBus / org.freedesktop.DBus.ReloadConfig
```

[Unit] - описание, информация о зависимостях **[Service]** - подготовка, запуск, перезагрузка службы

Переменные

Синтаксис: `$ПЕРЕМЕННАЯ`

Пример (sshd.service):

```
[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStartPre=/usr/sbin/sshd-keygen
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
ExecReload=/bin/kill -HUP $MAINPID
```

- **$OPTIONS, $CRYPTO_POLICY** - из EnvironmentFile
- **$MAINPID** - PID отслеживаемого процесса

Спецификаторы

Синтаксис: `%спецификатор`

- **%n** - имя текущего юнита
- **%H** - имя хоста
- **%I, %i** - имя экземпляра (для юнитов с @)

Пример: getty@.service → getty@tty1, getty@tty2 (динамическое создание).

Полный список см. systemd.unit(5)

6.3.4 Процесс работы systemd

systemctl - основная команда для управления systemd юнитами.

Основные команды

Операция	Команда	Описание
Список активных юнитов	systemctl list-units	Вывод сокращен; используйте –full для полных имен
Все юниты	systemctl list-units –all	Включает неактивные юниты
Статус юнита	systemctl status <unit>	Показывает статус, PID, cgroup, журнал
Запуск	systemctl start <unit>	Активация юнита
Остановка	systemctl stop <unit>	Деактивация юнита
Перезапуск	systemctl restart <unit>	Перезапуск юнита
Перезагрузка конфига	systemctl reload <unit>	Перезагрузка только для данного юнита
Перезагрузка всех	systemctl daemon-reload	Перезагрузка всех конфигураций
Текущие задания	systemctl list-jobs	Список активных заданий (изменений состояния)
Журнал юнита	journalctl –unit=<unit>	Все сообщения журнала

Пример просмотра статуса

```
$ systemctl status sshd.service
```

Вывод содержит: статус загрузки, состояние (active/inactive), основной процесс (Main PID), контрольную группу (cgroup) и сообщения журнала.

Добавление юнита

Размещайте юнит-файлы в **/etc/systemd/system** (не перезаписываются при обновлении дистрибутива).

```
# Создать файл /etc/systemd/system/test1.target
[Unit]
Description=test 1

# Создать файл /etc/systemd/system/test2.target с зависимостью
[Unit]
Description=test 2
Wants=test1.target

# Активировать
systemctl start test2.target

# Если есть секция [Install], включить перед активацией
systemctl enable test2.target
```

Удаление юнита

```
# 1. Деактивировать
systemctl stop <unit>

# 2. Отключить (если есть [Install])
systemctl disable <unit>

# 3. Удалить юнит-файл
```

Примечание: Задания (jobs) - это изменения состояния юнитов, не путать с заданиями оболочки. Требуются права суперпользователя для некоторых операций.

6.3.5 Отслеживание и синхронизация процессов systemd

systemd использует **cgroups** (группы управления ядра Linux) для отслеживания иерархии процессов. Параметр **Type** в юнит-файле указывает поведение службы при запуске.

Type	Описание
simple	Процесс не разветвляется, остается основным процессом
forking	Процесс разветвляется; systemd ждет завершения исходного процесса
notify	Служба отправляет уведомление systemd, когда готова
dbus	Служба регистрируется на D-Bus, когда готова
oneshot	Процесс полностью завершается после запуска; RemainAfterExit=yes по умолчанию
idle	Как simple, но запуск отложен до завершения всех активных заданий

Примечание: Type=simple не известен время запуска; используйте Type=notify, Type=dbus или отложенный запуск (раздел 6.3.7) для синхронизации зависимых юнитов.

6.3.6 Зависимости systemd

Гибкая система зависимостей требует баланса между строгостью и отказоустойчивостью. Слишком строгие правила могут заблокировать доступ при сбое некритичного сервиса. systemd предлагает несколько типов зависимостей для гибкого управления:

Тип	Описание
Requires	Строгие зависимости. При сбое зависимости юнит также отключается.
Wants	Зависимости только для активации. Сбой зависимости не влияет на юнит. Рекомендуется использовать по возможности.
Requisite	Юниты, которые уже должны быть активны. Если не активны, юнит не запустится.
Conflicts	Отрицательные зависимости. При активации юнита противоположная зависимость деактивируется.

Просмотр зависимостей:

```
systemctl show -p type unit
```

Порядок выполнения юнитов

По умолчанию юниты с Requires и Wants запускаются одновременно. Для упорядочивания используйте:

- **Before** - текущий юнит активируется раньше перечисленных
- **After** - текущий юнит активируется после перечисленных

systemd ждет активного статуса юнита перед активацией зависимостей.

Зависимости по умолчанию

systemd автоматически добавляет зависимость After к Wants-зависимостям. Эти неявные зависимости вычисляются во время загрузки и не сохраняются в файлах конфигурации.

Отключить зависимости по умолчанию:

```
DefaultDependencies=no
```

Условные зависимости

Для проверки состояния системы:

- **ConditionPathExists=p** - путь существует
- **ConditionPathIsDirectory=p** - путь является каталогом
- **ConditionFileNotEmpty=p** - файл существует и не пуст

При ложном условии юнит не активируется, но другие зависимости активируются независимо от условия.

Секция [Install] и включение юнитов

Раздел [Install] позволяет указывать зависимости в обратном порядке без изменения дополнительных файлов. Используйте **WantedBy** или **RequiredBy**.

Пример:

test1.target:

```
[Unit]
Description=test 1

[Install]
WantedBy=test2.target
```

test2.target:

```
[Unit]
Description=test 2
```

Включение юнита:

```
systemctl enable test1.target
# Создаст: /etc/systemd/system/test2.target.wants/test1.target
```

Отключение юнита:

```
systemctl disable test1.target
```

Важно: включение юнита не активирует его. Юнит с разделом [Install] считается отключенным по умолчанию и включается после перезагрузки.

Каталоги .wants и .require в /etc/systemd/system обычно управляются разделом [Install], но можно добавлять символические ссылки вручную для временных зависимостей без редактирования системных файлов юнитов.

6.3.7 Запуск по запросу и параллелизация ресурсов в systemd

systemd позволяет откладывать запуск юнита до момента, когда на него поступит запрос. Процесс:

```
1. Создается юнит systemd для сервиса (Unit A)
2. Определяется ресурс (сокет, файл, устройство)
3. Создается юнит ресурса (Unit R): .socket, .path или .device
4. systemd мониторит ресурс и активирует Unit A при обращении
```

При обращении к ресурсу:

```
1. systemd блокирует ресурс и буферизует входные данные
2. Активирует Unit A
3. Unit A берет управление ресурсом и обрабатывает буферизованные данные
```

Важные замечания:

- Юнит ресурсов должен охватывать все точки доступа сервиса
- Связь между юнитами может быть неявной (по имени) или явной (параметр Socket=)
- Не все серверы поддерживают взаимодействие с юнитами ресурсов

Пример: сокет-юнит и служебный юнит

Echo-сервис на TCP-порту 22222.

echo.socket:

```
[Unit]
```

```
Description=echo socket

[Socket]
ListenStream=22222
Accept=true
```

echo@.service:

```
[Unit]
Description=echo service

[Service]
ExecStart=/bin/cat
StandardInput=socket
```

Запуск:

```
systemctl start echo.socket
telnet localhost 22222
```

Остановка:

```
systemctl stop echo.socket
```

Экземпляры и передача управления

Символ @ в имени echo@.service означает параллельные экземпляры. При **Accept=true** в сокет-юните systemd принимает входящие соединения и создает отдельный экземпляр для каждого. Каждый экземпляр считывает данные как стандартный ввод.

Если сервис может самостоятельно принимать соединения, исключите @ из имени и не устанавливайте Accept=true. Сервис получит полный контроль над сокетом.

Документация: systemd.socket(5), systemd.path(5), systemd.device(5), systemd.exec(5)

Оптимизация загрузки с помощью юнитов ресурсов

Юниты ресурсов ускоряют загрузку путем параллельной активации:

Последовательный запуск	**Параллельный запуск с юнитом ресурса**
Служба E (ресурс R) запускается	Юнит R активируется немедленно
Службы A, B, C ждут E	Службы A, B, C, E стартуют параллельно
Долгое время загрузки	Быстрая загрузка, E берет управление когда готова

systemd предоставляет интерфейс юнита ресурса (R) еще до полной активации служебного юнита (E). Это позволяет зависимым юнитам (A, B, C) стартовать параллельно, а не ждать E.

Побочный эффект: Одновременный запуск многих юнитов может временно замедлить

систему.

Пример: journald и D-Bus обычно загружаются параллельно этим способом.

6.3.8 Вспомогательные компоненты systemd

systemd включает поддержку задач управления системой, помимо запуска и управления сервисами. Исполняемые файлы расположены в /lib/systemd и имеют префикс **systemd-**.

Служба	Назначение
udevd	Управление устройствами (см. Глава 3)
journald	Служба логирования; обрабатывает различные механизмы, включая syslog (см. Глава 7)
resolved	Демон кэширования DNS (см. Глава 9)

Примеры программ:

- systemd-udevd (интегрированный в systemd demon udevd)
- systemd-fsck (простая обертка для стандартных системных утилит)

Некоторые программы в /lib/systemd - обычные обертки, запускающие стандартные утилиты и уведомляющие systemd о результатах.

Справка: Неизвестную программу в /lib/systemd можно найти в man-страницах, где описаны и утилита, и соответствующий ей тип юнита.

6.4. Уровни выполнения в System V

System V init использует **уровни выполнения** (runlevel) для определения состояния машины - число от 0 до 6. Проверить текущий уровень:

```
who -r
```

Выход: `run-level 5 2019-01-27 16:43`

Уровни выполнения различают состояния: запуск, выключение, однопользовательский режим, консоль. Например, 2–4 - текстовая консоль, 5 - графический интерфейс.

Примечание: systemd поддерживает уровни выполнения как устаревшие, предпочитая им целевые юниты.

6.5. System V init

System V init - одна из старейших систем инициализации Linux. Устанавливается редко (встречается в RHEL < 7.0, встроенных системах), но пакеты могут содержать её сценарии, совместимые с systemd.

Состоит из:

- Конфигурационный файл: **`/etc/inittab`**
- Сценарии загрузки и символические ссылки

Файл	**Содержимое**
/etc/inittab	Четыре поля: идентификатор : уровни выполнения : действие : команда
/etc/rc.d/rc	Запускает программы из rc*.d в числовом порядке
/etc/rc*.d	Каталоги со сценариями (rc1.d, rc2.d и т.д.)

Пример строки inittab:

```
id:5:initdefault:
l5:5:wait:/etc/rc.d/rc 5
```

Действие	**Назначение**
initdefault	Уровень выполнения по умолчанию
wait	Выполнить команду один раз при входе на уровень, ждать завершения
respawn	Перезапустить команду при выходе (например, getty для виртуальных консолей)
ctrlaltdel	Действие при нажатии Ctrl+Alt+Del (обычно shutdown)
sysinit	Первое действие при запуске, до всех уровней выполнения

6.5.1. System V init: последовательность команд при запуске

Команда `rc 5` запускает сценарии из `/etc/rc5.d`:

- **S** (Start) - выполнить с аргументом `start`
- **K** (Kill) - выполнить с аргументом `stop`
- **Числа 00–99** - порядок выполнения

Пример каталога rc5.d:

```
S10sysklogd      S20ppp           S99gpm
S12kerneld       S25netstd_nfs    S99httpd
S15netstd_init   S30netstd_misc   S99rmnologin
```

Команда rc выполняет их последовательно:

```
S10sysklogd start
S12kerneld start
S15netstd_init start
...
S99sshd start
```

Сценарии оболочки запускают программы из `/sbin` или `/usr/sbin`. Для редактирования используйте `less` или аналогичные программы.

6.5.2. Ферма ссылок System V init

Содержимое rc*.d - **символические ссылки** на файлы в `/etc/init.d`:

```
lrwxrwxrwx . . . S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx . . . S12kerneld -> ../init.d/kerneld
lrwxrwxrwx . . . S99httpd -> ../init.d/httpd
```

Дистрибутивы используют эту **ферму ссылок (link farm)** для переиспользования одних сценариев на всех уровнях выполнения.

Запуск и остановка служб

```
/etc/init.d/httpd start     # Запустить
/etc/init.d/httpd stop      # Остановить
```

Изменение последовательности загрузки

Отключить службу без удаления:

```
mv S99httpd _S99httpd
```

Префикс _ заставит rc игнорировать ссылку.

Добавить службу: скопируйте сценарий в init.d, создайте символическую ссылку в rc*.d. Несущественные службы - номера 90+, после основных систем.

6.5.3. Команда run-parts

Утилита **run-parts** запускает все исполняемые программы в каталоге в определённом порядке.

Дистрибутив	**Функции**
Fedora	Простая версия - запускает всё подряд
Debian/Ubuntu	Сложная - фильтр по регулярным выражениям, передача аргументов

Используется в сценариях для массового запуска программ. Большинству пользователей детали неважны.

6.5.4. Управление System V init

telinit - команда управления уровнями выполнения:

Команда	**Действие**
`telinit 3`	Переключиться на уровень выполнения 3

Команда	Действие
`telinit q`	Перечитать `/etc/inittab`
`telinit s`	Однопользовательский режим

Предупреждение: init завершит процессы, отсутствующие в inittab для нового уровня.

6.5.5. Совместимость systemd и System V

systemd отслеживает System V сценарии:

1. Активирует ''<N>.target'' (N - уровень выполнения)
2. Идентифицирует скрипты в ''/etc/rc<N>.d''
3. Связывает скрипт со служебным юнитом (''foo.service'')
4. Активирует юнит, запускает с аргументом start/stop
5. Связывает процессы со служебным юнитом

Результат: можно использовать `systemctl` для управления, но режим совместимости всё равно запускает сценарии **последовательно**.

6.6 Завершение работы системы

Используйте **shutdown** для корректного завершения:

```
shutdown -h now          # Остановить сейчас
shutdown -r +10          # Перезагрузить через 10 минут
```

Параметр	Действие
-h	Halt - остановка и отключение питания
-r	Reboot - перезагрузка
now	Немедленно
+n	Через n минут

Перед завершением: shutdown создаёт `/etc/nologin` (запрет входа), уведомляет пользователей.

Процедура завершения

1. init просит каждый процесс завершиться
2. Через время применяет сигнал TERM
3. Если не помогает - KILL для оставшихся процессов
4. Блокирует системные файлы
5. Демонтирует все FS кроме корневой
6. Ремонтирует корневую FS в режиме read-only
7. Синхронизирует буфер (sync)
8. Указывает ядру перезагрузиться/остановиться (reboot(2))

halt/reboot: вызывают shutdown, или если система уже на уровне выключения - сигналят ядру напрямую. Флаг **-f** (force) - быстрое выключение без проверок.

6.7 Начальная файловая система оперативной памяти

initramfs - временное пользовательское пространство перед основной загрузкой. Решает проблему: **ядру нужны драйверы в виде модулей, но модули - это файлы, а FS не смонтирована.**

Как это работает

1. Загрузчик загружает архив с модулями ядра в оперативную память
2. Ядро распаковывает архив во временную RAM-FS
3. Утилиты в `initramfs` загружают необходимые драйверы
4. Монтируют реальную корневую FS
5. Запускают полноценный `init`

Дистрибутив	Реализация
Простые системы	Shell-скрипт, udev, монтирование корня
systemd системы	Полная systemd без модулей, minimal udev config

Отключение initramfs

Если все драйверы встроены в ядро - можно пропустить initramfs. Удалите строку `initrd` в GRUB (лучше через редактор меню, чем файл конфига).

Примечание: современные ядра требуют initramfs для UUID-монтирования и других функций.

Содержимое initramfs

Распакуйте архив:

```
unmkinitramfs /boot/initramfs-*.img .
```

Старые системы использовали **cpio** архивы (см. cpio(1)).

Создание initramfs

Обычно создаётся автоматически дистрибутивом. Основные утилиты:

- **mkinitramfs** - простая, распространённая
- **dracut** - более мощная

Примечание: термин initrd (initial RAM disk) - устаревший, использовал образ диска. Сейчас initramfs (cpio-архив) стандарт. Файлы конфига ещё содержат имя initrd.

6.8 Аварийная загрузка системы и однопользовательский режим

При проблемах с системой используйте **live-образ** дистрибутива или SystemRescueCD на съёмном носителе. Live-образ - полная Linux, загружающаяся без установки.

Задачи после сбоя

- Проверка файловых систем
- Сброс забытого пароля
- Исправление критичных файлов (`/etc/fstab`, `/etc/passwd`)
- Восстановление резервных копий

Однопользовательский режим

Быстрая загрузка в root-оболочку без служб.

Init система	Режим
System V	Уровень выполнения 1
systemd	rescue.target
Параметр загрузчика	**-s**

Недостатки: сеть часто недоступна, нет GUI, терминал может работать некорректно. **Live-образ более предпочтителен.**

Требует: пароль суперпользователя для входа.