Впервые опубликовано на developerWorks 31.08.2005

Учебник для экзамена LPI 201: Запуск системы

Администрирование, средний уровень (LPIC-2) тема 202

Запуск системы и процессы загрузки

Что происходит, когда включается компьютер с ОС Linux?

Давайте разделим процесс загрузки ОС Linux на девять этапов, которые имеют место практически для любой конфигурации ОС Linux:

1. Первый этап загрузки -- это считывание BOIS'ом компьютера или другими программно-аппаратными средствами MBR жесткого диска или другого загрузочного устройства (например, компакт-диска, гибкого диска или сетевого загрузочного устройства, etc.).

2. Начинается работа загрузчика. Linux на архитектуре x86 обычно использует LILO или GRUB. Некоторые старые системы могут использовать `loadlin`, чтобы загрузиться через вспомогательный DOS-раздел. В системах Power PC® это может быть `BootX` или `yaboot`. Вообще, загрузчик -- это простая программа, которая, тем не менее, знает, где искать ядро Linux, может выбрать, какую загружать из нескольких версий ядра или даже выбрать другую операционную систему на той же машине.

3. Загрузка ядра Linux.

4. Монтируется корневая файловая система. В некоторых случаях, временно монтируется начальная корневая файловая система из содержимого, например, RAM-диска, инициализируемого загрузчиком, чтобы дать возможность загружаться специальным драйверам и модулям, которые могут понадобиться для работы настоящей корневой файловой системы.

   Теперь у нас есть корневая файловая система, и мы можем начать собственно инициализацию.

5. Запускается процесс `init`, прародитель всех остальных процессов в ОС Linux.

6. Считывается содержание файла `/etc/inittab`, чтобы определиться с дальнейшим ходом загрузки. Особенно важно что прописано в файле `/etc/inittab` в строке, определяющей уровень запуска системы (и, следовательно, последующие этапы загрузки).

   Действительно, все происходящее после этого момента полностью

определяется содержимым файла `/etc/inittab`. Фактически, скрипты и другие инструменты, которые работают, подчиняются соответствующим настройкам, но, в принципе, вы могли бы полностью изменить `/etc/inittab`, чтобы управлять работой различных инструментов по вашему желанию.
Одна из установок в файле `/etc/inittab` особенно важна. Это строка, похожая на:

```
id:5:initdefault:
```

Обычно она находится ближе к началу файла и устанавливает уровень запуска системы. Уровень запуска определяет, какие действия будут предприняты в оставшихся предписаниях файла `/etc/inittab`.

Что происходит, когда сценарий `/etc/inittab` отработан? И особенно, какие именно файлы и директории принимают участие в процессе?

7. Инициализация, независимая от уровня запуска. Существуют ряд действий, которые будут выполняться независимо от установленного уровня запуска. Эти шаги обозначены в `/etc/inittab` строками, похожими на:

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

В некоторых системах Linux (в основном в системах на основе Debian), вы скорее увидите строчки, более похожие на следующие:

```
si::sysinit:/etc/init.d/rcS
```

В последнем случае файл `/etc/init.d/rcS` -- это просто скрипт, который по очереди запускает скрипты `/etc/rcS.d/[Ss]??*`. С другой стороны, если в вашей системе используется `/etc/rc.d/rc.sysinit`, для выполнения инициализации достаточно одного длинного скрипта, содержащегося в этом файле.

8. Инициализация, зависимая от уровня запуска. Фактически, вы можете определить столько действий, связанных с уровнем запуска, сколько захотите, и при этом каждое действие может относиться к одному или нескольким уровням запуска. Как правило, `/etc/inittab` будет содержать строки типа:

```
l0:0:wait:/etc/rc.d/rc 0
# ...
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

В свою очередь, скрипт `/etc/rc.d/rc` будет управлять всеми файлами, названными `/etc/rc$1.d/[KkSs]??*`. В следующем примере можно увидеть, что в данной системе, стартующей с уровнем запуска 5, будут выполняться (по порядку):

```
/etc/rc5.d/K15postgresql
/etc/rc5.d/S01switchprofile
/etc/rc5.d/S05harddrake
```

```
...
/etc/rc5.d/S55sshd
...
/etc/rc5.d/S99linuxconf
/etc/rc5.d/S99local
```

Файлы, начинающиеся с "K" или "k" являются *убивающими (kill) скриптами*, они завершают процессы или упорядочивают их действия (последствия). Файлы, которые начинаются с "S" или "s" -- *это запускающие (startup) скрипты*, они начинают новые процессы или подготавливают систему к работе с этим уровнем запуска. Большинство из них являются скриптами shell, и большая часть их будет ссылками (часто на `/etc/init.d/`).

В то время когда система Linux стартует с определенным уровнем запуска, вы хотите зарегистрироваться в системе как пользователь. Чтобы авторизация прошла успешно, используется программа `getty`. Множество разновидностей программ на основе `getty` используется создателями дистрибутивов, типа `agetty, mgetty,` и `mingetty`. Но все они делают примерно и то же.

9. Войдите в систему в приглашении. Уже знакомый нам `/etc/inittab` обычно запускает `getty` на одном или нескольких виртуальных экранах и делает это для нескольких уровней запуска. Уровни определены в строках типа:

```
# 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
```

Цифра в начале показывает, в каком виртуальном терминале будет работать программа `getty`; следующие несколько цифр -- это те уровни запуска, при которых это случится (например, запуск `mingetty` при каждом из уровней 2, 3, 4 и 5).

Следующие шаги инициируют запуск дополнительных служб, вход в графическое окружение, восстановление настроек пользовательского интерфейса или других более персонифицированных деталей, которые находятся вне рамок этого учебного пособия.

Понятие уровня запуска

Понятие уровня запуска несколько произвольно, по крайней мере, оно не прописано в ядро Linux. Действительные уровни запуска сопоставлены набору номеров, чтобы можно было установить (или изменить имеющийся) по умолчанию уровень запуска выбором номера от 0 до 6. В соответствии с соглашением, следующий смысл присваивается каждому номеру уровня запуска:

Листинг 1. Уровни запуска

```
# Default runlevel. The runlevels used by Mandrake Linux are:
#   0 - Halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you don't have
```

```
networking)
#    3 - Full multiuser mode
#    4 - Unused
#    5 - X11
#    6 - Reboot (Do NOT set initdefault to this)
```
Это соглашение, как можно видеть, используется в дистрибутиве Mandrake Linux, но большинство дистрибутивов используют то же самое соглашение. Может так оказаться, что текстовые или встроенные дистрибутивы не используют некоторые из уровней запуска, но все равно ими будут зарезервированы эти же номера.

Конфигурационные строки в `/etc/inittab`

Вы видели множество строчек из файла `/etc/inittab` в примерах, но что же конкретно они означают? Каждая строка имеет формат:

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

Поле `id` это короткое сокращение, обозначающее конфигурационную строчку. (1 - 4 буквы в свежих версиях `init`; 1 - 2 в более старых). Поле `runlevels` уже обсуждалось. Следующее поле `action` обозначает действие, предпринимаемое строкой. Некоторые действия могут быть "специальными," такие как:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Эта строка устанавливает действие для последовательности клавиш Ctrl-Alt-Delete (независимо от уровня запуска). Но большинство действий просто запускает соответствующие процессы. Частичный список действий включает:

- `respawn`: Процесс будет перезапущен всякий раз, когда завершится (как в случае с `getty`).
- `wait`:Процесс будет начат однажды, когда будет введен указанный уровень запуска, и init будет ждать его завершения.
- `once`: Процесс будет выполнен однажды, когда будет введен указанный уровень запуска.
- `boot`: Процесс будет выполнен во время загрузки системы (но после `sysinit`). Уровень запуска не имеет значения.

Настройка запуска системы и процесса загрузки

Что такое загрузчик?

Несколько лет назад для загрузки Linux на x86 системах в основном использовалась программа, названная LILO. Название LILO -- сокращение от "LInux LOader." Сейчас более популярна программа, названная GRUB (GRand Unified Bootloader). На системах, отличных от x86, используются другие загрузчики, но все они сконфигурированы аналогичным для LILO и GRUB способом.

Хотя существуют различия в их конфигурационном синтаксисе, и LILO и GRUB выполняют в значительной степени одну и ту же задачу. По существу, каждый из них предоставляет выбор операционной системы (включая, возможно, несколько ядер

Linux) и загружает ядро выбранной ОС в память компьютера. Обе программы позволяют вам передавать аргументы ядру Linux по ходу, и обе могут быть сконфигурированы с возможностью загрузки на том же компьютере ОС, отличных от Linux.

Либо LILO, либо GRUB (или какой-то другой загрузчик) находится на MBR (Master Boot Record) первичного жесткого диска, который автоматически загружается системным BIOS. LILO имеет ограничения на загрузку специального raw сектора жесткого диска. Загрузчик GRUB более изощрен и распознает разные файловые системы, например, такие как ext2/3, ReiserFS, VFAT или UFS. Это означает, что GRUB не нужно перезаписывать MBR каждый раз, как только изменился конфигурационный файл (как это делает LILO).

Настройка загрузчика LILO

Настройка загрузчика LILO производится при помощи содержимого файла /etc/lilo.conf. Для более детального изучения параметров настройки LILO, прочитайте страницы помощи man для lilo.conf. Общий характер поведения определяют несколько начальных параметров. Например, вы наверняка увидите boot=/dev/hda или нечто подобное. Эта команда устанавливает загрузчик на MBR первичного жесткого диска IDE. Вы можете также установить LILO внутрь конкретного раздела, обычно это нужно, когда вы используете другой основной загрузчик. Например, boot=/dev/sda3 устанавливает LILO на третий раздел первого SCSI диска. Другие параметры определяют внешний вид и время ожидания LILO.

Запомните, что после того, как вы внесете исправления в файл /etc/lilo.conf, вам необходимо запустить LILO для фактической установки нового загрузочного сектора, который используется во время загрузки. Можно легко забыть установить новые параметры, но загрузчик сам по себе не сможет прочесть новую конфигурацию, за исключением того случая, когда записаны фактические адреса секторов (которые LILO распознает в процессе работы).

Если используется LILO, особенное значение имеют строки типа `image=` и, может быть, `other=`, если имеется выбор между ОС Linux и другими операционными системами. Пример /etc/lilo.conf может содержать:

Листинг 2. Пример конфигурации LILO

```
image=/boot/bzImage-2.7.4
label="experimental"
image=/boot/vmlinuz
label="linux"
initrd=/boot/initrd.img
append="devfs=mount acpi=off quiet"
vga=788
read-only
other=/dev/hda3
label=dos
```

Такая конфигурация позволяет вам выбирать либо ядро версии 2.7.4, которое находится в стадии разработки, либо стабильное ядро (далее объявлено, что

использовать как стартовый RAM-диск (initrd) в процессе загрузки). Вы можете также выбрать DOS, которая находится на третьем разделе первичного IDE диска.

Настройка загрузчика GRUB

Бесспорным преимуществом GRUB является то, что его не надо переустанавливать всякий раз, после того как вы изменили параметры загрузки. Конечно, в первый раз GRUB все же нужно установить, обычно это делается командой типа `grub-install /dev/hda`. Как правило, в процессе инсталляции дистрибутив делает это для вас сам, так что вы, может статься, так ни разу сами этого и не сделаете.

Теперь, так как GRUB знает, как читать разные файловые системы, вы можете легко внести изменения в файл /boot/grub/menu.lst, чтобы изменить параметры следующей загрузки. Взгляните на пример конфигурации GRUB:

Листинг 3. Пример конфигурации GRUB

```
timeout 5
color black/yellow yellow/black
default 0
password secretword

title linux
kernel (hd0,1)/boot/vmlinuz root=/dev/hda2 quiet
vga=788 acpi=off
initrd (hd0,1)/boot/initrd.img

title experimental
kernel (hd0,1)/boot/bzImage-2.7.4 root=/dev/hda2 quiet

title dos
root (hd0,4)
makeactive
chainloader +1
```

Изменения параметров во время работы загрузчика (LILO)

И LILO и GRUB позволяют передать специальные параметры выбранному вами ядру. Если вы используете LILO, вы можете передать параметры приглашению boot добавлением их к выбранному вами ядру. Например, для обычных параметров загрузки вы можете ввести:

```
LILO: linux ether=9,0x300,0xd0000 root=/dev/ha2 vga=791 acpi=on
```

Эта строчка передает специальные параметры модулю Ethernet, указывает корневой раздел, выбирает режим видео, и т.д. Конечно, не все это удобно, так как вы должны знать точно подходящие значения этих параметров и уметь правильно ввести их.

Особенное значение имеет параметр, который изменяет уровень запуска системы загрузчиком. Например, в целях восстановления системы, вы хотите загрузить систему в однопользовательском режиме. Осуществляется это следующим образом:

```
LILO: experimental single
```

или:

```
LILO: linux 1
```

Другой специальный параметр -- аргумент `init=`, который позволяет вам использовать программы, отличные от init в качестве первичного процесса. Параметры для режима аварийной ситуации могут быть следующими: `init=/bin/sh`, что, по крайней мере, позволит вам иметь в своем распоряжении командную строку (Linux shell), если init совсем вышел из строя.

Изменения параметров во время работы загрузчика (GRUB)

С загрузчиком GRUB вы имеете еще большую гибкость. Фактически, GRUB представляет из себя оболочку командной строки и предоставляет пользователю базовую функциональность shell. GRUB дает возможность не только изменить базовую конфигурацию загрузчика, но даже читать файловые системы. Для настройки параметров загрузки, нажмите "e" в командной строке GRUB, после этого добавьте параметры (например, номер уровня запуска или ключевое слово "single" как в LILO). Все другие аргументы в приглашении загрузки, которые вы могли бы ввести, используя LILO, могут быть использованы в командной строке GRUB.

Для понимания своих возможностей вы можете открыть командную строку GRUB. Например, предположите, что вам кажется, что ваш файл /etc/inittab плохо сконфигурирован, и вы хотите исследовать это перед загрузкой. Вы могли бы ввести:

```
grub> cat (hd0,2)/etc/inittab
```

Это позволило бы заранее просмотреть ваш инициализационный файл, без запуска операционной системы. Если бы там обнаружилась ошибка, то можно было бы загрузиться в однопользовательский режим и исправить ее.

Настройка, осуществляемая после загрузчика

Как только вы осознаете шаги в загрузке Linux после загрузки ядра (другими словами, процесс `init` и все, что он вызывает), вы также осознаете, как их отредактировать. В основном, вся настройка осуществляется редактированием файла /etc/inittab и различных скриптов в каталоге /etc/rc?.d/.

Например, недавно мне понадобилось настроить видео BIOS на ноутбуке с Linux, базирующемся на Debian, использующем разработки третьих фирм. Если он не был запущен до того, как запустятся X11, мой драйвер XOrg не установил бы правильные режимы видео. Как только я выяснил, в чем была проблема, решение было столь же просто, как создание скрипта /etc/rcS.d/S56-resolution.sh. Другими словами, я запускал дополнительный скрипт при каждой загрузке системы.

Замечу, что я удостоверился, что этот скрипт исполняется раньше, чем /etc/rcS.d/S70xorg-common вследствии простого соглашения, что скрипты запускаются в алфавитном порядке (если бы я хотел, чтобы мой скрипт выполнялся позже, я, возможно, назвал бы его S98-resolution.sh вместо /etc/rcS.d/S56-resolution.sh). Может быть, я поместил бы этот скрипт только в каталог /etc/rc5.d/, чтобы он запускался, когда выполняются X11, но я могу вручную запустить startx из-

под другого уровня запуска.

Все настройки в процессе инициализации открыты для редактирования, прямо в файловой системе; почти все можно исправить с помощью текстовых скриптов.

Восстановление файловой системы

О восстановлении

Самое замечательное свойство Linux, которое рассматривается в перспективе обслуживания системы, это то, что все является файлом. Конечно, время от времени возникает вопрос , в каком файле что живет. Но зато, как правило, восстановление Linux означает применение основных утилит файловой системы, таких как `cp, mv, rm` и текстовый редактор типа `vi`. Для автоматизации этих действий полезны такие инструменты как `grep, awk` и `bash`; или на более высоком уровне, `perl` или `python`. Но в данном учебном пособии мы не ставим целью изучать обращение с файлами.

Предположим, что вы знаете, как редактировать файлы и управлять ими, вот только в порушенной системе файлы, которых коснулось повреждение, возможно останутся вообще непригодными для использования.

Исправление испорченной файловой системы с помощью fsck

Ваш лучший друг в восстановлении поврежденной файловой системы `fsck`. Следующий раздел (тема 203) содержит больше информации, таким образом, здесь мы лишь представим этот инструмент в общих чертах.

Команда `fsck` является фактически только началом команды для большого количества других инструментов `fsck.* -- fsck.ext2`, `fsck.ext3`, или `fsck.reiser`. Вы можете определить тип явно, используя опцию `-t`, но `fsck` предпримет усилие понять самостоятельно. Прочитайте страницу помощи `man` для `fsck` или `fsck.*` для получения более подробной информации. Основное, что вам нужно знать, что при использовании аргумента -a программа будет пытаться исправить все найденные ошибки.

Вы можете проверить неподмонтированную файловую систему, упоминая местонахождение устройства, на котором она находится. Например, введите `fsck /dev/hda8`, чтобы проверить неиспользующийся раздел. Вы можете также проверить корневую файловую систему, набрав `fsck /home`, но как правило, делают это, только если файловая система уже смонтирована как "только для чтения", а не для "чтения-записи".

Монтирование и отмонтирование с помощью mount and umount

Одно из основных преимуществ систем Linux состоит в гибкости пользовательского контроля, при монтировании и отмонтировании файловых системам. В отличие от Windows и некоторых других операционных систем, местоположения разделов не автоматически закреплены ядром Linux, а присоединены к иерархии корневой файловой системы командой `mount`. Кроме того, различные типы файловых систем (даже на различных устройствах) могут быть смонтированы в рамках той же самой

иерархии. Вы можете отмонтировать конкретный раздел командой `umount`, назначать любую точку монтирования (например, /home) или адрес устройства (например, /dev/hda7).

Когда производится восстановление файловой системы, возможность управлять точками монтирования позволяет вам проводить анализ состояния разделов, используя fsck или другие инструменты, без риска дальнейшего повреждения уже поврежденной файловой системы. Вы можете также в обычном порядке монтировать файловую систему, используя различные параметры; самые важные из них монтируют файловую систему для использования только в режиме чтения с помощью одного из синонимов `-r` или `-o ro`.

В качестве примера, вы могли бы хотеть заменить местоположение каталога одного пользователя на каталог другого, или из-за повреждения раздела, или просто хотите расширить дисковое пространство, или переместиться на более быстрый диск. Такое изменение можно выполнить, используя:

```
# umount /home # old /dev/hda7 home dir
# mount -t xfs /dev/sda1 /home # new SCSI disk using XFS
# mount -t ext3 /dev/sda2 /tmp # also put the /tmp on SCSI
```

Монтирование при загрузке с помощью /etc/fstab

Для восстановления, модернизации системы, и специальных целей полезно иметь возможность монтировать и отмонтировать файловые системы по желанию. Но для повседневной работы, вам будет удобно, чтобы необходимый конкретный набор подмонтирований осуществлялся автоматически при каждой загрузке системы. Вы управляете точками монтирования, прописывая нужные строки конфигурации в файл /etc/fstab. Типичная конфигурация могла бы выглядеть так:

Листинг 4. Пример конфигурации в /etc/fstab

```
/dev/hda7 / ext3 defaults 1 1
none /dev/pts devpts mode=0620 0 0
/dev/hda9 /home ext3 defaults 1 2
none /mnt/cdrom supermount
dev=/dev/hdc,fs=auto,ro,--,iocharset=iso8859-1,codepage=850,umask=0 0 0
none /mnt/floppy supermount
dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,sync,codepage=850,umask=0 0 0
none /proc proc defaults 0 0
/dev/hda8 swap swap defaults 0 0
```

Более полную информацию по этому вопросу вы найдете в следующем разделе (тема 203).