Лекция 1. Основы Linux. Установка системы

Цель лекции

- Познакомиться с компонентами операционных систем
- Узнать историю появления Linux
- Посмотреть на классификацию дистрибутивов Linux
- Разобрать процесс установки Linux на виртуальную машину
- Сделать первые шаги в работе с текстовым интерфейсом

План

1. Операционные системы.
2. Мир до операционных систем.
3. Ядро, библиотеки, приложения.
4. Системные вызовы в Linux.
5. Северные и настольные системы.
6. Предшественники – UNIX-системы.
7. Краткая история Linux.
8. Проект GNU.
9. Применение Linux в серверной среде.
10. Что такое дистрибутив Linux?
11. Классификация дистрибутивов Linux.
12. Red Hat и Debian дистрибутивы.
13. Ubuntu. Desktop, Server, LTS и обычный релиз.
14. Виртуализация в VirtualBox.
15. Настройка виртуальной машины.
16. Установка Ubuntu Desktop в виртуальную машину на VitrualBox.
17. Графический интерфейс – обзор. Настройки, терминал, файловый менеджер.
18. Структура файловой системы.
19. Текстовый интерфейс и подключение по SSH.
20. Установка пакетов в Ubuntu.
21. Сетевые режимы Мост и NAT в VirtualBox. Проброс портов.
22. Установка гостевых дополнений VirtualBox.

Операционные системы.

Добро пожаловать на курс! На следующих занятиях вы научитесь уверенно работать с операционной системой Linux и решать типичные для разработчика задачи в этой операционной системе (ОС).

Итак, что такое Linux? Linux сегодня это стандарт де-факто в среде серверных операционных систем для большинства применений. На специальных версиях Linux работает огромное количество сетевых устройств, система Android также основана на Linux. Операционные системы на базе Linux можно встретить в облаках, государственном секторе, оборонных информационных системах и конечно в крупнейших интернет-компаниях. Навыки работы в этой системе открывают для вас широчайшие возможности по развитию профессиональному росту.
Для начала давайте определимся, что такое операционная система. Это прослойка между приложением, которое мы запускаем и железом, на котором оно работает. Операционная система (ОС) реализует работу сети, управление ресурсами компьютера, файловые системы и другие низкоуровневые элементы. Такая архитектура резко упрощает разработку приложений. Например, при записи данных на диск разработчику приложения не важно, где эти данные хранятся физически – на жестком диске, в сетевом хранилище или в облаке. Все эти сложности скрыты ОС от приложения и процесс записи представляется в виде абстракции “записать данные на диск”.

Мир до операционных систем

Первые операционные системы появились еще в 50-х годах 20 века. Для нас особенно важно появление в 1970 году операционной системы UNIX.
Но как же работали первые компьютеры до появления операционных систем?
В первых компьютерах программисты были вынуждены управлять железом напрямую, например, вызывать низкоуровневые инструкции процессора и управлять накопителями информации напрямую. Программы в этом случае содержали машинные коды (инструкции), никаких языков программирования не существовало. Узнать, какие инструкции поддерживает компьютер и как они работают, можно было только из технической документации. Из-за такого подхода каждая программа могла работать только на том железе, которое использовалось при разработке. Любое изменение конфигурации компьютера или его обновление требовало полностью переписать весь софт.
Сегодня самой близкой аналогией можно назвать язык ассемблер, который преобразуется непосредственно в машинные коды и исполняется на процессоре.
С одной стороны, за счет написания программ в машинных кодах можно было решать реальные задачи с крайне ограниченными вычислительными ресурсами компьютеров того времени. С другой стороны, это требовало высокой квалификации программистов и огромного объёма ручного труда по отладке такого кода.
Постепенно стоимость компьютерного оборудования снижалась, а стоимость труда программистов возрастала по мере роста сложности задач. В результате, стало выгодно снять нагрузку с разработчиков софта и упростить создание нового программного обеспечения (ПО). Большая часть низкоуровневых механизмов работы с железом ушла из прикладных программ в ОС.

Ядро, библиотеки, приложения.

Любую современную операционную систему можно условно разделить на несколько уровней. Прежде всего, ОС характеризуется ядром (kernel). Это низкоуровневый компонент, который отвечает за взаимодействие с железом. Также ядро предоставляет интерфейс для более высокоуровневых компонентов (библиотек и приложений). Этот интерфейс обычно реализуется в виде набора системных вызовов (system calls).
Библиотеки (libraries) представляют собой набор общих функций, которые часто используются в приложениях. Например, это может быть библиотека, реализующая алгоритм сжатия zip или работу какого-либо алгоритма шифрования для работы с HTTPS-сайтами. За счет использования библиотек разработчики могут быстро подключать новые возможности и технологии в свой продукт.
Приложения (applications) – верхний уровень этой архитектуры. То, ради чего построены все нижние уровни. Приложение может быть пользовательским (текстовый редактор, браузер, калькулятор) или серверным (веб-сервер, почтовый сервер, сервер баз данных). Часть приложений могут входить с состав операционной системы, остальные устанавливаются дополнительно. Важно отметить, что совместимость операционной системы с нужными приложениями является важным фактором выбора ОС под конкретную задачу.

Системные вызовы в Linux.

Посмотрим, какие задачи могут решать системные вызовы в ОС Linux. Все системные вызовы делятся на группы, например управление процессами, управление файлами, работа с памятью и т.д.
Например, чтобы создать новый процесс вызывается fork() – он создаёт полную копию исходного процесса и может продолжать выполнение программы или запустить новую через вызов exec(). Завершить программу можно вызовом exit().
Также происходит с файлами: открыть через open(), прочитать с помощью read(), записать через write() и закрыть – close().
При использовании системных вызовов разработчику не нужно знать подробностей реализации каждого вызова, например, на какой файловой системе находится файл, как расположены блоки на диске и какой он имеет объём.

Серверные и настольные системы.

Формально мы можем разделить ОС на серверные (server) и настольные (desktop).
Серверные системы нацелены на поддержку работы серверных приложений. Они должны обеспечивать повышенную стабильность и иметь высокий уровень безопасности. Должны поддерживать серверное железо. Такие системы большую часть времени работают без взаимодействия с пользователем в автоматическом режиме выполняют серверные приложения. Поэтому часто они не имеют графического интерфейса, что повышает их эффективность и надёжность. Если

необходимо взаимодействие с системой оно обычно осуществляется удалённо с помощью текстового интерфейса.
Настольные системы предлагают широкую совместимость с пользовательским ПО. Имеют графический интерфейс и, как правило, используются локально, без удалённого управления. Должны поддерживать потребительское железо. Главное – удобство и совместимость, надёжность и стабильность вторичны.
При этом часто серверные и настольные системы используют одно и то же ядро (за исключением искусственных ограничений) и отличаются набором библиотек, драйверов, средств управления и уровнем тестирования компонентов.
Например, ядро Linux универсально и поддерживает любые применения.

Предшественники – UNIX-системы.

Предками Linux являются UNIX-системы. Это семейство ОС, объединенных общими принципами работы и подходами к функциональности. В большинстве своём это были коммерческие ОС, разработанные крупными корпорациями для работы на железе их собственной разработки. Сюда относятся системы AIX от компании IBM, HP-UX от компании HP, Irix от от компании Silicon Graphics (SGI), Solaris от компании SUN.
В основе UNIX-систем заложены несколько важных принципов, которые позволили сильно продвинуть развитие ОС и упростить жизнь разработчикам ПО. Часть UNIX-принципов мы будем рассматривать на занятиях этого курса. Например, важным принципом является “каждая программа должна делать что-то одно, но делать это хорошо”. На основе этого подхода в UNIX-мире разработано огромное количество простых утилит для решения задач (работа с файлами, текстом, процессами), которые могут объединяться в цепочки (конвейеры) для комплексных применений.
Коммерческая лицензия сильно ограничивала использование ОС как в учебных целях, так и в бизнесе. Поэтому были разработаны открытые операционные системы (open source), например FreeBSD. Такие системы можно было свободно распространять и вносить изменения через сообщества.
Потребность в совместимости различных UNIX-систем породила стандарт POSIX (Portable Operating System Interface — переносимый интерфейс операционных систем), который собрал все лучшие практики в мире UNIX и позволил легко переносить готовый софт между совместимыми ОС.

Краткая история Linux.

На базе всего наследия UNIX-систем на свет появляется Linux. Первоначально это студенческий проект Линуса Торвальдса. На первых этапах разработки использовалась учебная ОС Minix профессора Эндрю Таненбаума, далее её компоненты были заменены на новые.

С точки зрения принципов построения ПО Linux наследует многие принципы UNIX-систем, часто имеет одинаковые названия программ и параметров. Это позволило быстро переходить на использование Linux для специалистов по UNIX-системам.
Однако, изначально Linux представлял из себя только ядро и для полноценной системы не хватало важных компонентов: компиляторов, оболочки, утилит. В качестве донора этих компонентов для Linux выступает проект GNU.

Проект GNU.

Проект GNU был запущен как антипод коммерческим закрытым системам UNIX. То есть стояла задача повторить все лучшие черты UNIX-систем, убрав лицензионные ограничения. В основу были положены принципы открытости исходного кода и свободного распространения ПО. Именно этот проект дал импульс развитию движения свободного ПО, в основе которого лежит открытый исходный код и либеральная лицензия на использование.
В проекте GNU было одного слабое место – ядро. Именно здесь как нельзя кстати приходит Linux. В результате получаем систему GNU/Linux, однако для краткости систему принято называть просто “Linux”.

Применение Linux в серверной среде.

Сегодня Linux является стандартом де-факто в разряде платформ для веб-приложений. Гибкость и масштабируемость системы позволяет использовать её в самых различных применениях: от мобильных устройств (Android) до суперкомпьютеров. Вы наверняка столкнётесь с Linux в серверном варианте при первых шагах в профессии. Кроме того, многие разработчики предпочитают десктопный вариант Linux для своих рабочих машин.

Что такое дистрибутив Linux?

Ранее мы говорили про операционную систему Linux. Но также упоминали, что Linux строго говоря это только ядро. На самом деле мы пользуемся готовыми дистрибутивами, которые содержат все необходимые компоненты ОС. Что же такое “дистрибутив”?
Дистрибутив Linux – это готовая к использованию сборка всех необходимых компонентов ОС (ядро, библиотеки, оболочка, средства разработки, прикладные программы). В зависимости от направленности дистрибутив может иметь различную комплектацию: например, настольные версии имеют графический интерфейс, а серверные его не имеют. Так как области применения Linux очень разнообразные, существует огромный выбор дистрибутивов – более 200. Однако, не стоит отчаиваться – в таком разнообразии нет большой проблемы. Большинство отдельных “дистрибутивов” это просто клоны одной и той же системы с различной графической

оболочкой. Если же отбросить десктопные системы и говорить только о серверных, остаётся выбор нескольких ветвей развития.

Классификация дистрибутивов Linux.

Разделяются дистрибутивы по комплекту поставки и принятой модели упаковки программного обеспечения.
Прежде всего важна система пакетов ПО. Пакет – это готовое к установке ПО, упакованное в архив. Система пакетов отвечает за автоматизацию установки, удаления и конфигурации ПО. Существуют дистрибутивы без пакетных систем, но их меньшинство.
Наиболее популярные серверные дистрибутивы можно разделить на две группы: “Red Hat” и “Debian”.

Red Hat и Debian дистрибутивы.

Первая группа дистрибутивов может условно называться “Red Hat” – дистрибутивы, основанные на системе пакетов RPM (red hat package manager).
Вторая группа это Debian – дистрибутивы, основанные на DEB-пакетах. Внутри этой группы находится система Ubuntu, которую мы и будем использовать на занятиях.
В третью группу можно вынести все остальные дистрибутивы — для серверного рынка это скорее экзотика и можно не заострять на них своё внимание.
Таким образом, если получить опыт работы с дистрибутивами из первых двух групп, можно совершенно спокойно чувствовать себя с Linux-системами. При этом разница между дистрибутивами будет проявляться в командах по установке пакетов и размещению конфигурационных файлов. Также различия могут проявиться в наличии ПО, которое поставляется в комплекте. Некоторые дистрибутивы стремятся к стабильности и редко обновляют ПО, другие же наоборот предоставляют новейшие версии софта. Все основные принципы и архитектура Linux остаются неизменными.

Ubuntu. Desktop, Server, LTS и обычный релиз.

Ubuntu это серия дистрибутивов, поставляемых компанией Canonical. Прежде всего мы разделяем серверную и настольную версии. Внутри этих веток есть 2 типа релизов: LTS (Long Term Support – длительная поддержка) и не-LTS.
LTS версия выходит каждые 2 года и отличается повышенной стабильностью и сроком поддержки 5 лет после выхода релиза. То есть, LTS это лучший кандидат на установку для решения критичных задач.
Не-LTS версия выходит каждые полгода и содержит новейшие версии пакетов. В какой-то степени это экспериментальная площадка для создания следующего LTS-дистрибутива.
Серверная система не имеет графического интерфейса и в стандартном комплекте содержит пакеты для основных сервисов (веб-сервер, SSH, почтовый сервер, средства контейнеризации и т.д.)

Настольный вариант Ubuntu имеет полноценный графический интерфейс, установщик здесь также графический (в отличие от серверного). Но при этом на десктопной системе также доступны все серверные компоненты.
Таким образом, мы можем использовать настольную версию в этом курсе, получая преимущества графического интерфейса для первых шагов освоения системы. При этом все серверные компоненты также будут доступны.

Виртуализация в VirtualBox.

Освоить систему Linux нам поможет технология виртуализации. Это технология программной эмуляции компьютера для запуска операционной системы в качестве приложения. То есть, независимо от вашей основной операционной системы мы получаем рабочий экземпляр Linux в качестве виртуальной машины.
В процессе прохождения курса мы будем использовать виртуальную машину с Ubuntu. Нам не придётся устанавливать Ubuntu на реальный жесткий диск или выделять отдельный учебный компьютер. Также виртуальную машину очень просто сохранить, клонировать или восстановить. В качестве платформы виртуализации используем продукт VirtualBox от компании Oracle.
Такое решение позволяет использовать аппаратные возможности виртуализации современных процессоров. VirtualBox работает на Windows, Linux и MacOS.
Виртуальные машины позволяют свободно экспериментировать, выполнять учебные задачи без опасения за состояние основной операционной системы.
Ваша основная система будет называться хостовой (основной), виртуальная машина – гостевой.

Настройка виртуальной машины.

Перейдём к процессу создания настройки виртуальной машины. Основные требования нашей системы: около 2 GB оперативной памяти и 20 GB дискового пространства.
Создадим виртуальную машину в VirtualBox.

Установка Ubuntu Desktop в виртуальную машину на VitrualBox.

Для установки нам потребуется загрузочный диск дистрибутива Ubuntu 22.04 Desktop (ISO-образ). Скачать его можно в разделе Download на сайте ubuntu.com.
Далее нам нужно подключить этот диск в качестве загрузочного в ВМ.
Теперь всё готово к установке системы.

Графический интерфейс – обзор. Настройки, терминал, файловый менеджер.

Давайте освоимся в графическом интерфейсе Ubuntu.

Для начала выставим комфортное разрешение экрана в настройках (далее мы решим эту проблему установкой гостевых дополнений).
Слева у нас панель запуска приложений.
Сверху есть панель уведомлений, справа есть переход в меню с настройками командами выключения/перезагрузки системы.
Посмотрим состав базовых настроек – здесь самое необходимое.

Структура файловой системы.

Общая иерархия файловой системы в Linux отличается от модели Windows. Если в Windows отдельные разделы и диски принято обозначать буквами, то в Linux есть единая иерархия директорий, которая начинается с корня (root или /).
Все остальные диски и разделы подключаются (монтируются) в эту общую систему.
В корневой файловой системе существует набор стандартных директорий, каждая из которых несёт свою функцию. Эти директории описаны в стандарте FHS (Filesystem Hierarchy Standard). Стандарт FHS определяет, какие по назначению файлы должны находиться в определённых директориях. Например, исполняемые файлы должны находиться в каталоге /bin или /usr/bin, настройки в /etc и т.д. Подробнее назначение директорий показано ниже.
Ниже представлены основные стандартные каталоги в Linux.

/	Корневой каталог, содержащий всю файловую иерархию.
/bin	Основные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям (например: cat, ls, cp).
/boot	Загрузочные файлы (в том числе файлы загрузчика, ядро, initrd, System.map). Часто выносится на отдельный раздел.
/dev	Основные файлы устройств (например, /dev/null, /dev/zero).
/etc	Общесистемные конфигурационные файлы (имя происходит от лат. et cetera).
/home	Содержит домашние каталоги пользователей, которые в свою очередь содержат персональные настройки и данные пользователя. Часто размещается на отдельном разделе.
/lib	Основные библиотеки, необходимые для работы программ из /bin и /sbin.
/media	Точки монтирования для сменных носителей.
/mnt	Содержит временно монтируемые файловые системы.
/opt	Дополнительное программное обеспечение.

/proc	Виртуальная файловая система, представляющая состояние ядра операционной системы и запущенных процессов в виде файлов.
/root	Домашний каталог пользователя root.
/run	Информация о системе с момента её загрузки, в том числе данные, необходимые для работы демонов (pid-файлы, UNIX-сокеты и т.д.)
/sbin	Основные системные программы для администрирования и настройки системы, например, init, iptables, ifconfig.
/srv	Данные для сервисов, предоставляемых системой (например, www или ftp).
/sys	Содержит информацию об устройствах, драйверах, а также некоторых свойствах ядра.
/tmp	Временные файлы (см. также /var/tmp).
/usr	Вторичная иерархия для данных пользователя. Содержит большинство пользовательских приложений и утилит, используемых в многопользовательском режиме. Может быть смонтирована по сети только для чтения и быть общей для нескольких машин.
/usr/bin	Дополнительные программы для всех пользователей, не являющиеся необходимыми в однопользовательском режиме.
/usr/lib	Библиотеки для программ, находящихся в /usr/bin и /usr/sbin.
/usr/local	Третичная иерархия для данных, специфичных для данного хоста. Обычно содержит такие подкаталоги, как bin, lib, share.
/usr/sbin	Дополнительные системные программы (такие как демоны различных сетевых сервисов).
/var	Изменяемые файлы, такие как файлы регистрации, временные почтовые файлы, файлы спулеров.
/var/lib	Информация о состоянии. Постоянные данные, изменяемые программами в процессе работы (например, базы данных, метаданные пакетного менеджера и др.).
/var/log	Различные файлы журналов (логи).

Важно заметить, что при установке ПО размещается сразу в нескольких каталогах: исполняемые файлы в /bin, настройки в /etc/ и т.д. Поэтому важно использовать пакетные менеджеры для управлением ПО в Linux.

Текстовый интерфейс и подключение по SSH.

Основной вариант управления для серверных Linux-систем – текстовый интерфейс. Именно такой вариант позволяет максимально эффективно управлять сервисами. Текстовые команды могут составлять часть скриптов, которые автоматизируют многие действия по администрированию системы.
Еще одно важное преимущество текстового интерфейса – универсальность, то есть независимость от дистрибутива. Почти все базовые команды будут работать одинаково в любом варианте Linux.
Команды в текстовом интерфейсе вводим через оболочку (shell) с использованием терминала или консоли. В следующем занятии мы подробнее разберём все эти термины.
Давайте посмотрим, как можно запустить терминал в Ubuntu. У нас есть 2 варианта – через меню приложений и по сочетанию клавиш Ctrl+Alt+T.
При этом основным способом управления сервером является подключение по протоколу SSH (Secure SHell). Этот протокол позволяет удалённо и безопасно подключаться к системе для управления через оболочку. Безопасность достигается за счет сквозного шифрования данных сеанса.

Установка пакетов в Ubuntu.

Самый простой вариант установки софта в Ubuntu это пакетный менеджер apt. Процесс установки пакетов максимально автоматизирован и требует лишь знания названия пакета.
Формат команды установки пакета следующий:
sudo apt install {название пакета}
Команда sudo запускает следующую команду в режиме суперпользователя (администратора), apt – команда пакетного менеджера в Ubuntu, install – режим установки пакета. Далее нам остаётся указать название пакета. Его можно найти в Интернете или в консоли. После ввода команды обычно требуется ввести пароль от вашего основного пользователя (на несколько минут он кэшируется и не запрашивается повторно).
Если вы обнаружили, что пакет не устанавливается, необходимо проверить связность с Интернет. Если связь есть, но пакет не устанавливается необходимо запустить команду:
sudo apt update
Эта команда скачает новые списки пакетов из настроенных репозиториев. После неё нужно повторить установку.
Для примера установим пакет openssh-server, который понадобится нам в дальнейшем.
sudo apt install openssh-server

Сетевые режимы Мост и NAT в VirtualBox. Проброс портов.

Для подключения к виртуальной машине по сети мы должны выбрать и настроить сетевой режим в VirtualBox.
Наиболее удобный это “Сетевой мост”. В этом случае мы получаем виртуальную машину в нашей локальной сети и можем обращаться к ней как к обычному компьютеру. Однако, при использовании беспроводного подключения в хостовой системе режим Сетевой мост может не работать. Настройка сетевого моста простая – нужно только выбрать основной адаптер, через который хостовая система подключена к сети.
Продемонстрируем подключение по SSH с режимом “Сетевой мост”.
Альтернатива – режим NAT (Network Address Translation). В таком режиме виртуальная машина получает доступ во внешний мир без ограничений. Но подключиться к самой машине по сети мы можем только через механизм проброса портов.
Для того, чтобы иметь возможность подключаться к машине по протоколу SSH нам нужно сделать проброс портов 8022 (хост) – 22 (гость).

Установка гостевых дополнений VirtualBox.

Если мы используем настольный вариант Ubuntu, будет удобно установить дополнения гостевых ОС для VirtualBox. Дополнения гостевых ОС позволяют использовать общие папки, общий буфер обмена между гостевой и хостовой ОС, а также автоматически масштабируют разрешение экрана гостевой ОС под размер окна.
Для этого подключаем образ диска дополнений гостевой ОС через меню “Устройства” в VirtualBox.
Далее заходим в терминал Ubuntu и устанавливаем необходимые пакеты (перечисляем пакеты через пробел, чтобы установить всё сразу):
sudo apt install gcc make perl
После этого запускаем установку из файлового менеджера.
Для применения изменений нужно перезагрузить виртуальную машину после установки дополнений.

После установки гостевых дополнений нам становится доступными:

- автоматическое масштабирование экрана по размеру окна;
- общий буфер обмена (между хостовой и гостевой системами);
- общие директории для обмена файлами.

Итоги занятия

- Узнали, какие бывают операционные системы

- Изучили историю появления системы Linux
- Рассмотрели основные дистрибутивы Linux
- Установили виртуальную машину с Ubuntu Linux
- Подготовили площадку для прохождения курса