Загрузка операционной системы

Загрузка - это процесс перевода системы из выключенного состояния в работающую операционную систему. Понимание этого процесса очень важно для исправления проблем с загрузкой, когда они возникают, и изменения состояний системы, называемых уровнями запуска.

Процесс загрузки проходит в четыре основных этапа, некоторые из которых изменяются администраторами, в то время как для других достаточно просто знать о происходящих действиях:

- Этап прошивки (Firmware Stage)
- Загрузчик (Bootloader)
- Стадия ядра (Kernel Stage)
- Стадия инициации (Init Stage)

Firmware Stage

Этап прошивки - это первый этап, который происходит после включения компьютера. На этом этапе у компьютера есть питание, но ему нужно запустить какое-то программное обеспечение, которое в конечном итоге полностью загрузится.

Как упоминалось ранее, большинство встроенных программ называется базовой системой ввода-вывода (BIOS) и хранится на материнской плате в энергонезависимой памяти, такой как постоянное запоминающее устройство (ПЗУ) или флэш-память. BIOS обеспечивает упрощенное представление о компьютерном оборудовании, так что работа на следующем этапе будет намного проще. Жесткий диск компьютера может иметь гигабайты пространства для хранения программ, но на данном этапе жизни компьютера он имеет дело с программами, которые должны умещаться в килобайтах пространства.

Даже в системах, которые заменили традиционный BIOS на Unified Extensible Firmware Interface (UEFI), системное встроенное ПО все еще часто называют BIOS.

BIOS имеет ряд заданий, которые необходимо выполнить на первом этапе процесса загрузки. Одним из таких заданий является выполнение самотестирования при включении питания (POST), чтобы убедиться, что аппаратное обеспечение системы функционирует должным образом. POST выполняет некоторые базовые проверки работоспособности процессора, памяти и периферийных устройств, так что очевидные ошибки, такие как отсутствующие микросхемы памяти, обнаруживаются в начале цикла загрузки.

Современные серверы имеют расширенную периферию, которая может иметь собственный BIOS. BIOS материнской платы инициализирует периферийный BIOS, когда система готовится к загрузке.

Последняя задача BIOS - найти подходящий загрузочный диск на доступных устройствах хранения и загрузить главную загрузочную запись (MBR) с этого устройства. Основная загрузочная запись - это **первый** сектор (или 512 байт) диска. Он содержит таблицу разделов и очень небольшое количество исполняемого кода, называемого загрузчиком первого этапа, целью которого является загрузка более многофункционального загрузчика второго этапа.

Стадия загрузки

Загрузчик выполнит несколько операций, но основная задача - загрузить ядро Linux в память и выполнить его. Как только это произошло, ядро начинает загрузку системы.

Наиболее распространенные загрузчики, используемые на машинах, это Linux Loader (LILO) и Grand Unified Bootloader (GRUB). И LILO, и GRUB являются загрузчиками, которые могут загружать Linux из системы, использующей ПЗУ BIOS. Последняя версия GRUB поддерживает загрузку Linux из системы, использующей UEFI, в то время как загрузчик Efi Linux (ELILO) может использоваться вместо LILO в системах, которые используют UEFI вместо традиционного BIOS. Системы UEFI дают стадии

прошивки намного больше памяти и возможностей, так что она может обрабатывать гораздо большие и более сложные аппаратные средства.

За пределами IBM PC-совместимых архитектур используются дополнительные загрузчики. Для систем Linux, загружаемых на оборудовании Sparc, есть Sparc Improved bootLOader (SILO), а для оборудования PowerPC - еще один BOOTloader (YABOOT).

Также возможно загрузиться с сети через Preboot Execution Environment (PXE). В системе PXE совместимая материнская плата и сетевая карта содержат достаточно информации, чтобы получить адрес из сети и использовать Trivial File Transfer Protocol (TFTP) для загрузки специального загрузчика с сервера. Это чаще всего используется для быстрого запуска многих идентичных машин. Он мощный, но сложный и выходит за рамки LPIC1.

Поскольку загрузчик - это просто некое программное обеспечение, которое запускает ядро, можно загружать несколько операционных систем в разное время с одного компьютера в процессе, называемом двойной загрузкой. Ядро, которое пытается запустить загрузчик, может быть ядром Linux, может быть образом Microsoft Windows или загрузочным компакт-диском.

Загрузчик также может передавать параметры ядру, например загружаться в режиме обслуживания или включать или отключать определенное оборудование. Это делается путем манипулирования конфигурацией загрузчика. GRUB предоставляет достаточно мощный интерфейс командной строки, который позволяет администратору вносить изменения в ядро до его загрузки, не требуя записи конфигурации на диск.

Затем загрузчик загружает ядро с диска в память и передает управление. Теперь система работает под управлением Linux и может завершить загрузку.

Стадия ядра

Теперь, когда загрузчик загрузил ядро в память, для загрузки программ нужно проделать большую работу. Ядро должно инициализировать любые аппаратные драйверы и смонтировать корневую файловую систему для следующего этапа. Эти две задачи на самом деле довольно сложны, потому что возможности, предоставляемые BIOS для доступа к оборудованию, весьма ограничены. Ядро должно загружать систему в несколько этапов.

Само ядро очень похоже на обычный исполняемый файл, за исключением того, что оно должно быть самодостаточным - разделяемые библиотеки не доступны на данном этапе процесса загрузки, поэтому само ядро статически связано. Это ядро обычно находится в разделе **/boot**, который на большинстве аппаратных средств находится в отдельном разделе, который хранится в начале жесткого диска. Это расположение важно для некоторых комбинаций BIOS и загрузчика, которые могут получить доступ только к первым 1024 цилиндрам диска.

Поскольку размер ядра со временем увеличивался, разработчики обнаружили, что лучше сжать ядро, чтобы оно соответствовало ограничениям BIOS. Поэтому исполняемый файл, содержащий ядро, будет распаковываться сам по себе при загрузке, что приводит к имени **zImage** для ядра (буква **z** связана с библиотекой сжатия Unix под названием **zlib**.)

Ядро выросло еще больше, и стало проблемой загрузить все ядро в последовательный блок памяти. Был создан большой формат ядра **zImage**, **bzimage**, позволяющий загружать ядро в несколько блоков памяти.

Ядро Linux должно смонтировать корневую файловую систему /, чтобы перейти к следующему шагу и сделать систему полезной. Однако возможно, что корневая файловая система существует на устройстве, которое ядро не знает, как поддерживать. Решением этой проблемы является начальный RAM-диск **initrd**. Драйверы ядра, необходимые для продолжения загрузки, объединяются в файловую систему, которая хранится рядом с самим ядром в **/boot**. Ядро загружается, монтирует **initrd**, загружает драйверы внутри, а затем перемонтирует настоящую корневую файловую систему, используя новые драйверы. Это позволяет легко добавлять драйверы в ядро, и ядро может монтировать

корневую файловую систему практически на любом оборудовании хранения, даже если оно находится в сети.

Когда ядро загружается, оно может инициализировать оборудование и сделать обнаруженные устройства доступными для остальной части операционной системы.

Последняя задача ядра - запустить первый процесс в системе. Поскольку это первый процесс, он обычно имеет идентификатор процесса (PID), равный 1; имя этого процесса - **init.**

```
ps -ejH
```

```
  PID  PGID   SID TTY          TIME CMD
    1     1     1 ?        00:00:00 init
   32    32    32 ?        00:00:00   rsyslogd
   37    37    37 ?        00:00:00   cron
   39    39    39 ?        00:00:00   sshd
   56    56    56 ?        00:00:00   named
   69     1     1 ?        00:00:00   login
   79    79     1 ?        00:00:00     bash
  676   676     1 ?        00:00:00       ps
```

Первый процесс отвечает за большую часть работы системы, начиная с этого момента. Обычно этот процесс будет **/sbin/init**, хотя есть и другие параметры, такие как **systemd**. В качестве альтернативы на встроенном оборудовании процесс **init** может быть оболочкой или специализированным демоном. В любом случае, этот первый процесс запустит демоны, которые будет использовать остальная часть системы. Этот процесс будет родительским процессом для любого процесса, в котором иначе отсутствует родительский процесс. Этот процесс сохраняется в течение всей жизни системы.

Этап инициации

Этап **init** завершает загрузку системы; первый процесс операционной системы запускается, и этот процесс отвечает за запуск всех других системных процессов.

Первый загружаемый процесс имеет две важные обязанности: во-первых, продолжить процесс загрузки для запуска служб, отображения экранов входа в систему и прослушивания консолей. Второе - это некоторые основные процессы управления. Любой процесс, который теряет родителя, принимается **init.**

До недавнего времени этот процесс следовал дизайну, который был создан с выпуском System V Unix, который иногда называют SysVinit. Фактический процесс, который выполняется, является процессом init. Недавно появились другие программы, которые могут конкурировать и заменить традиционный процесс инициализации: **Upstart** и **Systemd.**

Если система использует традиционную программу инициализации, то файл **/etc/inittab** используется для определения того, какие сценарии будут выполняться для запуска служб, которые будут доступны в системе. Файл inittab указывает на другие скрипты, которые выполняют эту работу, обычно хранящиеся в /etc/init.d.

Если традиционный init был заменен на Upstart, сценарии в каталоге /etc/init используются для завершения инициализации системы.

Если традиционный init был заменен на Systemd, то файлы в каталоге /etc/systemd используются для запуска и запуска системы.

Даже если ваша система использует Systemd или Upstart в качестве замены традиционного процесса инициализации, обе замены используют исполняемый файл с именем init с путем **/sbin/init**. Это необходимо для обеспечения совместимости со многими устаревшими процессами. Таким образом, хотя некоторые действия Systemd и Upstart будут отличаться, у них есть некоторые функции, которые похожи на традиционный процесс инициализации.

Команда dmesg

Команда **dmesg** может быть выполнена после загрузки системы, чтобы увидеть сообщения, сгенерированные ядром во время загрузки. Это полезно, когда система не загружается правильно; отображаемые сообщения могут помочь администратору устранить неполадки в процессе загрузки.

Сообщения ядра хранятся в кольцевом буфере ограниченного размера, поэтому сообщения, которые генерируются во время загрузки, могут быть перезаписаны позже, когда буфер заполнится. Возможно, что некоторые сообщения ядра, сгенерированные во время загрузки, могут быть сохранены в файле **/var/log/dmesg**. Каждый раз, когда система загружается, файл **/var/log/dmesg** перезаписывается сообщениями, сгенерированными в процессе загрузки.

Обычно команда **dmesg** выполняется при подключении нового устройства к системе. Это позволяет администратору видеть, как ядро взаимодействует с новым устройством, и, как правило, видеть, какое имя пути было назначено новому устройству.

Менее распространенное использование команды **dmesg** - это изменение уровня сообщений, которые ядро будет печатать в системной консоли. Печатание в терминале и одновременная печать сообщений ядра могут стать раздражающими, команда **dmesg -n 1** может использоваться для отключения печати всех сообщений ядра, кроме самых критических.

Файл /var/log/messages

Сообщения ядра и другие системные сообщения обычно хранятся в файле **/var/log/messages**. Этот файл, который считается основным файлом журнала, в некоторых дистрибутивах альтернативно называется **/var/log/syslog.**

Традиционно основной файл журнала обновляется новыми записями журнала с помощью комбинации демонов **syslogd** и **klogd**. Замены для этих демонов включают в себя демоны **rsyslogd** и **syslog-ng.**

Основной файл системного журнала может быть полезен для анализа того, почему некоторые службы могут запускаться неправильно. Также может быть полезно определить, почему некоторые устройства могут не работать; ядро помещает записи журнала в этот файл журнала, если оно может обнаружить устройство и при загрузке соответствующих модулей ядра или драйверов для поддержки новых устройств.

Несмотря на то, что файл **/var/log/messages** считается основным файлом журнала, в папке **/var/log** есть другие файлы журнала, которые, возможно, потребуется изучить при попытке устранения неполадок, связанных с системным обслуживанием. Например, демон веб-сервера apache httpd управляет своими собственными файлами журналов в другом месте, например, **/var/log/httpd/error_log.** Традиционно все файлы журналов хранятся в каталоге **/var/log.**