LINUX

Что такое Linux?

История и определение

В 1969 году в дочернем подразделении компании AT&T – Bell Laboratories – была выпущена операционная система Unix, которая стала основной для большого количества операционных систем того времени. UNIX была проприетарной системой, лицензия на нее стоила порядка 40 000 долларов. Таким образом, позволить себе ее покупку могли только крупные компании. Это послужило толчком для старта в 1983 году проекта GNU – GNU is Not Unix. Его основоположник, Ричард Столлман, объявил целью проекта создание свободно распространяемой операционной системы. Чуть позже Столлманом был написан знаменитый манифест GNU, который стал основой для лицензии GPL (GNU General Public License), актуальной и по сей день. К началу 1990-х годов в рамках проекта GNU было написано большинство компонентов ОС – оболочка bash, компиляторы, отладчик, текстовый редактор и др. Не хватало лишь ядра операционной системы.

В 1991 году Линус Торвальдс, будучи студентом финского университета, увлекся идеей написания ядра операционной системы для своего персонального компьютера с процессором Intel. Вдохновлением и прототипом для будущего Linux стала совместимая с Unix операционная система для персональных компьютеров Minix. Уже в августе 1991 года было написано ядро операционной системы, в нее были портированы оболочка bash и компилятор gcc из проекта GNU. По признанию самого Линуса, изначально это было не более, чем хобби, однако проект оказался весьма востребованным, к нему начали присоединяться разработчики со всего мира. Дополненная массой программ, разработанных в рамках проекта GNU, ОС Linux стала пригодна для практического использования. При этом ядро системы распространялось под лицензией GNU General Public License, что гарантировало свободное распространение кода ядра системы.

Серверные дистрибутивы

Дистрибутив Linux — это операционная система, созданная на основе ядра Linux, которая включает в себя набор библиотек и утилит (пакетов), разработанных в рамках проекта GNU, а также систему управления пакетами (менеджер пакетов). В настоящее время существует более 500 различных дистрибутивов, разрабатываемых как при коммерческой поддержке (Red Hat / Fedora, SLED / OpenSUSE, Ubuntu и др.), так и исключительно усилиями добровольцев (Debian, Slackware, Gentoo, ArchLinux и др.).

Дистрибутивы делятся на несколько типов в зависимости от базового дистрибутива и системы управления пакетами. Вот несколько примеров наиболее популярных серверых дистрибутивов двух типов:

- RPM-based (используют формат пакетов .rpm)
  RedHat Enterprise Linux, CentOS, Fedora
- DEB-based (используют формат пакетов .deb)
  Debian, Ubuntu, Astra Linux

Стоит отметить, что существует великое множество различных дистрибутивов на любой вкус и цвет. Их невозможно даже сосчитать, так как практически каждый представитель сообщества может собрать свой собственный дистрибутив на основе ядра Linux.

Работа с Linux

Загрузка

Алгоритм включения сервера и загрузки Linux в большинстве случаев выглядит следующим образом:

- BIOS / UEFI → MBR / GPT
  Выполняется код, заложенный производителем аппаратного обеспечения. Этот код проводит тестирование системы POST (Power On Self Test) и передает управление загрузчику в MBR (Master Boot Record) / GPT (GUID Partition Table)
- MBR / GPT → GRUB2 (существуют и другие загрузчики)
  Загрузчик из MBR / GPT очень простой – он способен только найти на диске и запустить следующий загрузчик. Как правило это GRUB2, но существуют и другие загрузчики, например LILO (в настоящее время практически не используется)
- GRUB2 → Kernel
  GRUB2 расположен на разделе жесткого диска в каталоге /boot. GRUB2 загружает ядро Linux (vmlinuz)
- Kernel → Init
  Ядро запускает процесс инициализации операционной системы. Как правило это SystemD, но существуют и другие системы инициализации, например SystemV (в настоящее время практически не используется). Процесс инициализации запускает все остальные процессы в системе

Подключение

Командная оболочка

Подключиться к Linux для управления можно в интерфейсе командной строки (command-line interface, CLI) или в графическом интерфейсе (graphical user interface, GUI). При работе с серверной инфраструктурой в подавляющем числе случаев GUI отсутствует и взаимодействие с сервером осуществляется в CLI. При входе пользователя на сервер в CLI запускается командная оболочка (в GUI командную оболочку можно запустить через эмулятор, например Terminal). Командная оболочка (shell) – это программа, которая принимает команды с клавиатуры и передает их операционной системе для выполнения. Наиболее распространенной командной оболочкой в Linux является GNU bash (Bourne Again SHell). bash основывается на другой легковесной оболочке-предшественнике – sh (Bourne sh), созданной Стефеном Борном.

Команды можно выполнять с помощью командной строки, указав имя двоичного (бинарного, bin) исполняемого файла или сценария. По умолчанию в Linux много команд, которые позволяют перемещаться по файловой системе, устанавливать ПО, конфигурировать его и выполнять другие действия. Каждая запущенная команда является отдельным процессом. Важно отметить, что в Linux (в отличие от Windows) почти всегда учитывается регистр, включая имена файлов и каталогов, команды, аргументы и опции.

Установка программ (утилит) пакетным менеджером

Зачем нужны пакетные менеджеры?

На заре развития Linux установить приложение (утилиту) можно было только путем скачивания исходного кода программы и компиляции. Это не практично и не слишком удобно для пользователей, поэтому были разработаны пакетные менеджеры. Установка приложений в них производится из пакетов – архивов с файлами скомпилированной программы. Большинство популярных дистрибутивов Linux содержат пакетные менеджеры, способные устанавливать любое программное обеспечение. Пакетные менеджеры имеют свой список репозиториев – серверов с базой пакетов. Во время установки алгоритм менеджера находит необходимый пакет в базе и производит автоматическое скачивание, установку и настройку.

Существует несколько форматов пакетов, однако наибольшее распространение получили .deb и .rpm. Рассмотрим операционные системы и менеджеры пакетов для данных форматов:

- DEB (.deb)
  ОС – DEB-based, например Debian, Ubuntu, AstraLinux
  Система управления пакетами – DPKG (работает только с локальными пакетами)
  Пакетный менеджер – apt
- RPM (.rpm)
  ОС – RPM-based, например RedHat Enterprise Linux, Fedora, CentOS
  Система управления пакетами – RPM (работает только с локальными пакетами)
  Пакетный менеджер – yum (в последних дистрибутивах заменен на dnf)

Структура файловой системы и работа с файлами

Типы файлов

Все объекты в Linux являются файлами. Существуют следующие типы файлов:

- Обычные файлы **-**
  Символьные и двоичные данные (текст, картинки, программы и др.)
- Каталог (директория) **d**
  Список ссылок на файлы или другие каталоги
- Символьные ссылки **l**
  Ссылки на другие файлы по имени
- Блочные устройства **b,** символьные устройства **c**
  Интерфейсы для взаимодействия с аппаратным обеспечением (диски, терминалы, клавиатуры, принтеры и др.). Когда происходит обращение к файлу устройства, ядро операционной системы передает запрос драйверу этого устройства
- Сокеты **s** и каналы **p**
  Интерфейсы для взаимодействия процессов

SSH

Подключаться к Linux и работать с командной оболочкой можно локально (например включив ПК дома или подойдя к серверу в центре обработки данных), однако гораздо чаще работать с системой требуется удаленно. Для этого необходимо настроить SSH и подключаться через него. SSH (Secure SHell) – это протокол, позволяющий производить удаленное управление операционной системой и туннелирование TCP-соединений (например, для копирования файлов). SSH основан на клиент-серверной архитектуре, которая организует защищенное (зашифрованное) соединение поверх небезопасных каналов связи. Серверная часть устанавливается на удаленном сервере, а клиентская на компьютере, с которого осуществляется подключение.

Структура файловой системы

Структура файловой системы представляет собой дерево, корнем которой является каталог /.

Рассмотрим подробно структуру и назначение каталогов:

- /bin (binaries) – исполняемые файлы самых необходимых утилит. Может быть символьной ссылкой на /usr/bin
- /boot – файлы, необходимые для самого первого этапа загрузки – загрузки ядра (и обычно само ядро)
- /dev (devices) – блочные и символьные файлы устройств (диски, терминалы, клавиатуры, принтеры и др.)
- /etc (etcetera) – конфигурационные файлы системы и различных программ
- /home – домашние каталоги пользователей для хранения «личных» файлов
- /lib (libraries) – файлы библиотек (стандартных функций, необходимых многим программам), необходимых для работы утилит. Может быть символьной ссылкой на /usr/bin
- /mnt (mount) – каталог для подключения файловых систем (съемных носителей и др.)
- /opt (optional) – каталог для дополнительных программ (проприетарных драйверов, агентов мониторинга и др.)
- /proc (process) – файлы в оперативной памяти, в которых содержится информация о выполняемых в системе процессах
- /root – домашний каталог пользователя root
- /sbin (system binaries) – файлы системных утилит, необходимые для загрузки, резервного копирования и восстановления системы. Может быть символьной ссылкой на /usr/sbin
- /sys (system) – виртуальная файловая система sysfs, которая содержит информацию об аппаратном обеспечении (ЦПУ, ОЗУ, дисках, сетевых устройствах), драйверах, ядре системы и др.
- /tmp – каталог для временных файлов, обычно зачищается при каждой загрузке системы
- /usr – пользовательский каталог, который содержит каталоги исполняемых файлов и конфигурационных файлов
- /var (variable) – файлы, создаваемые или используемые различными программами (логи, очереди, идентификаторы процессов, БД и др.)

Права доступа

В Linux права доступа к файлам (в том числе к каталогам) задаются для трех видов пользователей – владельца, группы владельца и остальных. Также есть три типа доступа к файлу – чтение **r** (Read), запись **w** (Write) и исполнение **x** (eXecution), которые задаются для каждого из видов пользователей. Прочерк **-** означает отсутствие доступа.

Таким образом, права доступа к файлу выглядят следующим образом:

- права для владельца (u, user) – read, write, execution
- права для группы владельца (g, group) – read, write, execution
- права для остальных пользователей (o, other) – read, write, execution

Пример: rwx r-- --- означает, что у владельца есть права на все, у группы владельца доступ только на чтение, а у остальных доступа нет. В двоичной системе счисления эти права выглядят как три группы цифр – 111 100 000, что равносильно трем цифрам 7 4 0 в восьмеричной и десятичной системах счисления.

Процессы и потребление ресурсов сервера

Процессы

Если предельно упростить, то процесс – это любая программа, которая выполняется в системе. В ходе работы с системой может быть запущено множество программ, которые, в свою очередь, могут запустить множество процессов. Простейший пример процесса – командная оболочка bash. Каждому процессу в Linux присваивается уникальный идентификатор процесса (PID), который используется ядром для управления процессом до завершения программы или команды, с которой он связан.

Процесс может находиться в следующих статусах:

- Выполнение (**R**, Running)
  Выполнение или ожидание ЦПУ для выполнения
- Сон (**S**, Sleep)
  Прерываемое программно ожидание
- Непрерываемый сон (**D**, Direct)
  Ожидание «прямого» сигнала от аппаратной части для прерывания
- Приостановлен (**T**, Tracing)
  Отладка
- Зомби (**Z**, Zombie)
  Выполнение завершено, однако ресурсы не освобождены

Почти любой процесс (кроме процесса в статусе D) может быть принудительно прерван администратором в случае необходимости («убит»). Это не всегда безопасно, однако возможно.

Запущенные процессы требуют использования аппаратных ресурсов сервера – ЦПУ, ОЗУ, дисков,

Практика

На практике пользователю необходимо просматривать списки процессов и останавливать процессы, а также просматривать имеющиеся и потребляемые ресурсы сервера в системе.

Ниже приведены несколько примеров наиболее часто используемых команд:

```
# руководство (справочная информация)
man <utility> # просмотр справочной информации по утилите
<utility> --help # просмотр справочной информации по утилите

# процессы
top
sudo ps aux
sudo kill -9 <pid> # убийство процесса по PID
sudo killall -s 9 <name> # убийство всех процессов по имени

# утилиты для мониторинга использования ресурсов
htop # использование ресурсов по процессам (может потребоваться установка пакета htop)
nmon # использование процессора по ядрам, памяти, дисков и др. (может потребоваться установ
iostat # использование процессора в среднем по ядрам и чтение/запись по дискам

# ЦПУ
lscpu # общая информация
cat /proc/cpuinfo # подробная информация

# ОЗУ
cat /proc/meminfo # общая информация и потребление
free -h # удобное представление на основе данных из файла meminfo
ps aux --sort -rss # использование памяти по процессам

# диски
lsblk # общая информация
df -h # просмотр занятного места по разделам
du -ch <dir> # просмотр занятого места в каталоге

# сеть
/sys/class/net/<interface>/speed # просмотр максимальной скорости интерфейса
```

Сеть

На практике пользователю необходимо уметь просматривать сетевые настройки сервера, а также уметь проводить простейшую диагностику сетевых проблем.

Ниже приведены несколько примеров наиболее часто используемых команд:

```
# руководство (справочная информация)
man <utility> # просмотр справочной информации по утилите
<utility> --help # просмотр справочной информации по утилите

# просмотр сетевых настроек сервера
ip a # IP адреса
ip n # ARP таблица
cat /etc/resolv.conf # конфигурация DNS
sudo netstat -tulpn # открытые порты
sudo ss -tulpn # открытые порты (молодежный вариант)

# диагностика сетевых проблем
nslookup <hostname> # проверка разрешения DNS имени
ping <host> # отправка ICMP пакетов до хоста
traceroute <host> # трассировка до хоста UDP пакетами
telnet <host> <port> # проверка доступности TCP порта на хосте
nmap <host> # сетевое сканирование хоста
nmap -p T:<port> <host> # проверка доступности TCP порта на хосте (молодежный вариант)
nmap -p U:<port> <host> # проверка доступности UDP порта на хосте (молодежный вариант)
```

`.bash_profile` — содержит информацию о пользовательском окружении и запускаемых при авторизации пользователя программах. В некоторых дистрибутивах, основанных на Debian, данного файла по умолчанию не существует, но ты можешь создать его самостоятельно;

`.bash_login` — этот файл исполняется, если отсутствует `.bash_profile`, и выполняет схожую функцию. Этого файла не существует по умолчанию ни в дистрибутиве Debian, ни в дистрибутиве Red Hat;

`.profile` — выполняется при отсутствии `.bash_profile` и `.bash_login`;

`.bash_logout` — сценарий, который выполняется автоматически при завершении работы командной оболочки;

`.bash_history` — хранит информацию обо всех командах, набранных в bash;

`.ssh` — каталог, в котором хранятся ключи шифрования для подключения по SSH;

`.bashrc` — сценарий, который обычно настраивается другими сценариями для своих собственных нужд — например, запуска демонов или обработки каких-либо команд.