Конфигурация оборудования

Понимание того, как просматривать и настраивать аппаратное обеспечение компьютера, является критически важным навыком для администратора Linux. Даже если для некоторого оборудования может не потребоваться настройки, администратор может использовать сведения при попытке диагностировать какие-либо проблемы.

Основное оборудование

Компоненты, которые необходимы для работы компьютера, считаются основными аппаратными средствами. Каждая система должна включать центральный процессор, оперативную память и какую-то прошивку. Другое аппаратное обеспечение, необходимое для полной системы, включает в себя некоторые типы устройств постоянного хранения, такие как механический жесткий диск или твердотельный накопитель (SSD), устройство оптического привода, которое поддерживает DVD и/или компакт-диски, устройство отображения или монитор, клавиатура и мышь. Большинство систем также включают одно или несколько сетевых устройств, звуковое устройство и порты для подключения внешних устройств.

Центральное процессорное устройство

Центральный процессор (CPU) - это мозг компьютера, где обрабатываются инструкции для выполнения вычислений или манипулирования данными. Существует множество типов процессоров, которые могут работать с Linux, но наиболее распространенными являются 32-битный тип x86 и 64-битный тип x86_64. Оба этих типа процессоров обратно совместимы с процессором, использованным в первом персональном компьютере IBM (ПК), процессоре Intel 8088.

Есть несколько способов собрать информацию о типе процессора в системе. Настройки прошивки должны отображать эту информацию. Файл **/proc/cpuinfo** будет содержать чрезвычайно подробную информацию, включая название модели, скорость в МГц и конкретные функции, доступные в списке флагов. Для получения менее подробной информации о процессорах в вашей системе выполните команду **uname -m** или команду **lscpu**.

```
root@L-FW:~# uname -m
x86_64
root@L-FW:~# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:   0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
```

Оперативная память

Оперативное запоминающее устройство (ОЗУ) системы используется для временного хранения данных и инструкций для операционной системы и программ, которые выполняются. Когда IBM PC дебютировал в 1981 году, он смог получить доступ к 1 МБ памяти, из которых 640 КБ - это ОЗУ, а 384 КБ - для системного ПЗУ.

Максимальный объем памяти, который можно использовать с 32-разрядным процессором, составляет 4 ГиБ, тогда как 64-разрядный процессор теоретически может использовать 16 EiB памяти. В действительности, многие 32-битные системы могут

фактически ограничиваться использованием 3 ГБ памяти, а аппаратное обеспечение еще не может использовать максимальный объем памяти в 64-битной системе.

С практической точки зрения, если система в настоящее время не имеет по крайней мере 1 ГБ ОЗУ, то она не сможет использовать графический интерфейс пользователя (GUI) с Linux. При использовании только интерфейса командной строки (CLI) требования к памяти намного меньше.

Если в системе недостаточно оперативной памяти, она будет использовать виртуальную память (в Linux это называется пространством подкачки **swap**). Пространство подкачки (**swap**) - это пространство жесткого диска, которое временно используется для хранения данных, превышающих объем доступной оперативной памяти. Когда система начинает исчерпывать память, она «выгружает» данные, которые наименее востребованы в тот момент, так что пространство ОЗУ может «высвободиться» для чего-то, что в данный момент востребовано (обычно новый процесс). Если система постоянно использует пространство подкачки, она будет работать плохо по сравнению с системой, которая этого не делает; увеличение производительности может быть получено, если в систему будет добавлено больше оперативной памяти.

Объем оперативной памяти в системе можно посмотреть в настройках прошивки. Чтобы получить подробную информацию о том, сколько памяти имеет система и как она используется, просмотрите файл **/proc/meminfo**. Для быстрого ознакомления с оперативной памятью и пространством подкачки выполните команду **free**:

```
root@L-FW:~# free
              total        used        free      shared  buff/cache   available
Mem:         504320       60172      272120        2956      172028      428304
Swap:       2095100         124     2094976
```

Вывод выше показывает, что эта система содержит около 500 МБ оперативной памяти, почти все она не используется; много свободной памяти.

Несмотря на то, что свободного ОЗУ так мало, обратите внимание, что пространство подкачки около 2000 МБ вообще не использовалось; хотя, если в этой системе возникла большая потребность в памяти, это пространство подкачки доступно. Еще одна важная информация из этого вывода - то, что настроенное пространство подкачки примерно в четыре раза больше объема оперативной памяти. Для систем с небольшим объемом оперативной памяти обычно пространство подкачки должно быть в два раза больше объема оперативной памяти.

Прошивка

Прошивка - это программное обеспечение, которое было записано в энергонезависимую память, такую как постоянная память (ПЗУ) или флэш-память. Существует несколько типов прошивок, которые могут присутствовать в компьютерной системе. На каждом устройстве, которое предоставляет услуги системе (например, сетевая карта или графический дисплей), имеется микросхема ПЗУ, которая содержит прошивку для устройства.

Самая важная прошивка содержит код, который позволяет интегрированным компонентам системы работать вместе. Прошивка тестирует компоненты при запуске, идентифицирует и инициализирует эти компоненты и пытается найти загрузчик для загрузки операционной системы.

Первоначально эта прошивка была известна как системное ПЗУ, или ПЗУ Basic Input Output System (BIOS). BIOS используется для предоставления основных услуг ввода и вывода перед загрузкой операционной системы, поэтому пользователь может вводить данные через клавиатуру или просматривать выходные данные на мониторе даже до запуска загрузчика или операционной системы.

Недавно производители компьютеров начали заменять традиционный BIOS на то, что называется Unified Extensible Firmware Interface (UEFI); однако функции UEFI выглядят настолько похожими на BIOS, что многие люди все еще называют

микропрограмму системы BIOS. Как системы на основе UEFI, так и системы на основе BIOS предоставляют запатентованную программу меню, которая позволяет включать или отключать интегрированные устройства. Микропрограмма, которая входит в комплект поставки, зависит от поставщика системы, поэтому, к сожалению, не существует стандартного способа запуска этой программы или стандартного пункта меню для включения или отключения устройств.

Для входа в программу, которая позволит изменять настройки прошивки, обычно требуется, чтобы назначенная клавиша была нажата сразу после включения системы. Многие системы используют функциональные клавиши, такие как F2 или F12, в то время как другие могут использовать клавишу ESC или DEL для запуска программы настройки встроенного программного обеспечения. Когда компьютер включен, на заставке обычно появляются сообщения с указанием соответствующей клавиши для нажатия. Если правильный ключ не найден на заставке, может потребоваться обратиться к документации поставщика.

Если в системе установлено встроенное ПО UEFI, загрузка операционной системы Linux может оказаться более сложной из-за функции Secure Boot. Если включена безопасная загрузка, то загрузчик должен быть криптографически подписан цифровым ключом, который распознается микропрограммой. Если загрузчик не подписан должным образом, загрузка все равно возможна путем отключения Secure Boot в настройках прошивки в пользу модуля поддержки совместимости (CSM).

Параметры встроенного ПО могут использоваться не только для включения или отключения безопасной загрузки, но также могут быть изменены параметры, которые будут влиять на то, какие устройства и в каком порядке встроенное ПО будет искать загрузочное устройство. Настройки прошивки могут даже использоваться, чтобы повлиять на использование внешних устройств, таких как клавиатуры.

Как правило, серверные системы могут быть настроены на работу «без головы» *headless*, то есть без клавиатуры, мыши или монитора. Как правило, BIOS препятствует загрузке компьютера без присутствия клавиатуры. Возможно, потребуется изменить параметры прошивки, чтобы отключить эту функцию для *headless*.

Устройства хранения

Хотя для Linux строго не требуется запоминающее устройство, большинство систем будет включать одно или несколько таких устройств. Наиболее распространенным запоминающим устройством является механический жесткий диск (или фиксированный диск). Эти типы дисков поставляются с различными интерфейсами или способами их подключения к компьютерной системе.

Есть довольно много интерфейсов запоминающих устройств, которые все еще используются сегодня:

Интерфейс небольшой компьютерной системы (SCSI) является одним из старейших и требует наличия в системе контроллера SCSI для управления одним или несколькими дисками, подключенными к нему.

Интерфейс типа Integrated Drive Electronics (IDE) или Parallel Advanced Technology Attachment (PATA) включает контроллер непосредственно на каждом диске и был очень популярен для жестких дисков в течение 1990-х годов. Этот тип до сих пор используется для некоторых оптических приводов сегодня.

На сегодняшний день наиболее распространенным интерфейсом, используемым для внутренних устройств хранения данных, является тип SATA (Serial Advanced Technology Attachment). Каждый диск SATA подключен напрямую к системной плате с помощью кабеля. Чтобы настроить основной диск SATA, подключите его с помощью кабеля к разъему системной платы, который обозначен как основной порт.

Для внешних накопителей интерфейс Universal Serial Bus (USB) является наиболее распространенным, но существуют и другие стандарты, такие как FireWire и Thunderbolt.

Устройства хранения данных используют последовательные или параллельные интерфейсы. Символы S в SCSI, SATA и USB означают слово serial. Вот почему они отображаются как **/dev/sda1**, что означает DEVice Serial Drive A раздел 1.

Параллельные накопители, с другой стороны, отображаются как **/dev/hda1**, что означает DEVice Hard Drive A раздел 1.

Команда **df -h** может использоваться, чтобы определить, какой тип диска используется на компьютере под управлением Linux.

```
root@L-FW:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            236M     0  236M   0% /dev
tmpfs            50M  3.1M   47M   7% /run
/dev/sda1       2.0G  885M  980M  48% /
tmpfs           247M     0  247M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           247M     0  247M   0% /sys/fs/cgroup
tmpfs            50M     0   50M   0% /run/user/0
```

Plug and Play

Помимо основного оборудования, есть много других компонентов, которые могут использоваться с компьютерной системой. Важно знать, требует ли компонент, чтобы питание компьютерной системы было отключено во время его подключения, или компонент может быть подключен, когда компьютерная система включена. Если питание компьютерной системы включено, и компонент требует выключения питания, подключение устройства может привести к повреждению компонента, компьютерной системы или обоих.

Устройства, которые должны быть подключены при отключенном питании, называются устройствами холодного подключения. Устройства, которые можно подключать при включенном питании, называются устройствами горячего подключения. Чтобы «подключи и работай» (Plug and Play) или подключите устройство, когда система включена и работает, устройство, интерфейс, драйвер и операционная система должны поддерживать горячее подключение этого устройства.

Как правило, USB-устройства поддерживают горячую замену.

Аппаратные ресурсы

Чтобы устройство работало правильно, ему должны быть выделены определенные ресурсы. Изначально у ПК были очень ограниченные ресурсы, которыми приходилось управлять вручную; администратор должен будет установить перемычки или переключатели на устройствах, чтобы настроить их для ресурсов, которые они будут использовать. Это было проблематично, так как ошибка могла создать ситуацию, когда один ресурс был распределен более чем одному устройству, в результате чего одно или оба устройства не работали. В некоторых случаях неправильное распределение ресурсов может даже привести к неработающей системе.

Сегодня не только доступно больше ресурсов, но они автоматически распределяются и управляются операционной системой, а не вручную администратором.

Существует четыре типа аппаратных ресурсов, которые устройства используют для связи с системой. Поскольку некоторые из этих ресурсов упоминаются как ввод и вывод, часть их имени может быть сокращена до IO. Четыре ресурса: порты ввода-вывода, память ввода-вывода, запросы прерываний и каналы прямого доступа к памяти (DMA). За исключением запросов прерывания (IRQ), эти ресурсы не могут быть разделены между устройствами.

Чтобы узнать, какое устройство или драйвер может использовать определенный ресурс, просмотрите следующие файлы в каталоге **/proc**: **dma, iomem, ioports** и файлы каталога **/proc/irq**. Имейте в виду, что администраторам редко требуется просматривать

эти данные в современных системах Linux, поскольку конфигурация устройств почти всегда прозрачна и автоматическая.

Просмотр оборудования

Чтобы просмотреть многие устройства, подключенные к вашей системе, используйте команды **lsusb** и **lspci**. Большинство устройств, которые находятся внутри компьютера, будут отображаться с помощью команды **lspci**, так как эти устройства обычно используют шину периферийного межкомпонентного соединения (PCI). Видео, звук, сетевые и дисковые контроллеры обычно находятся на шине PCI.

Для просмотра внешних устройств команда **lsusb** покажет те, которые подключены к универсальной последовательной шине (USB).

Использование параметра **-v** с командами **lsusb** или **lspci** может быть полезно, чтобы эти команды отображали более подробную информацию о подключенных устройствах. Примечание. Для получения более подробной информации команда **lspci** также поддерживает параметры **-vv** и **-vvv.**

Для устранения неполадок с устройством, которое не работает, одним из первых шагов является его конкретная идентификация. С помощью команды **lspci** используйте параметр **-nn**, чтобы показать все устройства с кодом поставщика и устройства:

```
lspci -nn
```

```
00:00.0 Host bridge [0600]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host b
ev 01)
00:01.0 PCI bridge [0604]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bri
 01)
00:07.0 ISA bridge [0601]: Intel Corporation 82371AB/EB/MB PIIX4 ISA [8086:7110]
00:07.1 IDE interface [0101]: Intel Corporation 82371AB/EB/MB PIIX4 IDE [8086:71
00:07.3 Bridge [0680]: Intel Corporation 82371AB/EB/MB PIIX4 ACPI [8086:7113] (r
00:07.7 System peripheral [0880]: VMware Virtual Machine Communication Interface
)
00:0f.0 VGA compatible controller [0300]: VMware SVGA II Adapter [15ad:0405]
00:11.0 PCI bridge [0604]: VMware PCI bridge [15ad:0790] (rev 02)
00:15.0 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.1 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.2 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.3 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.4 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.5 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
00:15.6 PCI bridge [0604]: VMware PCI Express Root Port [15ad:07a0] (rev 01)
```

Используйте vendor и код устройства, чтобы получить подробную информацию об устройстве с помощью опции -d. Например, представьте, что команда lspci -nn показывает сетевую карту с кодом устройства 15ad:07a0. Чтобы получить подробный обзор карты, выполните:

```
lspci –v –d 15ad:07a0
```

```
00:15.0 PCI bridge: VMware PCI Express Root Port (rev 01) (prog-if 00 [Normal decode])
        Flags: bus master, fast devsel, latency 0, IRQ 24
        Bus: primary=00, secondary=03, subordinate=03, sec-latency=0
        I/O behind bridge: 00004000-00004fff
        Memory behind bridge: fd500000-fd5fffff
        Prefetchable memory behind bridge: 0000000020000000-00000000201fffff
        Capabilities: [40] Subsystem: VMware PCI Express Root Port
        Capabilities: [48] Power Management version 3
        Capabilities: [50] Express Root Port (Slot+), MSI 00
        Capabilities: [8c] MSI: Enable+ Count=1/1 Maskable+ 64bit+
        Kernel driver in use: pcieport
        Kernel modules: shpchp
```

Хотя некоторые из этих выходных данных носят технический характер, такие как ресурсы памяти и порты ввода-вывода, которые использует устройство, другие выходные данные являются просто описательными, такими как имя поставщика и модель

устройства. Последняя строка вывода важна, так как она определяет имя модуля ядра, который поддерживает устройство.

Чтобы изолировать детали устройства USB, найдите код поставщика и устройства в выходных данных команды **lsusb**:

```
lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Чтобы получить изолированные данные устройства 002, выполните:

```
lsusb -v -d 0e0f:0003 | head -20
```

Обратите внимание, что вывод огромен, когда с командой **lsusb** используется опция **-v**. Как правило, вы на самом деле просто ищете тип USB-устройства, которое из приведенных выше выходных данных представляет собой виртуальную мышь.

Аппаратные подсистемы

Linux постоянно развивает подсистемы, которые она использует для управления оборудованием. Подсистема - это компонент ядра (известный как модуль ядра), который помогает ядру управлять аппаратными устройствами. Различные подсистемы включают **udev, sysfs, hald и dbus.**

Подсистема udev поддерживает псевдофайловую систему в каталоге **/dev**; файлы в этом каталоге представляют устройства, подключенные к системе. Когда ядро Linux обнаруживает подключенное устройство, демон udev используется для создания файла устройства (или узла) в каталоге **/dev**. Если устройство удалено, то демон udev удаляет узел устройства в каталоге **/dev.**

Файлы конфигурации в каталоге **/etc/udev/rules.d** используются для определения правил, которые присваивают конкретным владельцам, разрешениям и постоянным именам эти файлы устройств.

Когда ядро обнаруживает устройство, оно также помещает информацию об устройстве в файлы в каталоге **/sys**. Демон Hardware Abstraction Layer (HAL) отвечает за обнаружение и поддержание списка подключенных устройств и их атрибутов путем мониторинга этих файлов в каталоге **/sys**. Чтобы просмотреть список устройств и их атрибутов, которые были сохранены командой **hald**, выполните команду **lshal.**

Вывод команды **lshal** будет содержать тысячи строк. Чтобы просмотреть информацию о конкретных устройствах, используйте команду **grep**, как показано в следующем примере:

```
lshal | grep cdrom | grep true
  storage.cdrom.dvd = true   (bool)
  storage.cdrom.mrw = true   (bool)
  storage.cdrom.mrw_w = true   (bool)
  storage.cdrom.support_media_changed = true   (bool)
  storage.cdrom.support_multisession = true   (bool)
```

Когда программы хотят получить информацию об устройствах, они могут запросить запрос с помощью **dbus**. Программы могут также зарегистрироваться в **dbus** для получения уведомлений от **hald**, когда происходят определенные типы аппаратных событий. Когда состояние аппаратного устройства изменяется, hald использует dbus для отправки уведомлений тем программам, которые были зарегистрированы для этого типа аппаратного события.

Модули ядра

В дополнение к обнаружению устройств ядро Linux может загружать программное обеспечение, называемое модулями ядра, для поддержки устройства. Некоторые устройства настолько распространены, что программное обеспечение для их поддержки

обычно компилируется в само ядро, например, программное обеспечение для процессора. Другие не столь распространенные устройства будут иметь модули, которые загружаются только при обнаружении устройства, например, определенная сетевая карта.

Модуль ядра может даже загружать дополнительное программное обеспечение, такое как встроенное программное обеспечение, из файла или с самого устройства. Одной из проблем некоторых пользователей является то, что эти файлы встроенного программного обеспечения могут содержать код, который не является открытым исходным кодом, что означает, что у пользователя нет возможности узнать, что именно делает код, потому что он является частным и не доступен для общественности. Если ядро загружает «несвободный» ("non-free") код, то ядро считается «испорченным», поскольку код самого ядра является бесплатным.

Модули ядра можно использовать не только для поддержки устройств. Поскольку модули - это просто программное обеспечение, которое может работать в ядре, их можно использовать практически для всего. Некоторые общие применения, помимо драйверов устройств, включают в себя модули файловых систем, модули сетевых протоколов и модули криптографических алгоритмов.

Чтобы просмотреть список загруженных модулей ядра, используйте команду **lsmod**:

```
root@L-FW:~# lsmod | grep ext4
ext4                  585728  1
crc16                  16384  1 ext4
jbd2                  106496  1 ext4
fscrypto               28672  1 ext4
mbcache                16384  2 ext4
```

Вывод выше показывает модули, которые работают вместе для поддержки файловой системы ext4. Столбцы данных:

Первый столбец: имя загруженного модуля

Вторая колонка: размер в байтах модуля

Третий столбец: указывает, сколько «вещей» зависит от текущего загружаемого модуля. Эти «вещи» могут включать в себя другие модули, процессы или другие функции (например, монтируемую файловую систему).

Четвертый столбец: показывает название модуля, которое зависит от текущего модуля.

Более подробную информацию о модуле можно узнать с помощью команды modinfo. Например, чтобы узнать больше о модуле snd, выполните команду modinfo snd:

```
root@L-FW:~# lsmod | grep ext4
ext4                  585728  1
crc16                  16384  1 ext4
jbd2                  106496  1 ext4
fscrypto               28672  1 ext4
mbcache                16384  2 ext4
root@L-FW:~# modinfo snd
filename:       /lib/modules/4.9.0-7-amd64/kernel/sound/core/snd.ko
alias:          char-major-116-*
license:        GPL
description:    Advanced Linux Sound Architecture driver for soundcards.
author:         Jaroslav Kysela <perex@perex.cz>
depends:        soundcore
retpoline:      Y
intree:         Y
vermagic:       4.9.0-7-amd64 SMP mod_unload modversions
parm:           slots:Module names assigned to the slots. (array of charp)
parm:           major:Major # for sound driver. (int)
parm:           cards_limit:Count of auto-loadable soundcards. (int)
```

Чтобы получить список всех доступных модулей, используйте команду **modprobe -b:**

```
modeprobe –b | head
```

Обычно модули ядра загружаются ядром автоматически. Чтобы загрузить модуль вручную, выполните команду modprobe с именем модуля.

Например, чтобы загрузить модуль ext4, выполните команду **modprobe ext4**. Обратите внимание, что если модуль загружен успешно, эта команда не выводится.

Приятной особенностью команды modprobe является то, что она автоматически загружает все модули зависимостей, поэтому в случае загрузки модуля ext4 она автоматически загружает модули **crc16**, **mbcach**e и **jdb2**.

Команду **modprob**e также можно использовать для удаления модулей из памяти с помощью параметра **-r.** Выполнение команды modprobe -r ext4 приведет к удалению модуля ext4 из памяти и автоматически удалит модули crc16, mbcache и jdb2 из памяти, если от них больше ничего не зависит:

```
lsmod | grep ext4
ext4                  465245  2
crc16                  12503  1 ext4
mbcache                17476  1 ext4
jbd2                   82480  1 ext4

modprobe -r ext4
lsmod | grep ext4
```