Процесс init и файл /etc/inittab

Если в параметре начальной загрузки "init=" не задан запуск какой-то другой программы, после монтирования корневой файловой системы в режиме "только для чтения" ядро запускает процесс `init`, который, как известно, является родоначальником всех других процессов в Linux. Сам по себе init в принципе ничем не отличается от других программ в системе Linux, просто это первая (и единственная) программа, которая запускается непосредственно ядром, все остальные процессы являются потомками процесса init. Файл программы init вы можете найти в каталоге /sbin среди других исполняемых файлов.

Init отвечает за продолжение процедуры загрузки, и перевод системы от начального состояния, возникающего после загрузки ядра, в стандартное состояние обработки запросов многих пользователей. Основная задача, которая стоит перед `init`,заключается в том, чтобы запускать в определенной последовательности другие программы в процессе загрузки системы и останавливать процессы в случае переключения уровня выполнения (в частности, при остановке системы). Init выполняет еще массу различных операций, необходимых для дальнейшей работы системы: проверку и монтирование файловых систем, запуск различных служб (демонов), запуск процедур логирования, оболочек пользователей на различных терминалах и т.д. Но, прежде чем рассматривать работу процесса init более детально, нужно сказать несколько слов о стилях загрузки и так называемых "уровнях выполнения".

Уровни выполнения

Уровни выполнения (run levels) – это просто несколько стандартных вариантов загрузки системы, каждый из которых определяет перечень действий, выполняемых процессом `init`, и состояние системы после загрузки, т. е. конфигурацию запущенных процессов. Уровень выполнения идентифицируется одним символом. В большинстве дистрибутивов ОС Linux используется 6 основных уровней выполнения. В следующей таблице показано уровни используются в Red Hat:

Уровень	**Red Hat**
0	остановка системы
1	однопользовательский режим
2	многопользовательский режим без NFS (то же, что и 3, если компьютер не работает с сетью)
3	полный многопользовательский режим
4	использование не регламентировано
5	запуск системы в графическом режиме
6	перезагрузка системы;

В некоторых дистрибутивах (например, в Debian), кроме того, используется дополнительный уровень S (или s) — примерно то же, что и однопользовательский режим, но S и s используются в основном в скриптах.

Как видите, уровни 0, 1 и 6 зарезервированы для особых случаев. Относительно того, как использовать уровни со 2 по 5, единого мнения не существует. Некоторые системные администраторы используют разные уровни для того, чтобы задать разные варианты работы, например, на одном уровне запускается графический режим, на другом работают в сети и т. д. Вы можете сами решить, как использовать разные уровни для создания разных вариантов загрузки. Но для начала проще всего воспользоваться тем способом определения разных уровней, который был задан при установке.

Уровень выполнения может быть задан как одна из опций, передаваемых ядру загрузчиком. Обычно единственной причиной, по которой уровень загрузки может быть задан как аргумент при загрузке, является необходимость запуска системы в однопользовательском режиме (уровень выполнения 1) для выполнения каких-то административных задач или в случае повреждения диска. Но если уровень выполнения не задан как опция загрузки, то **init** будет загружать систему на уровень, заданный в файле `/etc/inittab`.

Если вам нужно узнать, на каком уровне выполнения работает ваша система, то можно воспользоваться командой **runlevel**. Будучи запущенной без параметров, она сообщает предыдущий и текущий уровень выполнения в виде двух цифр (лучше сказать символов, потому что существуют уровни S и s), разделенных пробелом. Если уровень выполнения не изменялся, то первый символ (идентифицирующий предыдущий уровень) примет значение N.

Изменять уровень выполнения в в работающей системе можно с помощью команды **telinit**. Естественно, что для ее запуска нужно иметь права суперпользователя. Но об этой команде мы поговорим чуть позже, а пока давайте посмотрим что из себя представляет файл `/etc/inittab`.

Файл inittab

Конфигурационный файл /etc/inittab состоит из отдельных строк. Если строка начинается со знака # или пуста, то она игнорируется. Все остальные строки состоят из 4 полей, разделенных двоеточиями:

```
id:runlevels:action:process
```

где:

- `id` — идентификатор строки. Это произвольная комбинация, содержащая от 1 до 4 символов. В файле inittab не может быть двух строк с одинаковыми идентификаторами;
- `runlevels` — уровни выполнения, на которых эта строка будет задействована. Уровни задаются цифрами или буквами без разделителей, например, 345;
- `process` — процесс, который должен запускаться на указанных уровнях. Другими словами в этом поле указывается имя программы, вызываемой при переходе на указанные уровни выполнения;
- `action` — действие.

В поле `action` стоит ключевое слово, которое определяет дополнительные условия выполнения команды, заданной полем `process`. Допустимые значения поля `action`:

- `respawn` — перезапустить процесс в случае завершения его работы;

Кодовое слово `respawn` заставляет init запустить команду, которая указана в этой строке и, если вызванная программа завершает работу, запустить ее снова. Возьмем, например, следующую строку в файле inittab:

```
1:2345:respawn:/sbin/mingetty tty1
```

Программа `getty` обеспечивает открытие виртуальной консоли (в данном случае - первой) и запуск в ней программы login, выводящей приглашение ко входу пользователя в систему. Ключевое слово `respawn` в приведенной строке означает, что после того, как пользователь выйдет из системы, приглашение login будет выведено на первую виртуальную консоль снова.

- `once` — выполнить процесс только один раз при переходе на указанный уровень;
- `wait` — процесс будет запущен один раз при переходе на указанный уровень и `init` будет ожидать завершения работы этого процесса, прежде, чем продолжать работу; Например, появление этого слова в строке

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

означает, что init при переходе на 5-ый уровень запустит на выполнение команду /etc/rc.d/rc 5 и будет ожидать окончания работы этой программы прежде чем приступить к чему-либо другому.

- `sysinit` — это ключевое слово обозначает самые первые действия, выполняемые процессом `init` еще до перехода на какой-либо уровень выполнения (поле `id` игнорируется). Процессы, помеченные этим словом, запускаются до процессов, помеченных словами `boot` и `bootwait`;
- `boot` — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения;
- `bootwait` — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения, и `init` будет дожидаться его завершения;
- `initdefault` — строка, в которой это слово стоит в поле action, определяет уровень выполнения, на который система переходит по умолчанию. Поле process в этой строке игнорируется. Если уровень выполнения, используемый по умолчанию, не задан, то процесс `init` будет ждать, пока пользователь, запускающий систему, не введет его с консоли;
- `off` — игнорировать данный элемент;
- `powerwait` — позволяет процессу `init` остановить систему, когда пропало питание. Использование этого слова предполагает, что имеется источник бесперебойного питания (UPS) и программное обеспечение, которое отслеживает состояние UPS и информирует `init` о том, что питание отключилось;
- `ctrlaltdel` — разрешает `init` перезагрузить систему, когда пользователь нажимает комбинацию клавиш <Ctrl>+<Alt>+<Del> на клавиатуре. Обратите внимание на то, что системный администратор может определить действия по комбинации клавиш <Ctrl>+<Alt>+<Del>, например игнорировать нажатие этой комбинации (что вполне разумно в системе, где много пользователей).

Этот список не является исчерпывающим. Более подробно о файле inittab можно узнать из man-страниц init (8), inittab (5) и getty (8).

Стили начальной загрузки

А теперь давайте вспомним, что существует два разных стиля начальной загрузки операционной системы типа UNIX, происхождение которых уходит корями в историю развития UNIX-систем: так называемый стиль BSD (используемый также в таких системах как FreeBSD, NetBSD и OpenBSD), и стиль System V (или стиль ATT). Различие между ними проявляется в организации и размещении стартовых сценариев (скриптов), обеспечивающих управление процессами загрузки системы. В классических BSD-системах эти файлы хранятся в каталоге /etc и их имена начинаются с префикса "rc". В системах семейства System V файлы сценариев располагаются в каталоге /etc/init.d, а ссылки на них созданы в каталогах /etc/rc0.d, /etc/rc1.d и т.д. Как пишет В.Попов в статье [42] главное отличие стиля BSD от стиля System V состоит в организации скриптов **rc**: если в системах стиля System V вызывается один и тот скрипт **rc**, только ему передается параметр, задающий уровень выполнения, то для систем BSD-стиля характерно наличие отдельного скрипта для каждого из уровней выполнения. Вариант организации в стиле System V является более четким и позволяет аккуратнее выполнять останов системы. Если вы хотите подробнее узнать о различиях в двух названных стилях загрузки, прочитайте главу 2 книги [41] или статьи [42]-[45].

Большая часть дистрибутивов Linux использует на этапе загрузки стиль System V. К этому классу относятся Debian, все клоны Red Hat, включая Mandrake и российские дистрибутивы ASPlinux и ALT Linux. В стиле BSD организована загрузка в дистрибутивах Gentoo). Однако тот или иной стиль сценариев начальной загрузки выдерживается не очень четко. Поскольку

стиль System V взят за основу при создании стандарта LSB (Linux Standart Base), дистрибутивы, ранее использовавшие стиль BSD. Структура каталогов, в которых хранятся инициализационные скрипты, в основных дистрибутивах существенно различается:

Листинг

Fedora (Red Hat, Centos)	**/etc/init.d/**-----символьная ссылка на **/etc/rc.d/init.d/** **/etc/rc.d/**------скрипты **rc, rc.sysinit** и **rc.local** | **/init.d/**-----множество файлов-скриптов **/rc0.d/**-----символьные ссылки **@K*.*** и **@S*.*** **/rc1.d/**-----символьные ссылки **@K*.*** и **@S*.*** ..... **/rc6.d/**-----символьные ссылки **@K*.*** и **@S*.*** **/rcS.d/**-----символьные ссылки **@K*.*** и **@S*.***
Debian	**/etc/init.d/**-----скрипты **rc, rcS, reboot** и множество других, **/etc/rc0.d/**-----символьные ссылки **@K*.*** и **@S*.*** **/rc1.d/**-----символьные ссылки **@K*.*** и **@S*.*** ..... **/rc6.d/**-----символьные ссылки **@K*.*** и **@S*.*** **/rcS.d/**-----символьные ссылки **@K*.*** и **@S*.***

Как видите, даже в Red Hat и Debian, которые следуют стилю System V, структура каталогов несколько отличается. А вот Red Hat, наоборот, завел скрипт rc.local, напоминающий одноименный сценарий во FreeBSD. В процессе загрузки он выполняется последним. Правда, в последних версиях Red Hat и Fedora Core этот скрипт "пустой", то есть практически не используется. И авторы книги [41] не советуют добавлять в него собственные команды, рекомендуя лучше воспользоваться средствами System V.

Но структура каталогов и названия инициализационных скриптов, может быть, и не самое главное. Более существенно то, что в этих скриптах содержится и как они используются. Во всех дистрибутивах кроме приведенных инструкций определяются еще действия по комбинации клавиш <Ctrl>+<Alt>+<Del> и выполняется запуск виртуальных терминалов

Fedora Core (Red Hat)	`id:3:initdefault:`
	
	`# System initialization.`
	`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`
	
	`x:5:respawn:/etc/X11/prefdm -nodaemon`

Debian	`id:5:initdefault:`
	
	`# Boot-time system configuration/initialization script.`
	`si::sysinit:/etc/init.d/rcS`
	
	`# What to do in single-user mode.`
	`~~:S:respawn:/bin/bash -login >/dev/tty1 2>&1 </dev/tty1`
	
	`l0:0:wait:/etc/init.d/halt`
	`l1:1:wait:/etc/init.d/rc 1`
	`l2:2:wait:/etc/init.d/rc 2`
	`l3:3:wait:/etc/init.d/rc 3`
	`l4:4:wait:/etc/init.d/rc 4`
	`l5:5:wait:/etc/init.d/rc 5`
	
	`# Run X Window session from CDROM in runlevel 5`
	`w5:5:wait:/bin/sleep 2`
	`x5:5:wait:/etc/init.d/xsession start`

Как видите, во всех случаях порядок действий примерно одинаков:

- Вначале задается уровень выполнения (runlevel).
- Затем выполняются действия начальной инициализации системы, не зависящие от заданного уровня. В разных дистрибутивах для этого вызываются разные скрипты:
  - в Fedora Core - /etc/rc.d/rc.sysinit
  - в Debian - /etc/init.d/rcS
- Затем выполняется скрипт перехода на заданный уровень выполнения. При этом, если в Fedora Core и Gentoo для каждого уровня выполнения задана своя строка

При использовании стиля System V используется один и тот же скрипт перехода на заданный уровень, а то, что он делает, определяется содержимым каталога /etc/rc.d/rcN.d. Этот каталог содержит перечень ссылок на скрипты запуска тех системных сервисов, которые должны работать на уровне N. Сами скрипты размещаются в каталоге /etc/init.d или /etc/rc.d/init.d.

В отличие от стиля System V в BSD-стиле каждому уровню загрузки соответствует свой сценарий. И сначала всегда происходит переход на уровень S (однопользовательский), а затем уже переход на заданный уровень. Поскольку стиль System V взят за основу при создании стандарта LSB (Linux Standart Base), дистрибутивы, ранее использовавшие стиль BSD. Можно по-разному оценивать, какой из вариантов организации инициализационных

скритпов лучше или хуже. Тут скорее всего тут просто дело привычки. Но сошлюсь на мнение авторов одной из книг которые высказали следующие мнения о скриптах трех наиболее распространенных дистрибутивов:

- "В сценариях Red Hat непросто разобраться."
- "Сценарии запуска системы - это та область, в которой SUSE превосходит другие дистрибутивы Linux. Эти сценарии четко организованы, надежны и хорошо документированы."
- "Сценарии Debian ненадежны, недокументированы и невероятно противоречивы. Печально, но отсутствие стандартных правил организации сценариев запуска привело в данном случае к хаосу."

Как бы то ни было, разобраться в сценариях запуска возможно, что мы и попытаемся сделать в следующем разделе. А пока рассмотрим в целом, какие задачи выполняет процесс **init**.

Действия, выполняемые процессом init

Для примера возьмем файл inittab из дистрибутива ASP Linux 11, то есть будет рассматриваться процедура начальной загрузки в стиле System V.
Листинг . Файл /etc/inittab системы

```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

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

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"


# Run gettys in standard runlevels
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

# Run xdm in runlevel 5
x:5:once:/etc/X11/prefdm -nodaemon
```

Обработка файла /etc/inittab процессом `init` начинается в однопользовательском режиме (уровень 1), в котором единственным пользователем является root, работающий с консоли. Первым делом `init` находит строку, которая определяет, какой уровень выполнения запускается по умолчанию:

```
id:3:initdefault:
```

Это и будет тот уровень, в котором запустится и будет работать система после загрузки, поэтому естественно, что нельзя указывать в строке `initdefault` уровни 0 и 6.

Далее `init` выполняет команды, указанные в строке с ключевым словом `sysinit`. В нашем примере здесь выполняется скрипт rc.sysinit из каталога /etc/rc.d:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

После этого процесс `init` просматривает файл /etc/inittab и выполняет скрипты, соответствующие однопользовательскому уровню (1 во втором поле строки):

```
l1:1:wait:/etc/rc.d/rc 1
```

всем уровням (строки с пустым вторым полем):

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
```

и уровню, заданному по умолчанию:

```
l3:3:wait:/etc/rc.d/rc 3
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
```

Как видим, вначале вызывается скрипт rc из каталога /etc/rc.d. Какой бы уровень выполнения не был задан, вызывается один и тот скрипт, только в зависимости от уровня выполнения ему передается соответствующее значение параметра вызова, так что, например, для 3-го уровня вызов скрипта осуществляется с параметром 3.

Следующая важная функция, которую выполняет этот процесс (на уровнях со 2 по 5) — запуск шести виртуальных консолей (процессов getty), чтобы предоставить пользователям возможность регистрироваться в системе с терминалов. В нашем примере запускается только 3 виртуальных консоли, поскольку еще 3 строки в файле /etc/inittab закомментированы (я сделал это в целях экономии системных ресурсов, мне вполне хватает трех консолей).

Функции, выполняемые скриптами rc.sysinit и rc, а также процесс запуска виртуальных консолей мы подробнее рассмотрим ниже, а сейчас вернемся к краткому обзору действий процесса `init`.

После завершения загрузки `init` продолжает работать в фоновом режиме, отслеживая изменения в состоянии системы. Например, если будет подана команда `telinit`, позволяющая изменить уровень выполнения, процесс `init` обеспечит выполнение команд, заданных для нового уровня файлом /etc/inittab. Этот файл прочитывается заново и в случае поступления сигнала HUP; эта особенность избавляет от необходимости перезагружать систему для того, чтобы сделать изменения в начальной конфигурации.

Таким образом, процесс начальной загрузки `init` постоянно находится в оперативной памяти и при получении соответствующих сигналов повторно выполняет цикл чтения из файла /etc/inittab инструкций о том, что нужно делать, причем этот набор инструкций различен для разных уровней выполнения.

Когда суперпользователь останавливает систему (командой `shutdown`), именно `init` завершает все другие исполняющиеся процессы, размонтирует все файловые системы и останавливает процессор.

К вопросу о том, что делает процесс init после завершения процесса загрузки системы, нам, видимо, еще придется вернуться, а пока обратимся к разбору инициализационных скриптов.

Замечание

Если вы некорректно модифицируете файл /etc/inittab, **система может перестать загружаться.** Так что перед внесением каких-либо изменений в этот файл по меньшей мере запаситесь загрузочной флешкой и сохраните копию исходного файла на случай фатальных ошибок.