Ядро операционной системы выполняет 2 главные задачи:

1. Общение с железом.
2. Распределение ресурсов.

**Почему мы изучаем именно GNU/Linux?**

1. Открытость. Надежность. Популярность. Стабильность.
2. Мощь администрирования. Даже PowerShell в Windows создан с учетом опыта Linux-администрирования.
3. Каждый оценит прозрачность и предсказуемость Linux в администрировании.
4. Знание Linux обязательно для системных администраторов, веб-разработчиков, DevOps.
5. Требует научиться мышлению Linux-администрирования. RTFM. Google.

**UNIX-подобная ОС.** Это значит, что архитектура, идеология и удобство работы напрямую исходят из истории и архитектуры UNIX-систем. Усвоив архитектуру Linux, просто перейти к работе с другими UNIX-подобными ОС: FreeBSD, OpenBSD, OpenSolaris, Mac OS X.

**Многопользовательская операционная система** с удачной реализацией управления пользователями и правами, благодаря подобию UNIX.

**Многозадачная операционная система, которая использует механизмы защиты процессора.** Ядро Linux изначально было написано под процессор Intel 80386. Поэтому, в отличие от DOS и Windows 9x, в GNU/Linux не нужно было обеспечивать совместимость в ущерб надежности. В процессоре Intel 80386 уже стала классической полноценная поддержка колец защиты процессора. Это система впервые появилась в процессоре 80286 и позволяла отделить код ядра операционной системы от кода приложений. До этого весь код выполнялся в одном режиме, имея доступ ко всем ресурсам.  Поэтому, например, DOS мог напрочь зависнуть в результате ошибки в прикладной программе. Очень долго Microsoft в целях совместимости приходилось поддерживать выполнение определенных фрагментов пользовательского кода на уровне ядра и в Windows. Это приводило к появлению синего экрана смерти и необходимости перезагрузки. В Linux не было необходимости поддерживать старый код, потому что операционная система сразу использовала механизмы защиты процессора и сейчас является довольно устойчивой и надежной. Kernel panic — сообщение-аналог синего экрана смерти в WIndows, которое говорит о крахе ядра, в Linux можно увидеть очень редко.  Разве что в процессе отладки драйверов для Linux, работающих как модули ядра.

**Сетевая операционная система**, реализующая стек TCP/IP на уровне ядра и обладающая широким набором сетевого программного обеспечения: веб-сервера, почтовые сервера, системы мониторинга и т.д. и т.п.

**Свободная операционная система.** Linux распространяется по лицензии GNU GPL, которая, с одной стороны, позволяет распространять ОС бесплатно, а с другой — обязывает предоставлять исходные коды. Это делает разработку, поиск ошибок и их исправление  прозрачными и надежными. Поэтому многие решения для Linux являются опробованными и надежными, а найденные уязвимости быстро закрываются сообществом. Далеко не всегда то же самое можно сказать о проприетарном (собственническом) программном обеспечении. Но есть и Enterprise-решения — коммерческие решения, созданные на базе свободного Linux. Самый известный пример — Red Hat. Основа остается бесплатной, но производитель может добавлять дополнительные услуги и решения, платную поддержку. Иногда встречается и откровенное нарушение лицензии GPL, когда ОС полностью продается без поставки исходного кода, например Linux XP и, возможно, МСВС. Тем не менее, большая часть операционных систем Linux остается открытой, и даже у коммерческих версий есть функциональные аналоги, например Centos для Red Hat.  Но в таком случае система используется на свой страх и риск — поддержки от производителя нет.

**Надежная операционная система.**  В Linux продуманная система прав, наследуемая из UNIX. Linux не отягощена наследием неверного использования модели безопасности, как Windows, где система безопасности не используется в полную меру из соображений совместимости. Наконец, Linux — прозрачная система, где быстро закрываются все уязвимости. Все вместе это дает надежную систему, на которую всегда можно положиться.

**Операционная система, дающая полный контроль пользователю.** Linux позволяет системному администратору или разработчику полностью управлять операционной системой: разрабатывать скрипты и автоматизировать рутинные действия. Здесь и возможности командной строки, и использование конвейеров и виртуальных файловых систем с доступом ко всем параметрам ядра ОС, процессов и устройств.

**Опробованная система с широким опытом внедрения.** Linux лидирует в мире серверного программного обеспечения, а также используется на десктопах, в т.ч., во встраиваемом программном обеспечении. Это дает огромный пользовательский опыт в технологическом и специализированном программном обеспечении, сформированные пользовательские сообщества. Благодаря этому, во-первых, есть уверенность, что решение испытано, а во-вторых, можно обращаться к пользовательскому опыту в случае возникновения сложностей.

**Операционная система с возможностью работы в графическом оконном режиме.** Механизм X Windows Server в Linux вместе с SSH позволяет реализовать даже удаленное выполнение оконных приложений. Это удобнее, чем, например, механизм RDP. Кроме того, он включает набор сред рабочего стола, например, Gnome, KDE, и позволяет выбрать систему рабочего стола по вкусу. При этом приложения выполняются независимо от используемой среды. И, наконец, проект Wine в Linux реализует трансляцию вызовов WinAPI в системные вызовы ядра и тем самым позволяет запускать в оконной среде Linux приложения Windows.

**Операционная система с возможностью виртуализации.** Набор механизмов Linux позволяет как изолировать пользователей и процессы (как cgroups и chroot), так и запускать гостевые ОС на хост-системе. Для этого используются механизмы виртуализации на уровне ядра, например kvm (в том числе и для Windows-машин), либо контейнерная виртуализация, то есть на уровне ОС, когда запускаются в гостевом режиме другие экземпляры Linux. Например, это LXC или OpenVZ.

**Кому и зачем нужен Linux?**

* Системные администраторы
* Специалисты по информационной безопасности
* Тестировщики
* Разработчики PHP/JS
* Python разработчики
* Разработчики веб-приложений на Java
* Data Science

**Немного истории**

…

В 286 впервые появились механизмы защиты процессора. В 386 механизмы были доработаны и позволили полноценно разрабатывать ОС с механизмами защиты. Большинство современных ОС (и Linux в т.ч.)  используют только 2 кольца из 4-х, с наибольшим приоритетом для ядра ОС и с наименьшим приоритетом для пользовательского окружения.

Linux стал ядром для GNU.

**Сетевые возможности**

**Что нужно знать начинающему пользователю Linux?**

Компьютеры, работающие в сети, взаимодействуют благодаря **IP-адресам**. **IP-адрес** позволяет глобально идентифицировать ЭВМ в сети, где бы компьютер ни находился. Если вы в Москве, ваш компьютер с IP-адресом 1.2.3.4 подключен к Интернету, вы можете установить связь с компьютером, находящимся, скажем, в Лос-Анджелесе, и имеющим IP-адрес 130.140.105.1

**IPv4-адрес** (как правило, используются адреса IP версии 4, хотя постепенно и осуществляется переход на **IPv6**) состоит из 4-х чисел (байт), именуемых октетами, и записываемых, как правило, в десятичной системе счисления. Есть специальные IP-адреса (адреса сетей и широковещательные адреса), но большинство IP-адресов служат для идентификации хостов, то есть неких машин (не важно, в роли клиентов или серверов они выступают). Машина, которая предоставляет некие услуги, обычно именуется сервером, машина, которая запрашивает эти услуги, выступает клиентом. Терминология клиент-сервер применяется как к аппаратному обеспечению (ваш ноутбук будет выступать клиентом, а компьютер-сервер в серверной стойке в дата-центре, соответственно, сервером), так и к программному (например, если веб-сервером выступает Apache или Nginx, роль клиента на вашем компьютере будет играть Google Chrome или, например, Mozilla Firefox).

Для подключения к серверу используется **IP-адрес**. Но на практике часто в качестве адреса сервера используются доменные имена: geekbrains.ru, yandex.ru, google.com, linkedin.com. Доменные имена благодаря распределенной системе **DNS** преобразуются в **IP-адреса**. То есть, когда вы в браузере вбиваете имя geekbrains.ru, ваш компьютер, выступая в роли **DNS-клиента**, обращается к **DNS-серверу** (предоставленному провайдером или настроенному вами вручную), чтобы узнать, на какой **IP-адрес** должен подключиться браузер в роли **http-клиента**. Выяснив, что **IP-адрес**, соответствующий доменному имени geekbrains.ru — 5.61.239.22, браузер подключится к серверу, обращаясь к машине с IP 5.61.239.22. Сообщения, которыми будут обмениваться клиент и сервер, адресуемые на **IP-адреса** отправителя и получателя (похоже на телеграммы, не правда ли?), именуются **IP-пакетами** (и иногда IP-дейтаграммами). **IP-пакеты** могут проходить несколько маршрутизаторов, компьютеров или устройств, определяющих дальнейший путь в соответствии с таблицами маршрутизации. Это обеспечивается, в частности, тем, что часть **IP-адреса** составляет адрес сети, а часть — идентификатор хоста. **Например**, для некоего условного адреса IP 199.20.30.5 — 199.20.30 — компонент, который указывает на сеть, а 5 — на конкретную машину в данной сети.

Не все адреса могут маршрутизироваться. Например, пакет, направленный на адрес 127.0.0.1, никогда не покинет машину. Он будет доставлен другому приложению, находящемуся на данной машине. Стоит отметить, что это тоже нормальный способ использования, например, PHP-скрипт, выполняемый на сервере, может обращаться к приложению MySQL, находящемуся на той же машине, используя адрес 127.0.0.1. Такой адрес называется локальная петля. Для таких целей может использоваться любой IP-адрес, начинающийся с 127.

Есть адреса, которые используются для локальных сетей. Вы можете объединить несколько машин в локальную сеть и использовать адреса из диапазонов таких сетей, например 192.168.1.X или 10.X.X.X — и машины смогут обмениваться между собой информацией. Но нельзя обратиться на машину с адресом 192.168.1.1 из сети Интернет. Более того, и обратное не было бы верным без специальных средств. Если вы посмотрите настройки TCP/IP-соединения, то обнаружите, что, скорее всего, у вас тоже используется адрес из такого диапазона. Но как тогда осуществляется выход в сеть?

Если вы зайдете на сайт вроде myip.ru, то увидите, что адрес, под которым вы видны указанному серверу, не совпадает с вашим собственным IP-адресом, указанном в настройках TCP/IP-соединения. **Это означает, что ваш провайдер маскирует адреса, подменяя их своим внешним IP-адресом, запоминая, с какого компьютера был осуществлен какой запрос, и заменяя IP-адрес получателя со своего на ваш при прохождении ответа.** Обладая «серым» IP-адресом при использовании механизма трансляции адресов (NAT), вы можете обращаться к другим серверам, но внешние машины не смогут инициировать соединение к вашей машине как к серверу.

Следующий вопрос — как сервисы определяют, какое приложение на какой запрос должно реагировать. Ведь один и тот же сервер может отдавать и веб-страницы, и файлы по протоколу http, и почту по протоколу SMTP. Более того, если вы администрируете сервер, наверняка вам понадобится доступ к нему по протоколу VNC или ssh.

Соответственно, **протокол** — некий набор правил, который определяет, как то или иное приложение (сервер или клиент) будет взаимодействовать с аналогичным (клиентом или сервером) по сети. **Для веб-содержимого** используются протоколы **HTTP** и **HTTPS** (шифрованный HTTP), **для работы с файлами** — **FTP** и **FTPS** (шифрованный FTP), **для администрирования** — шифрованный **SSH** и **SFTP** (надстройка над SSH, реализующая схожий с FTP доступ к файлам).

Теперь осталось понять, каким образом сервер понимает, какому приложению следует отдать тот или иной пакет. Для этого используются порты. **Порт** — число от 0 до 65535, которое используется для идентификации приложения. Существует **65535 TCP-портов**, служащих для надежных соединений, и **65535 UDP-портов**, для которых надежное соединение не требуется. Одно приложение может использовать несколько портов. **Например**, веб-сервер обычно использует **80 TCP-порт** для установки незашифрованного соединения по протоколу **HTTP** и **443 TCP-порт** для установки шифрованного соединения по протоколу **HTTPS**. Для удаленного администрирования по протоколу **SSH (и его составной части SFTP)** используется **22 TCP-порт**. Соответственно, если вы купите VDS-сервер и будете его администрировать (с помощью клиента ssh в Linux или Mac OS X, либо с помощью PuTTY в Windows) **вам понадобится указать**: доменное имя (или IP-адрес) вашего сервера, номер порта, по которому запущен сервер ssh (обычно 22), ваши логин и пароль от операционной системы.

**Кстати, система DNS для преобразования доменных имен в IP-адреса (и не только) использует UDP-порт с номером 53.**

Мы рассмотрели упрощенно модель TCP/IP.

Она состоит всего из четырех уровней:

* **4 уровень** — прикладные протоколы **(DNS — 53 UDP-порт, HTTP — 80 TCP-порт, HTTPS — 443 TCP-порт, SSH и SFTP — 22 TCP-порт)**. Реализуют набор правил взаимодействия приложения-клиента и приложения-сервера.
* **3 уровень** — транспортные протоколы **(UDP — протоколы без подтверждений, TCP — протоколы с установкой соединения и надежной доставкой)**. Именно транспортные протоколы в заголовках содержат номера портов, позволяя идентифицировать приложения.
* **2 уровень** — сетевой — протокол IP. Заголовок содержит IP-адреса отправителя и получателя, позволяя идентифицировать машину-отправителя и машину-получателя.
* **1 уровень** — межсетевые протоколы — реализуют доступ к физической среде передачи информации. В качестве примеров можно привести Ethernet и Wi-Fi.

**IP-адрес (IPv4)**

- 4 октета, например 8.8.8.8, 5.255.255.5.

- Служат для идентификации хостов.

**Номер порта**

- Два типа портов: TCP и UDP.

- Двухбайтовое слово (от 0 до 65535).

- Служат для идентификации приложений.

**Протоколы**

Протокол – это набор правил (стандарт), который описывает взаимодействия функциональных блоков при передаче данных.

- Клиент и сервер.

- Порт отправителя и порт получателя.

- Динамические порты.

- Надежная и ненадежная доставка (UDP и TCP).

- Защищенные (SFTP, FTPS, SSH, HTTPS).

- Незащищенные (FTP, Telnet, HTTP).

- HTTP (не шифрованный) 80 TCP-порт.

- HTTPS (шифрованный) 443 TCP-порт.

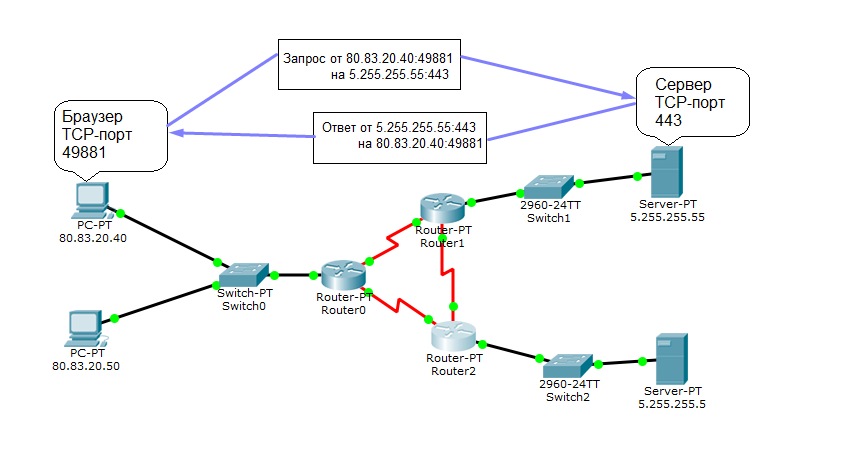
- DNS — 53 UDP-порт (используется также 53 TCP).

- DHCP.

**Адресация**

- IP-адрес идентифицирует хост (куда).

- Порт идентифицирует приложение (кому).



**Для чего нужна виртуальная машина?**

Виртуальные машины, используют так называемый **гипервизор**, работа которого заключается в том, чтобы транслировать ядро виртуальной машины в хостовую (главную) операционную систему.

Т.е. когда ядро у нас хочет получить доступ к какой-нибудь железяке и поскольку у нас это оборудование виртуальное, то там есть гипервизор, который выполняет эту трансляцию и передает запросы к хостовой системе.

**Виртуализация**

- Серверная виртуализация.

- Виртуализация на Desktop.

- Трансляция вызовов.

- Паравиртуализация.

- Аппаратная виртуализация.

- Контейнерная виртуализация.

**Серверная виртуализация**

- Xen

- KVM

- OpenVZ

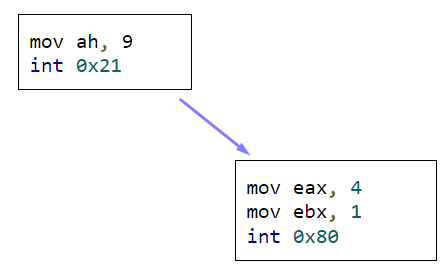
- LXC

**Десктопная виртуализация**

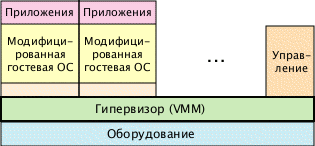
- VirtualBox

- VMWare Player

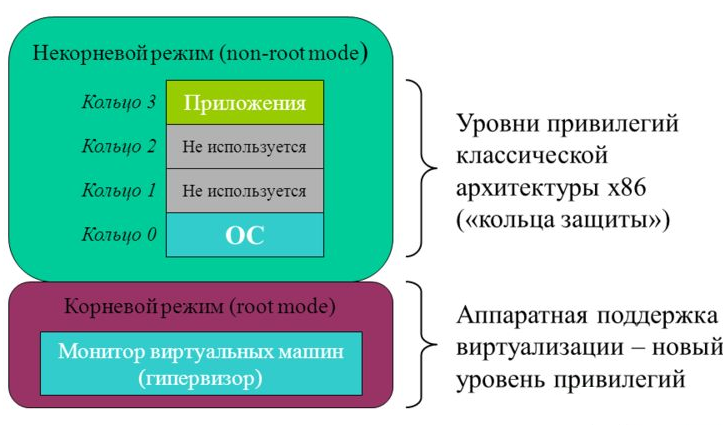
**Трансляция вызовов**



**Паравиртуализация**



**Аппаратная виртуализация**



**Контейнерная виртуализация**



**Файловая система в Linux**

- Нет дисков C:, D:, E:, F:.

- Все в одной иерархии — /home/user/Desktop.

- Диски монтируются в пустые директории /mnt/win\_disk\_d.

- Множество смонтированных виртуальных ФС.

- /proc — структуры ОС.

- /dev — устройства.

- /tmp — RAM-диск для временных файлов.

**Операции с носителями**

**Сами устройства имеют имена в /dev:**

- /dev/sda1

- /dev/sda2

- /dev/sdb1

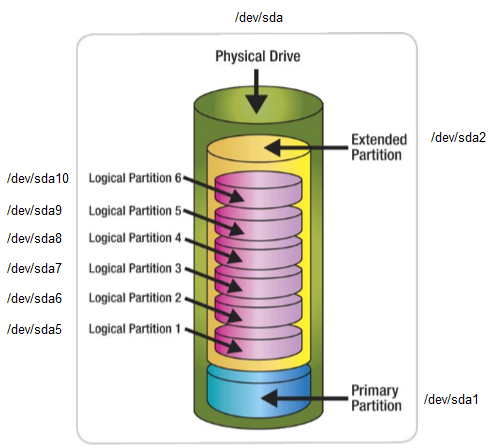
**Для доступа монтируются в пустые директории:**

- /dev/sda1 → /

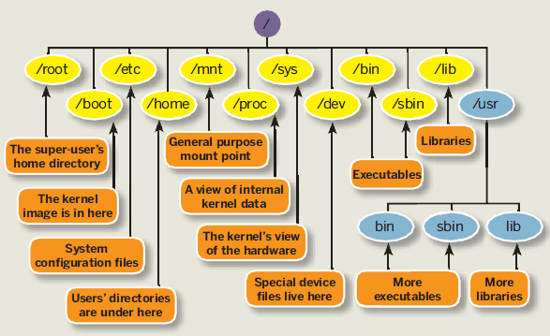
- /dev/sdb1 → /mnt/win\_disk\_d

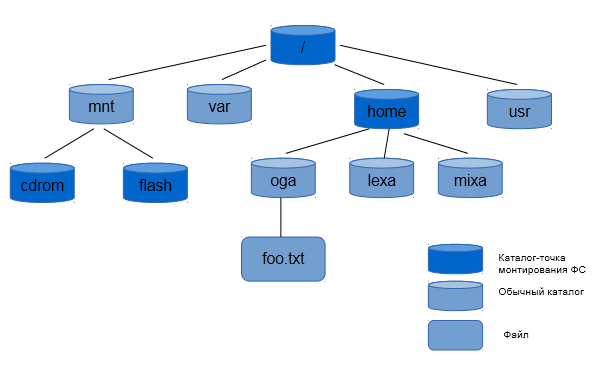
- /dev/sr0 → /cdrom

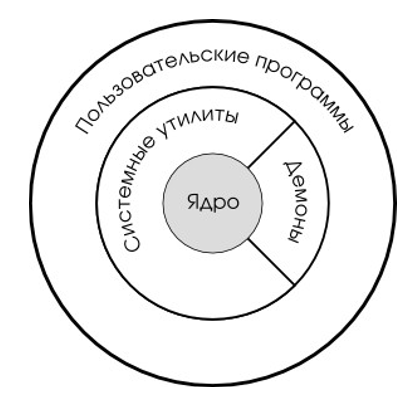
**Первичные, расширенные и логические разделы**

**Прим. В GPT есть**

**только первичные.**







**GNU/Linux сегодня**

- Ядро Linux.

- Окружение GNU.

- X-Server (реализация графической подсистемы).

- Systemd (система инициализации).

**Как работает X Windows System**

