Архитектура вычислительных систем

Лекция 9. Операционные системы

Artem Beresnev

t.me/ITSMDao

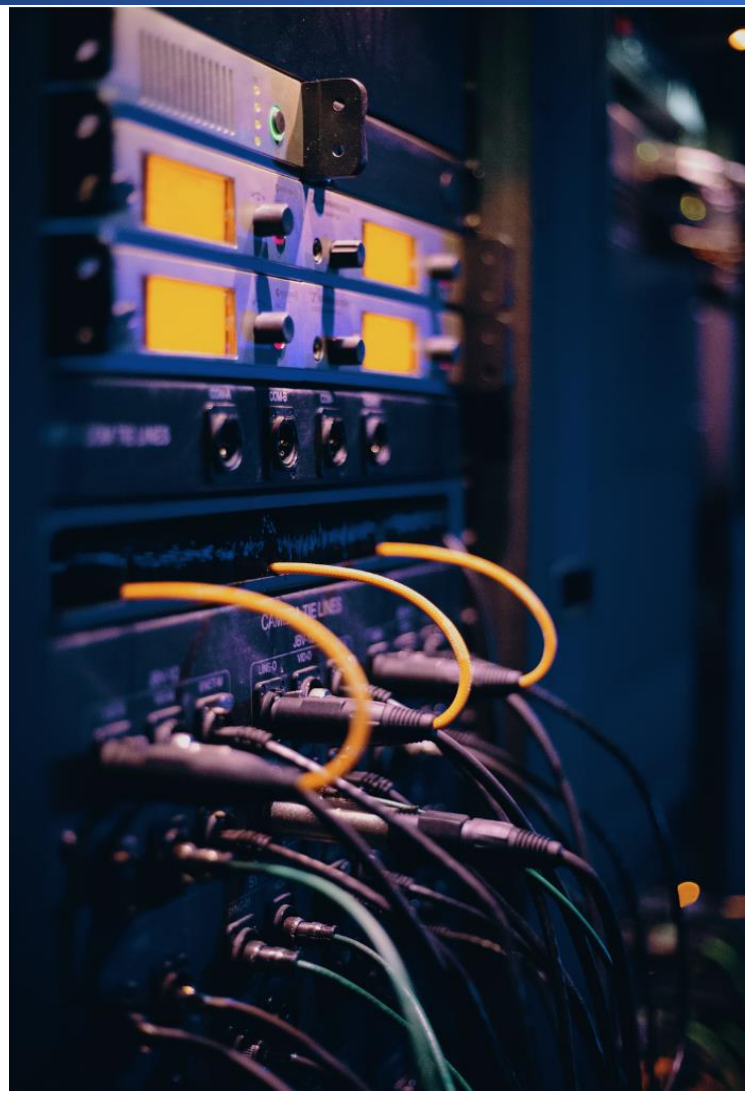
t.me/ITSMDaoChat

План

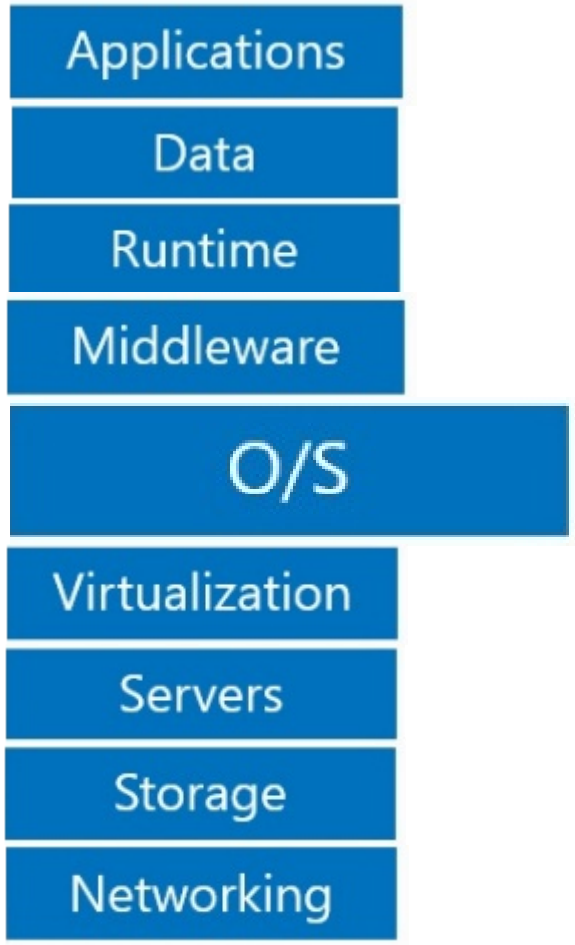
- Вспомним задачи слоя ОС
- Задачи ОС
- Обобщенная архитектура ОС
- Классификация
- Опять про Linux
- Ядро и дистрибутивы
- Загрузка ОС
- SYSTEMD

Слои IT-инфраструктуры

Что там было про OS?

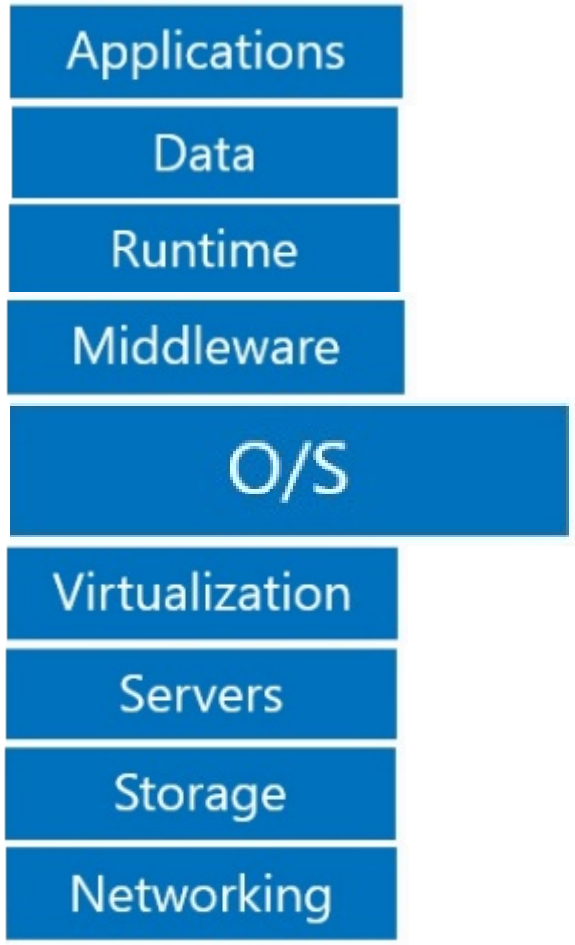


ИТ-инфраструктура



Операционные системы (operating system, OS) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.
Операционная система управляет аппаратными ресурсами и предоставляет сервисы для выполнения различных программ и приложений.
API.

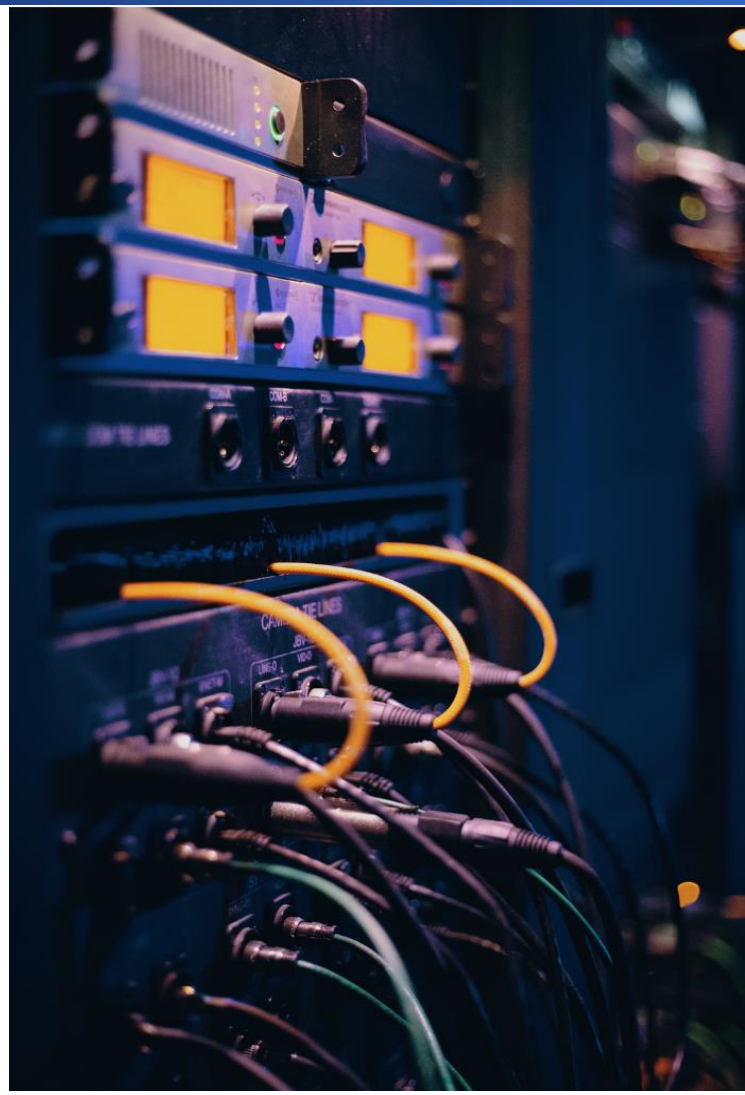
ИТ-инфраструктура



Примеры:
- Ubuntu
- Windows Server
- Red Hat Enterprise Linux
- Android

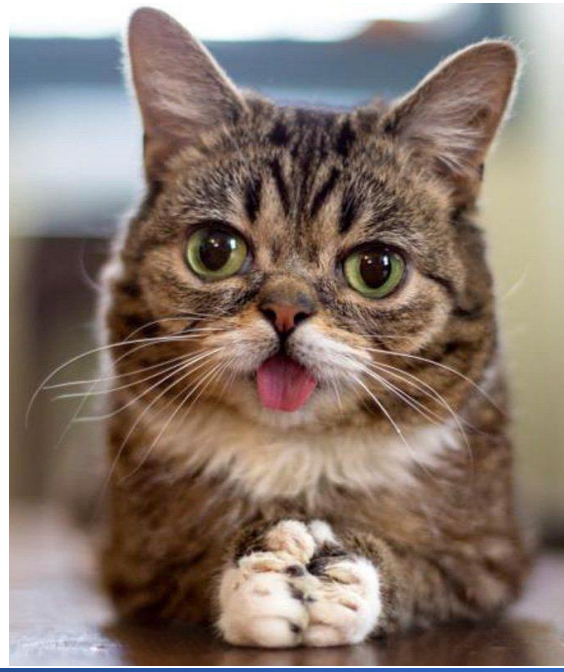
Зачем нужна ОС?

Какие задачи решает современная ОС? Почему нельзя работать без ОС? Или можно?



Задачи ОС

- Взаимодействие с оборудованием (Процессор, Прерывания, I/O, внешними устройствами)
- Управление ресурсами (процессами, памятью, файловыми системами)
- Защита данных и безопасность
- Сетевое взаимодействие
- API
- Предоставление пользовательского интерфейса

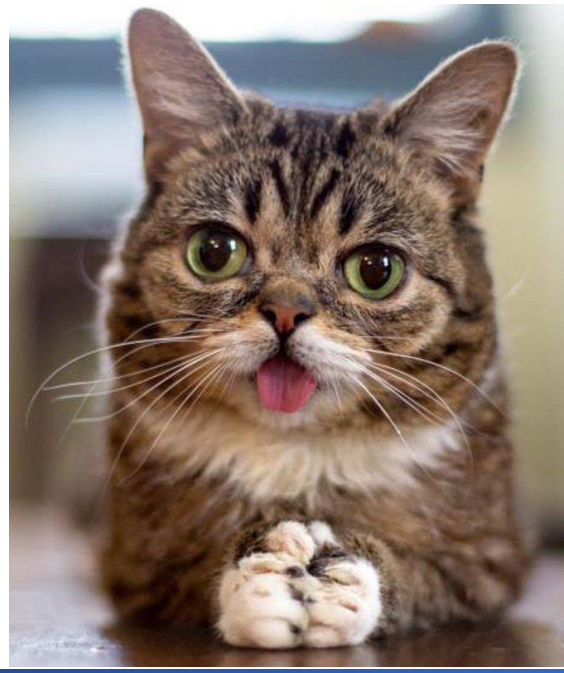


Можно без ОС

Можно, но в особых случаях.

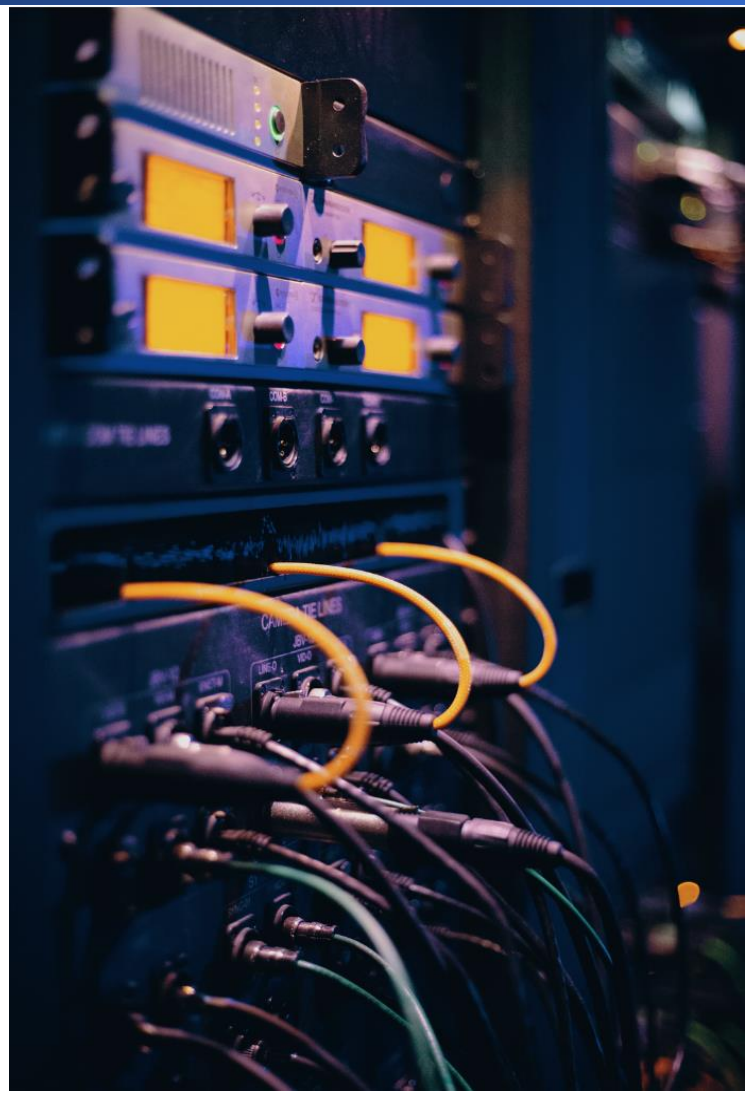
- Встраиваемые системы. Устройства на микроконтроллерах, работают без традиционной ОС. Специальное программное обеспечение, которое напрямую управляет аппаратными ресурсами.

- Сервисные программы BIOS/UEFI, которые работают без загрузки основной ОС.

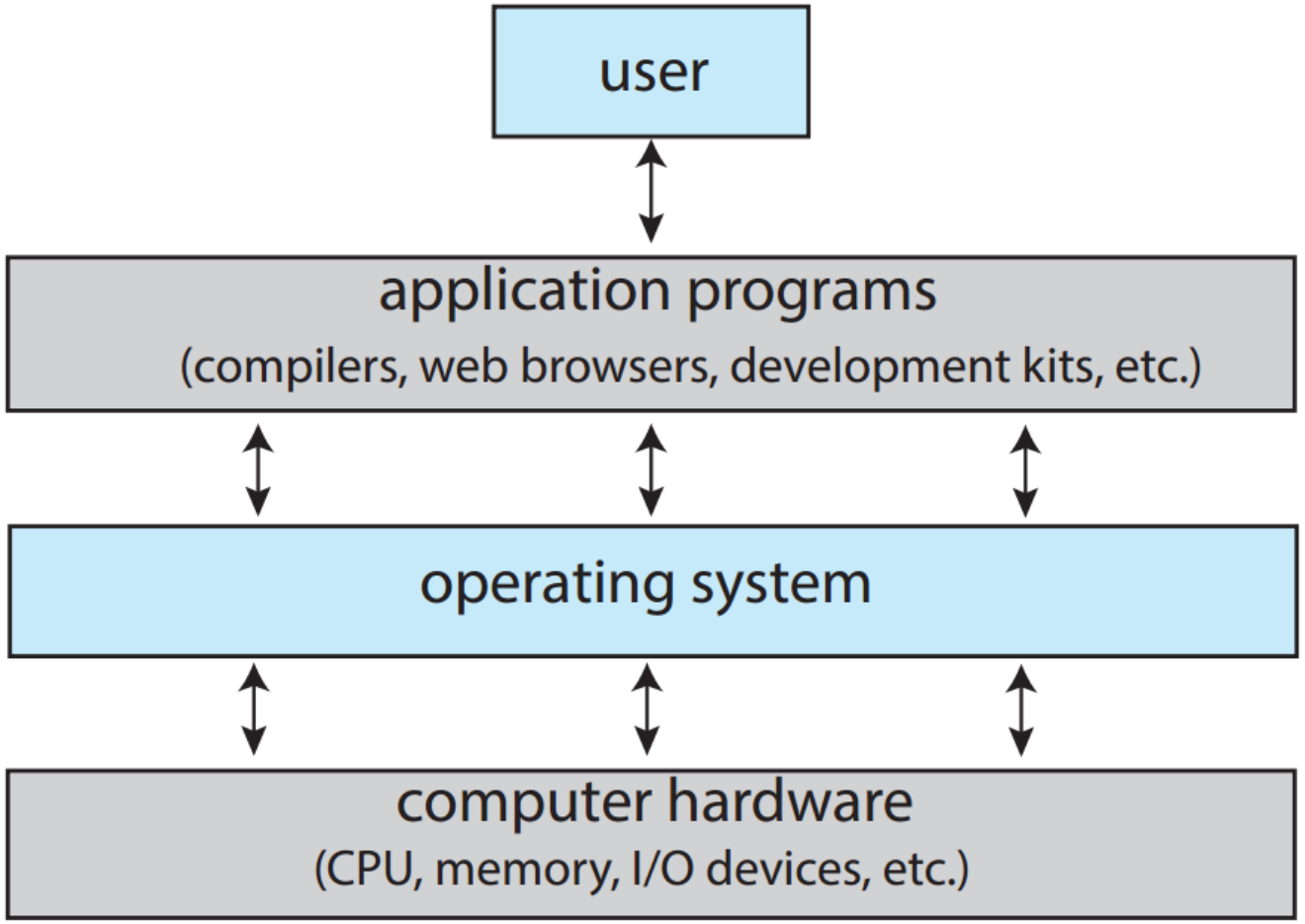


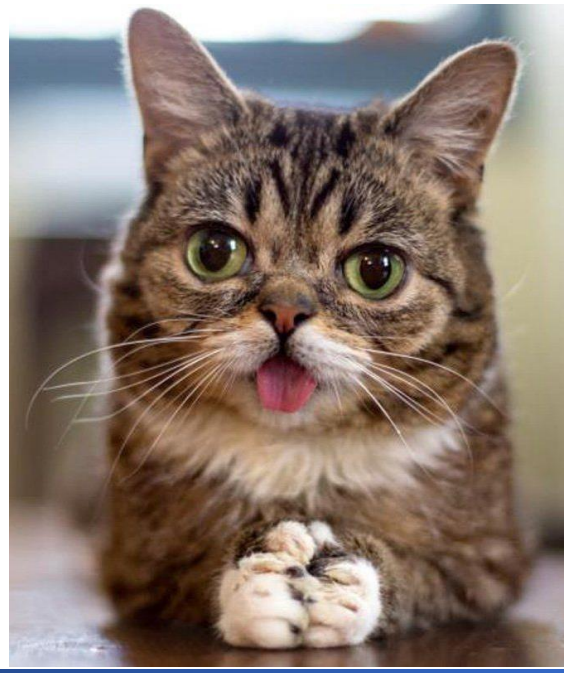
Обобщенная архитектура ОС

Из чего состоит современная усредненная ОС?



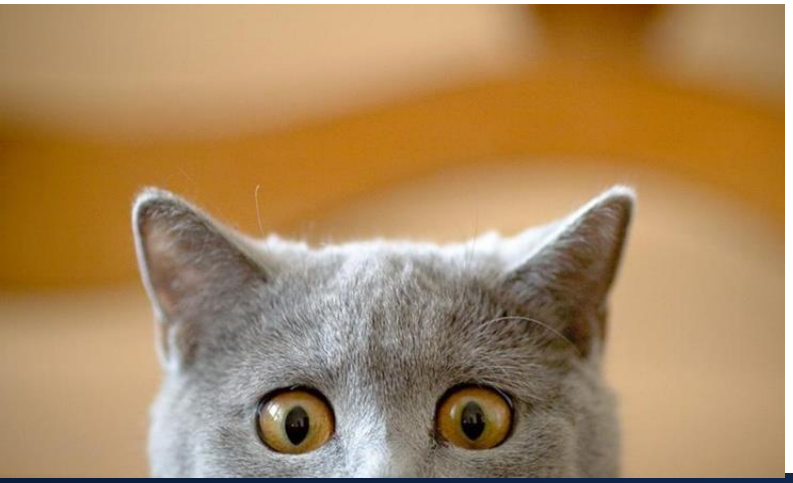
Обобщенная архитектура





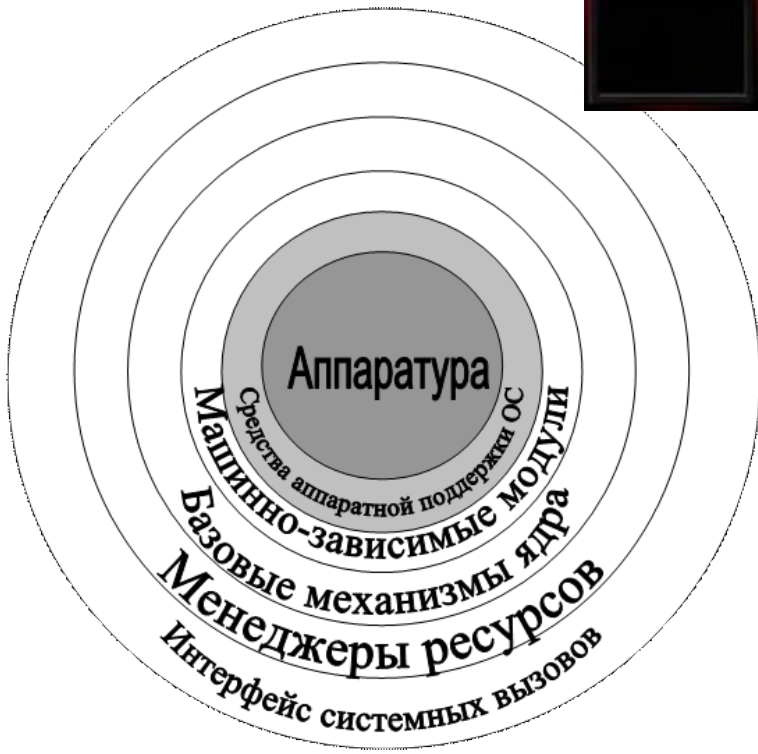
Типовая архитектура

- наличие аппаратных компонентов
- слои абстракции (опять???)
- распределение задач
- интерфейсы



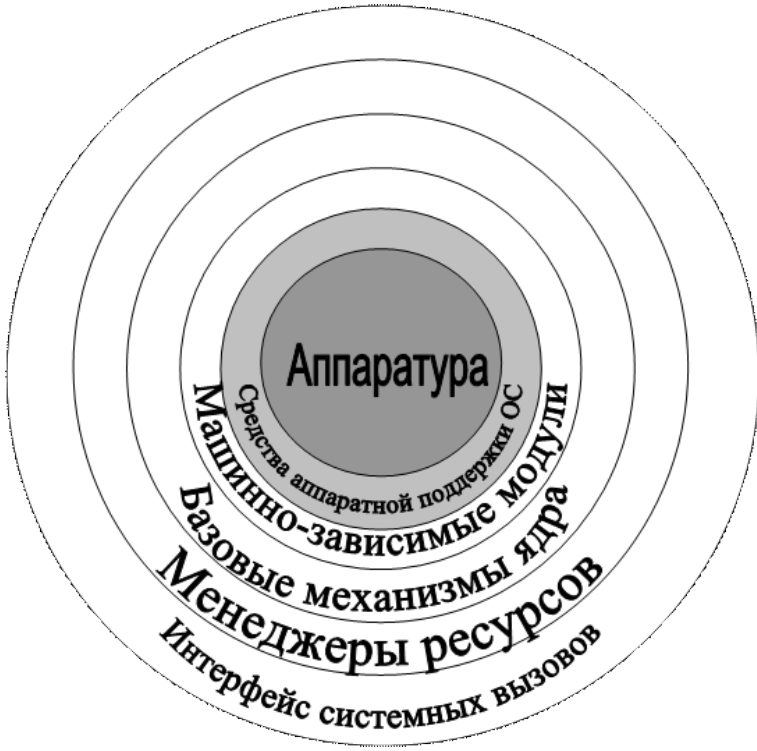
Проблемы сетевой коммуникации

- **Средства аппаратной поддержки** - средства поддержки привилегированного режима, распределение памяти, переключение контекстов процессора и т.п.
- **Машинно-зависимые компоненты –** прослойка, которая в пределе полное экранирование остальных компонентов ОС.
- **Базовые механизмы ядра** - выполнение наиболее примитивные операции ядра: как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно и т. п.



Проблемы сетевой коммуникации

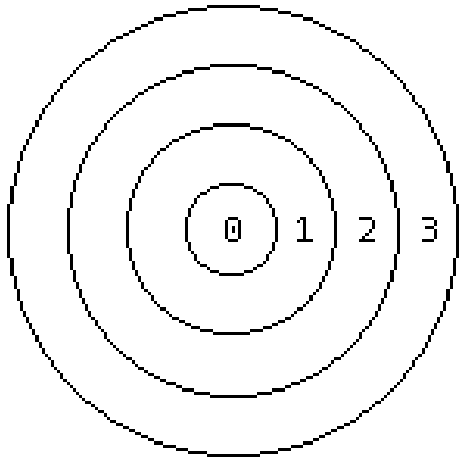
- **Менеджеры ресурсов** - модули, управляющие основными ресурсами вычислительной системы, например диспетчеры процессов, ввода-вывода, файловой системы и оперативной памяти.
- **Интерфейс системных вызовов** - самый верхний слой ядра, взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы. Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.



Зависимость от аппаратуры

- Причины аппаратной зависимости
  - **Архитектура и Набор команд**
  - Разрядность 16\32\64 в далекой перспективе 128
  - Использование аппаратных функций безопасности и распределения ресурсов
  - Одно-многопроцессорные системы
  - Реализация разграничения кода на аппаратном уровне
- Как обеспечить переносимость
  - Архитектура ОС
  - Перекомпиляция кода, как компонентов ОС, так и приложений

Аппаратная поддержка разграничения доступа кода



КОЛЬЦА ЗАЩИТЫ ПРОЦЕССОРА 80286 (И ВЫШЕ)

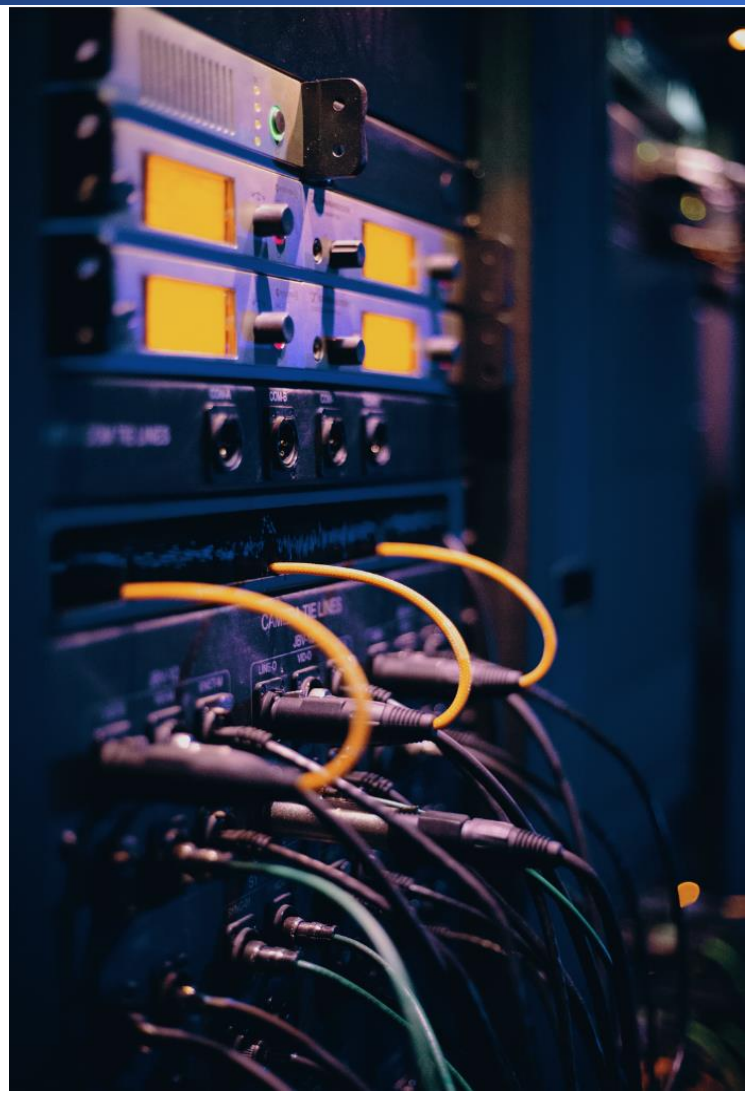
0 - Ядро операционной системы и драйвера;
1 - Оболочка операционной системы;
2 - Менее важные части операционной системы;
3 - Пользовательские приложения.

Нулевое кольцо имеет наивысшую степень защиты, а третье кольцо - наименьшую.

https://habr.com/ru/company/smart_soft/blog/184174/

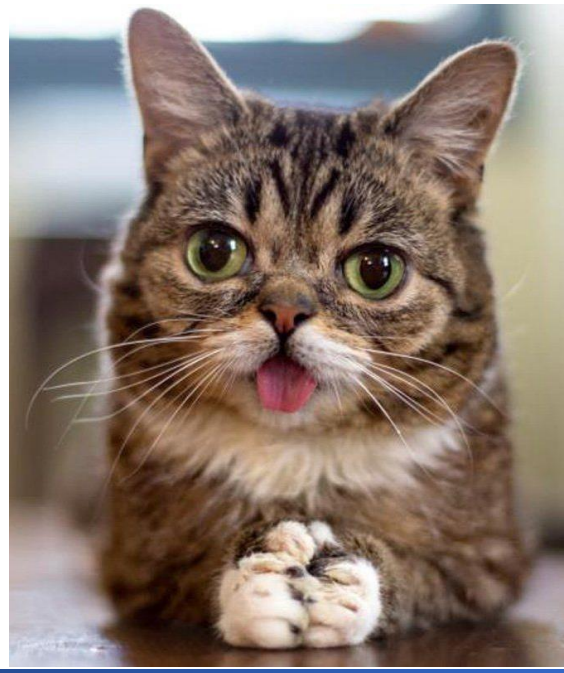
Классификация

А давайте отклассифицируем ОС

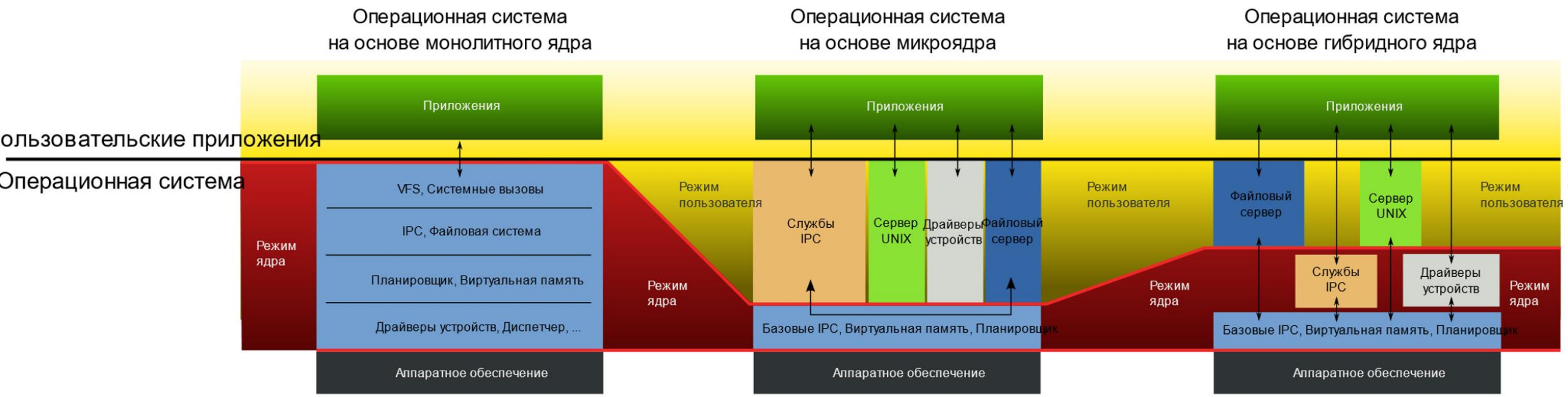


Классификация

- По архитектуре ядра
  - Микроядерные архитектуры
  - Архитектуры с полным ядром
  - Гибридные архитекторы
- По назначению
  - Пользовательские
  - Серверные
  - Мобильные
  - Встраиваемые
  - Общего назначения
- По времени обработки событий
  - Реального времени
  - Обычные
- Многопользовательские
- Сетевые
- По лицензированию
  - Проприетарные
  - «Свободные»

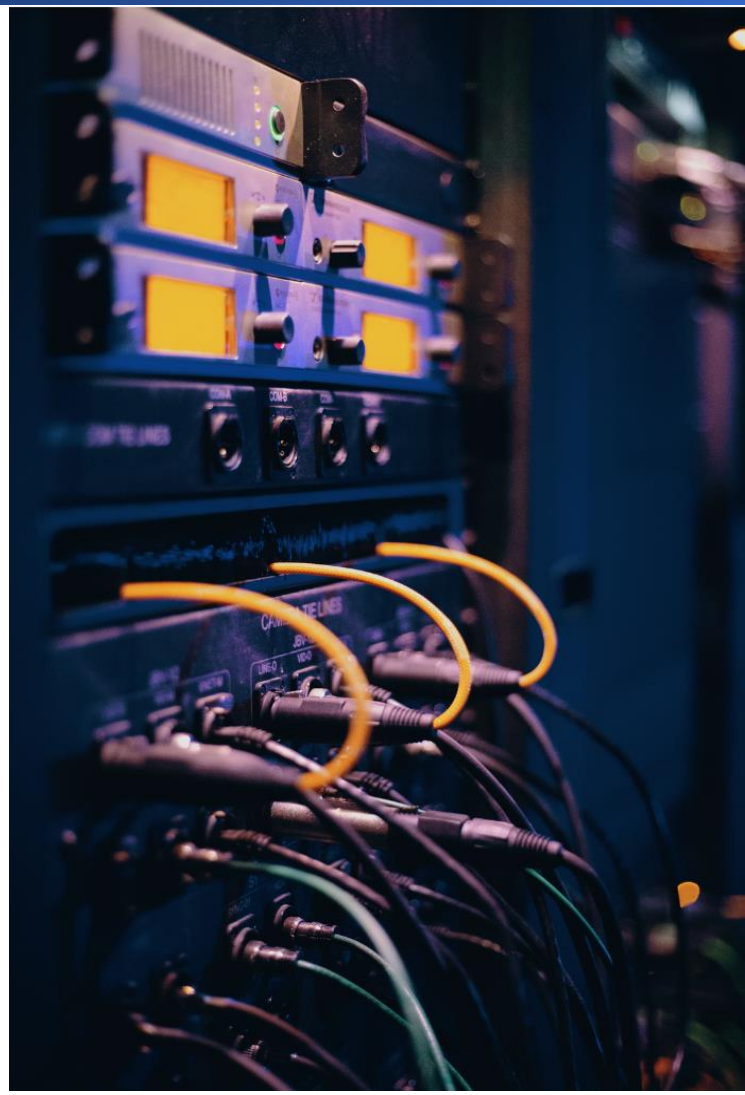


По архитектуре ядра



История Linux

Это очень поучительная история о счастливой случайности, чтении новостей и техническом творчестве

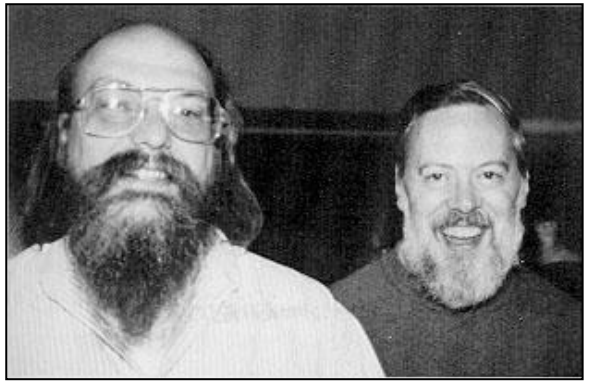


История ~~Linux~~ Unix

UNIX

- 1969 Bell Labs, Деннис Ричи, Кен Томпсон
- 1971 PDP-11 первая UNIX
- AT&T System V vs BSD

http://www.levenez.com/unix/history.html#22



Д. Ричи и К. Томпсон

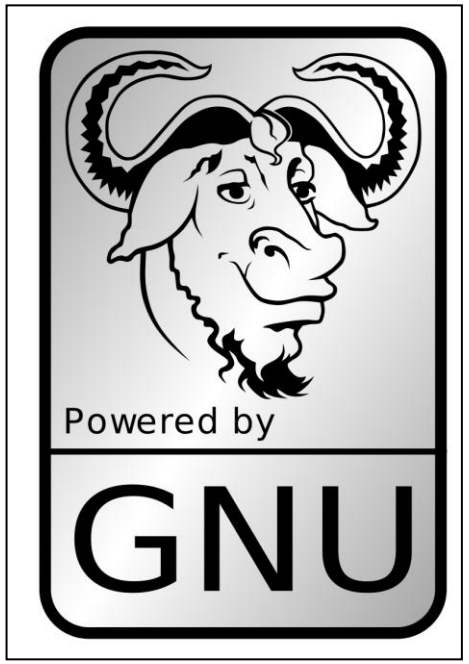
Основные идеи:

- Ключевые компоненты процесс и файл
- Компонентная архитектура – одна программа – одна функция
- Минимизация ядра
- Максимально абстрагирование от оборудования
- Все файл

PDP-7

GNU GNU’s Not UNIX

1983 основание проекта



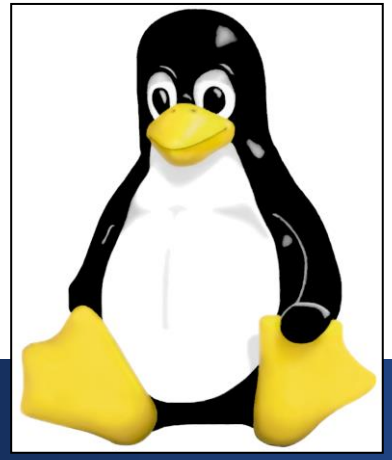
Ричард Столлмэн

GNU toolchain
GNU Emacs
GNU Linux

История Linux

- 1988 год Ларс Вирзениус, работая на компьютере с Ultrix опечатался и вместо rm набрал rn и попал в UseNet…
- Линус Торвальдс и Ларс Вирзениус проходят курс программирования на Си …
- 1990 Торвальдс покупает себе комп ☺ (386 / Minix) и начинает писать собственный терминал…
- 25 августа 1991 года – первое сообщение от Торвальдса о бесплатной ОС.
- 17 сентября 1991 / Linux 0.0.1 / POSIX
- FREAX или Linux?
- Март 1994 Linux 1.0 под лицензией GNU

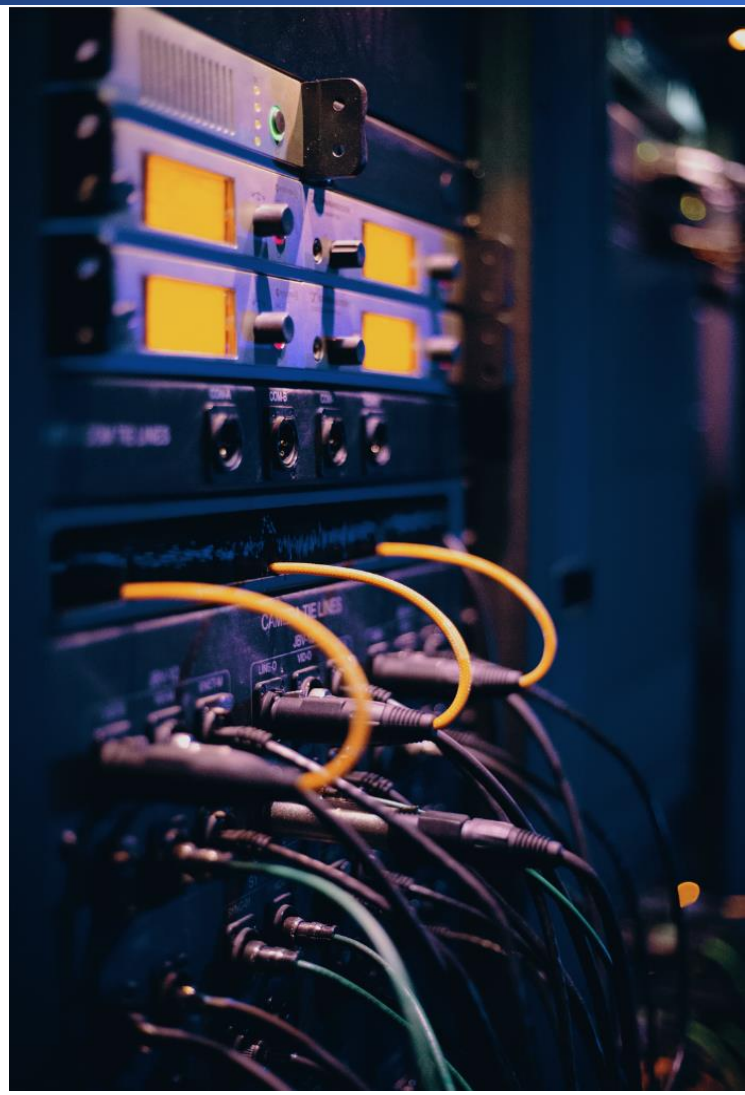
Линус Бенедикт Торвальдс



1996 Ларри Ирвинг

Что же такое Linux?

Что такое Linux? А что такое дистрибутив Linux? В чем разница?



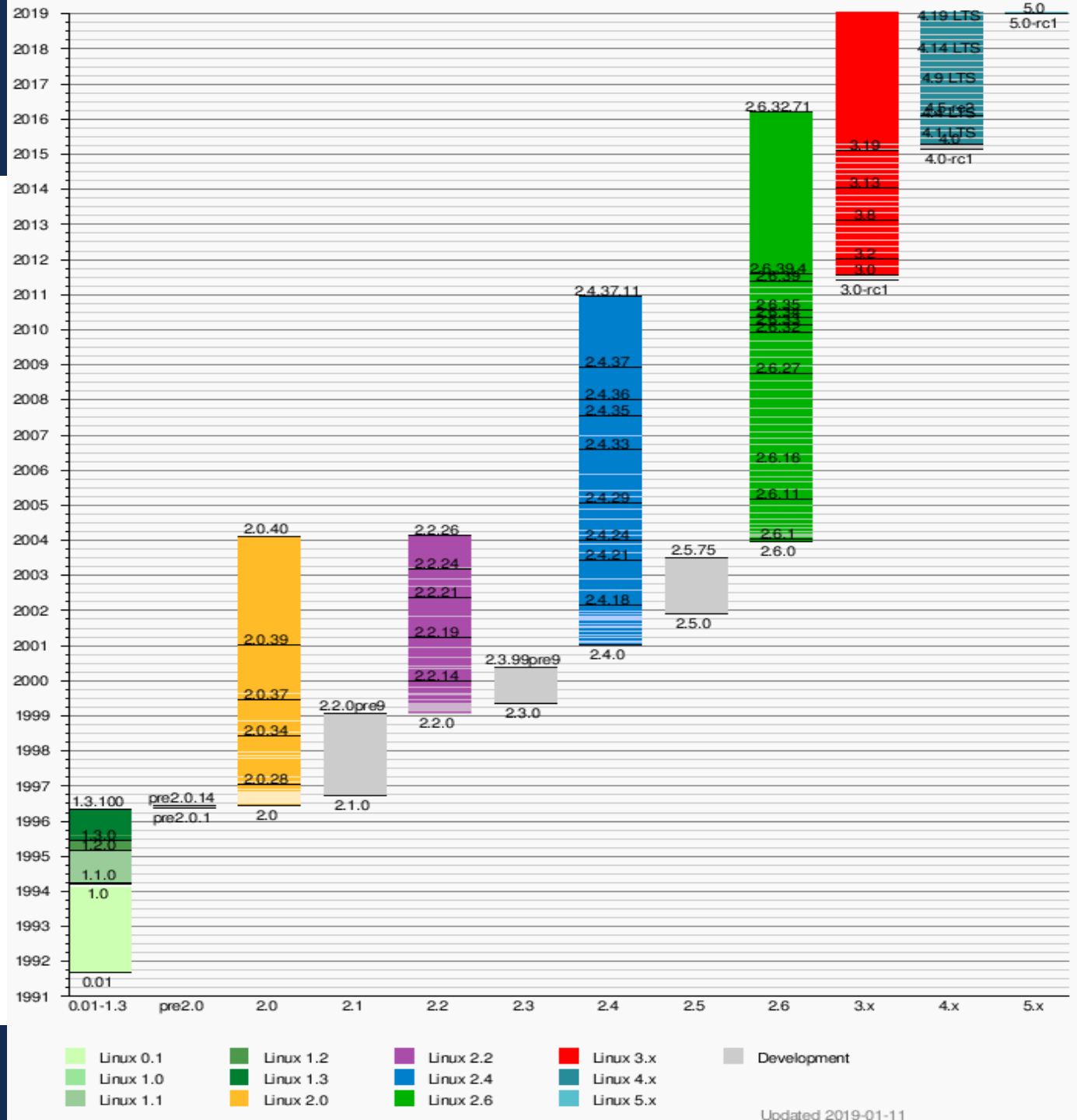
Linux – это ядро ОС

- Linux – это ядро!
- uname
- /boot/vm*
- www.kernel.org
- Управление: Линус Торвальдс – Ответственные за версию ядра – Мейнтейнеры модулей – Сообщество
- Тоталитарная демократия
- Любые изменения проходят монго уровней тестирования. Сообщество активно участвует в этом процессе, выявляя и исправляя ошибки.
- Ядро Linux выпускается в виде стабильных версий каждые несколько месяцев. Между основными версиями существуют тестовые (релиз-кандидаты), которые помогают выявить и устранить ошибки до выхода финальной версии.
- Система контроля версий - Git.
- Кто за все платит? Linux Foundation

Версии ядра Linux

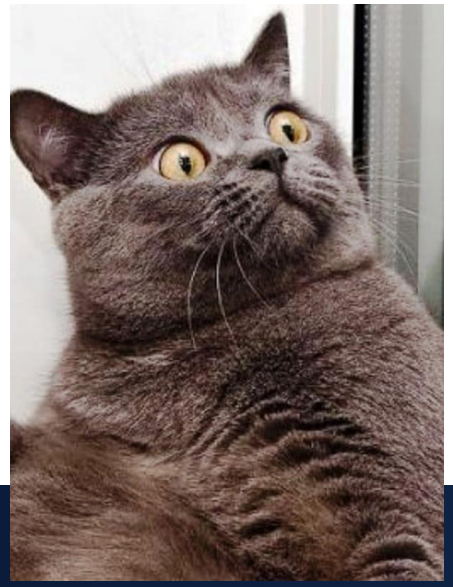
- https://en.wikipedia.org/wiki/Linux_kernel_version_history

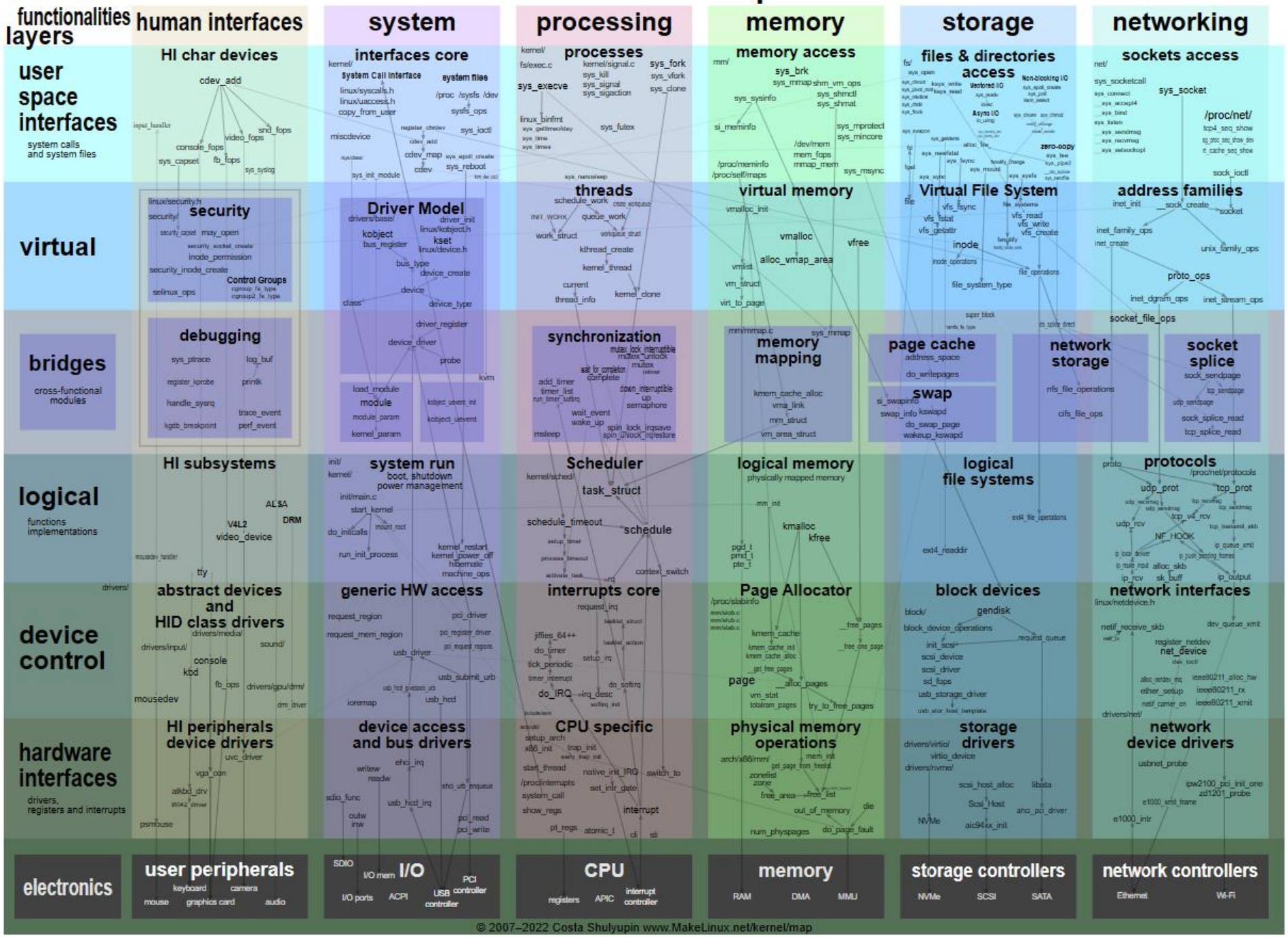
- A.B.C[.D]
- A – версия ядра
- B – старшая версия ревизии ядра (чет\нечет до 3-й версии)
- C – младшая версия ревизии ядра
- D – версия исправления «смертельной ошибки»



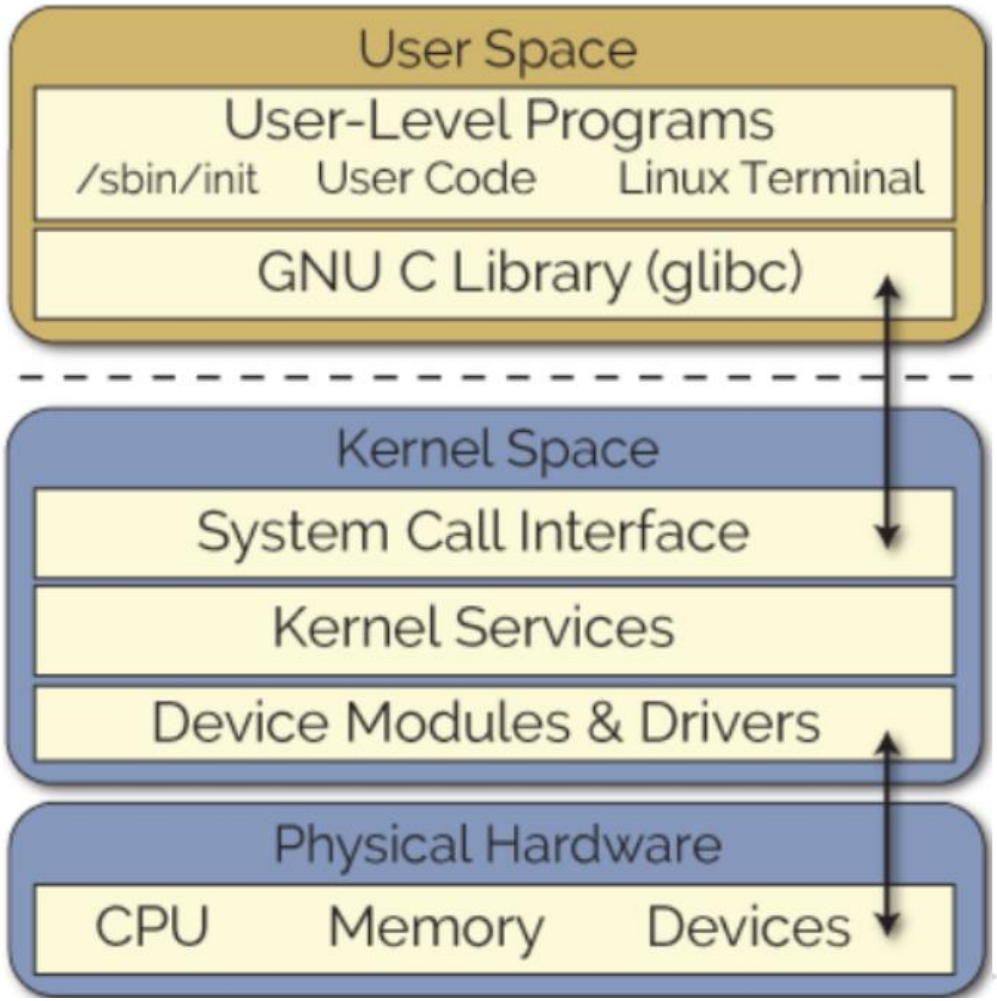
Ядро

https://makelinux.github.io/kernel/map/





Архитектура ОС



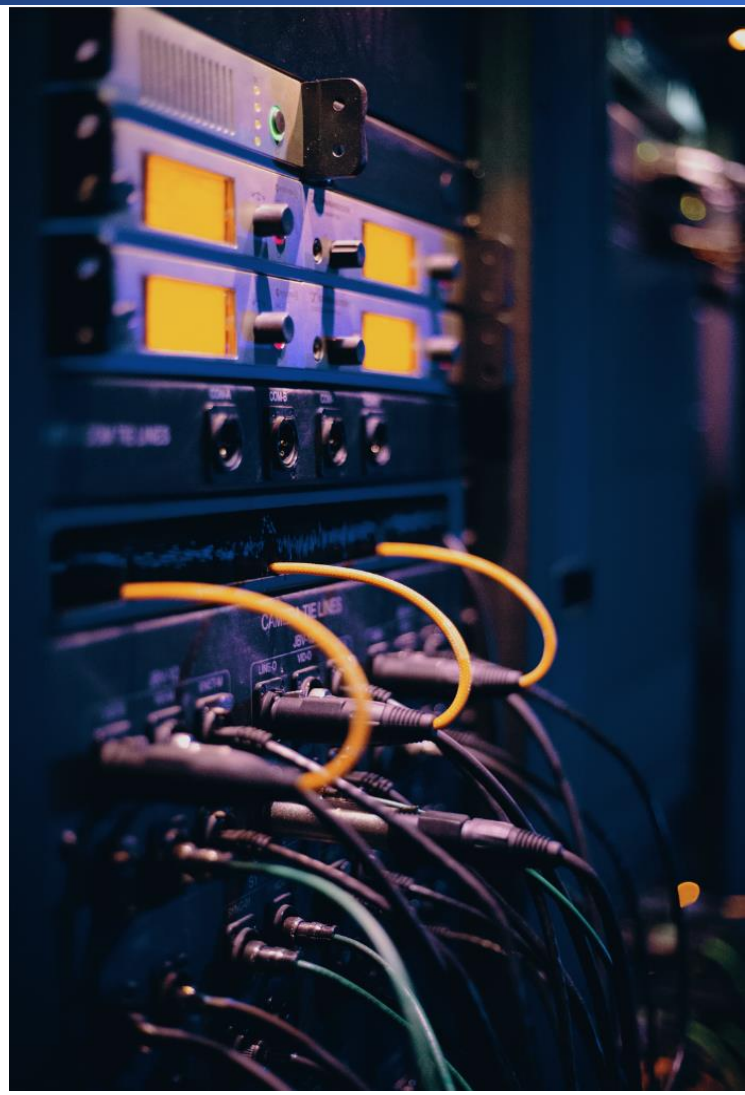
- **Программы user-space** - библиотеки, утилиты, прикладное ПО, службы(сервисы, демоны).
- **Glibc** предоставляет реализацию многих стандартных функций языка C, описанных в стандартах ISO C и POSIX. Это включает в себя обработку ввода-вывода, работу с файлами, управление памятью, математические операции, строковые функции и многое другое. Обеспечивает переносимость.
- **Системные вызовы** - выполняются в ядре - функции операционной системы, реализуемые ядерными компонентами и доступные внеядерным компонентам
- **Монолитное ядро**, но с загружаемыми модулями! Разделяемые системные библиотеки (system libraries) содержат стандартный набор функций, используемых приложениями для запросов к системным сервисам ядра.

Что в ядре?

Содержит: **Process Scheduler** - планировщик процессов (делит CPU), **Memory Manager** - система управления памятью - формирует из оперативной памяти и свопа виртуальную память, делит ее на kernel и user space, **Inter-Process Communication** - механизмы межпроцессного взаимодействия, **Virtual File System** - виртуальная файловая система, **Network Interface** - сетевая подсистема - реализует концепции сетевых интерфейсов и стеков сетевых протоколов

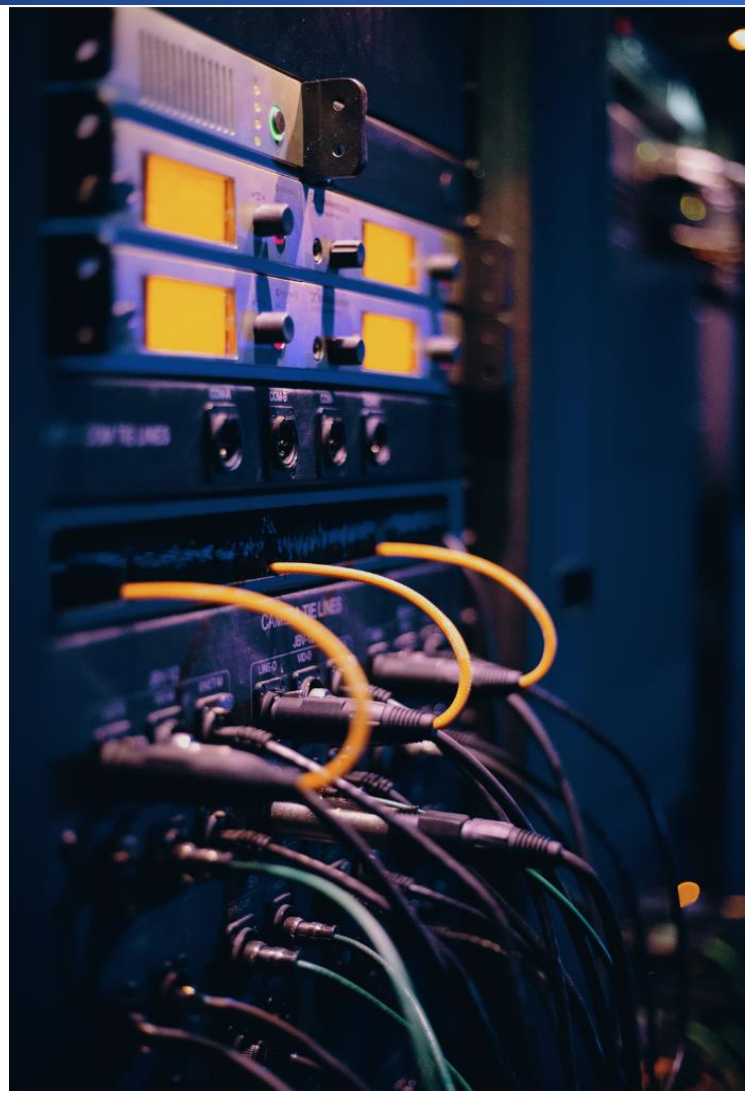
Что же такое дистрибутив Linux?

Что такое дистрибутив Linux? Почему их так много?



Дистрибутив Linux

Как запускается Linux?



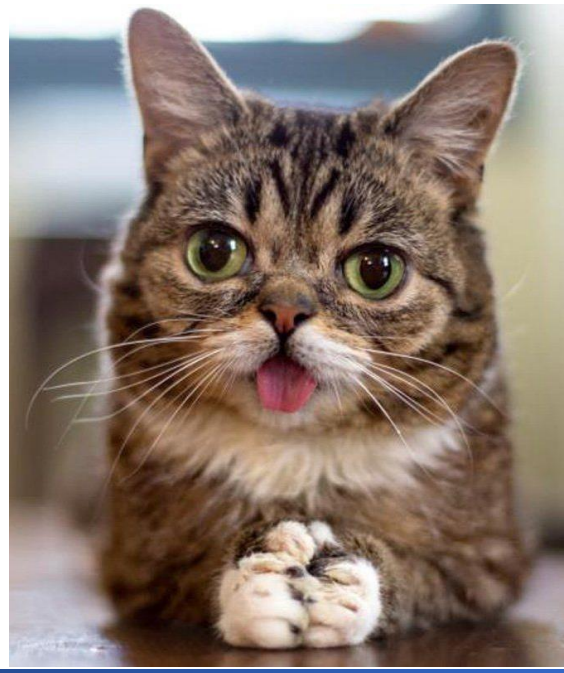
Как запускается Linux?

Этапы загрузки системы:

- BIOS POST
- Работа загрузчика (GRUB2)
- Загрузка ядра
- Запуск инициализатора

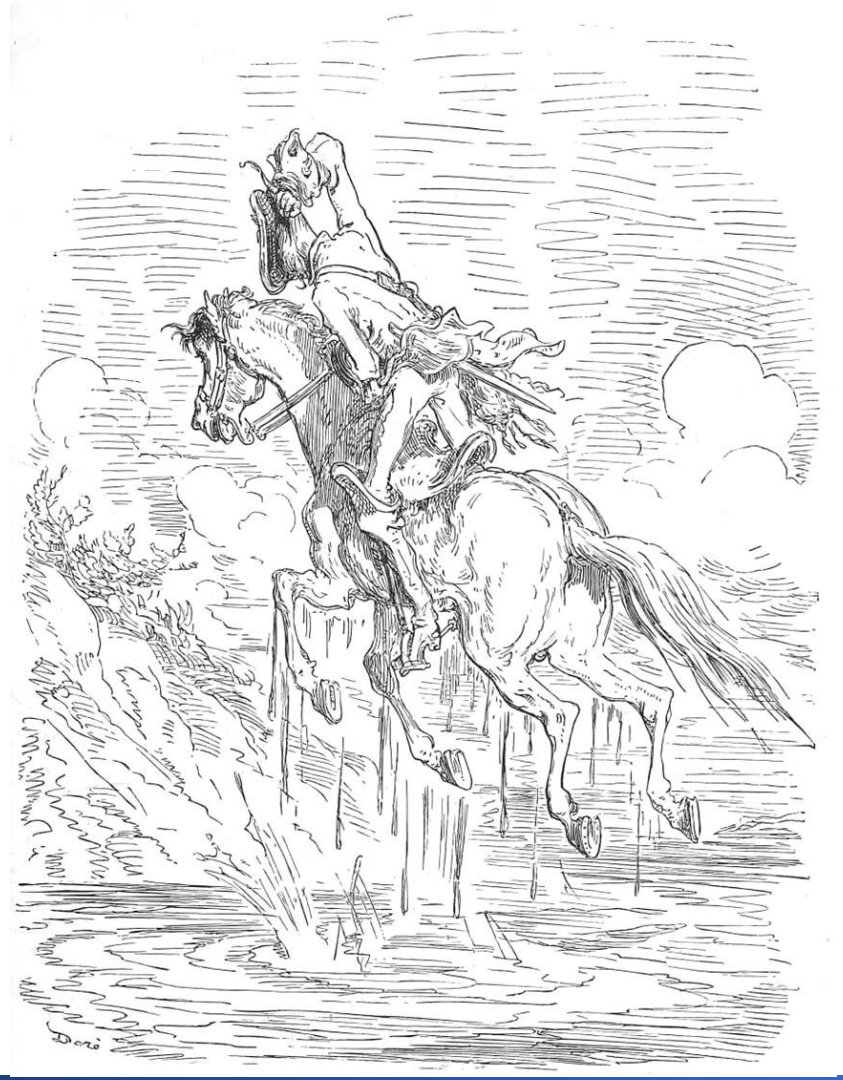
BIOS и Загрузчик

- BIOS POST
  - проверяет аппаратуру
  - читает с загрузочного устройства первый сектор
  - передает управление считанному коду
- Загрузка ядра (GRUB2)
  - Позволяет выбрать ядро
  - Загружает драйвер файловой системы
  - Загружает выбранное ядро (/boot/vmlinuz)



Ядро и systemd

- Ядра
  - Монтирование /boot/initramfs в /tmpfs как первичную файловую систему
  - запускает systemd
- Systemd
  - Монтирует остальные файловые системы
  - Выбирает target (~ветку загрузки)
  - Загружает системные модули и службы

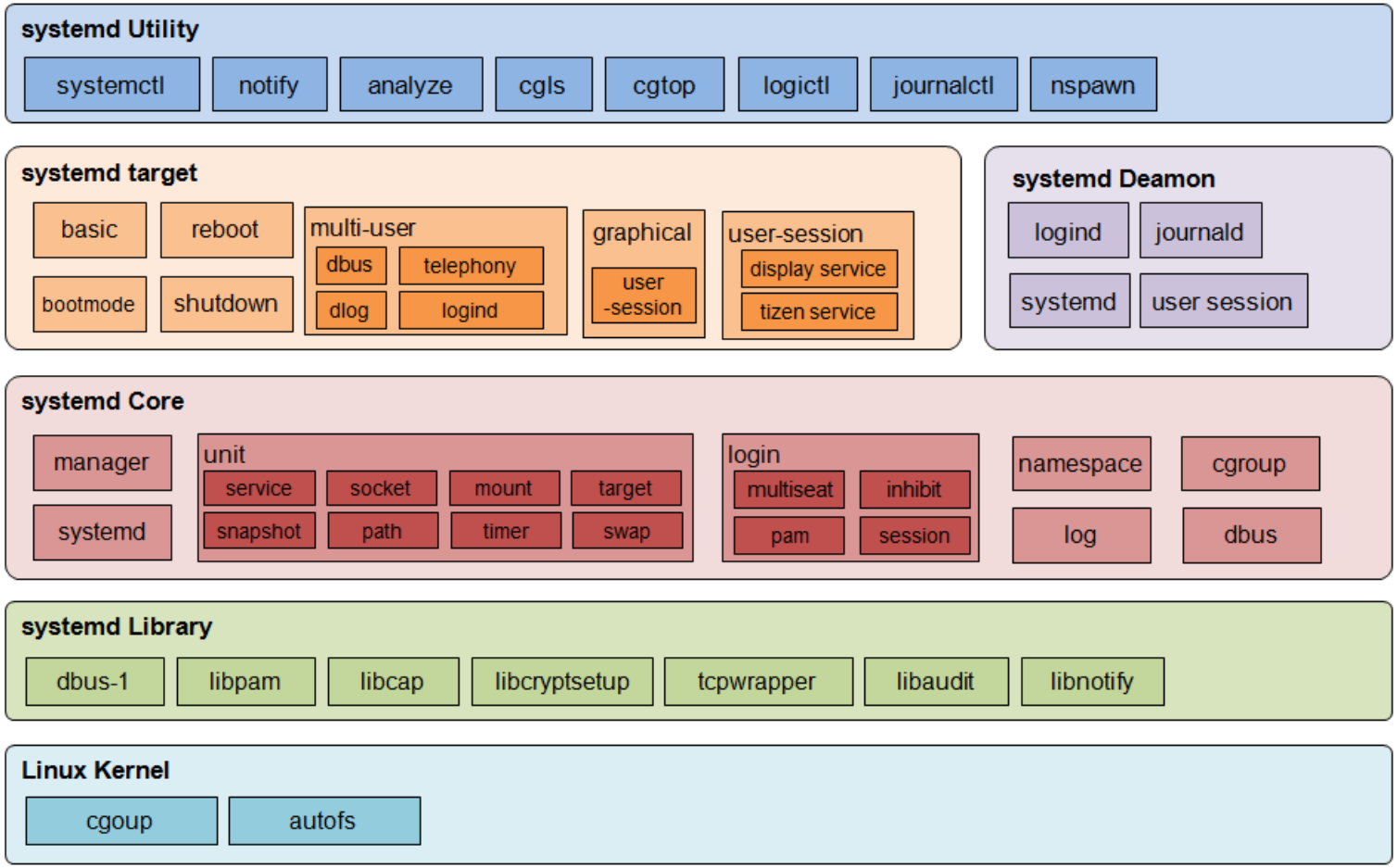


Что такое systemd?

- Systemd подсистема инициализации и управления службами в Linux
- создана Леннартом Пёттерингом и Кеем Сиверсом
- заменила init в большинстве дистрибутивов
- это система больших, скомпилированных исполняемых файлов

Что такое systemd?

systemd — подсистема инициализации и управления службами в Linux



Unit в systemd

Служба в Systemd описывается файлом юнита. Существуют такие типы служб:

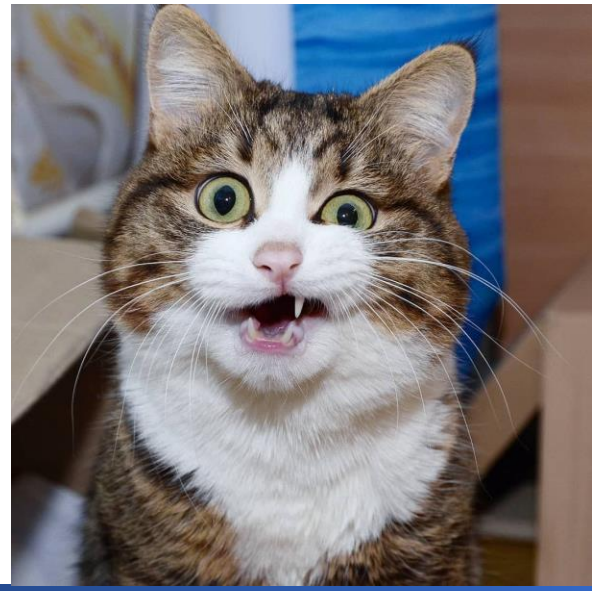
- **service - обычная служба, программа**
- target - группа служб
- automount - точка автоматического монтирования
- device - файл устройства, генерируется на этапе загрузки
- mount - точка монтирования
- path - файл или папка
- scope - процесс
- slice - группа системных служб systemd
- snapshot - сохраненное состояние запущенных служб
- socket - сокет для взаимодействия между процессами.

Каталоги systemd

- /usr/lib/systemd/system - системные юниты, поставляемые обычно вместе с приложениями;
- /run/systemd/system - динамически создаваемые юниты (т.е. на лету);
- /etc/systemd/system - юниты и исправления, внесённые администратором

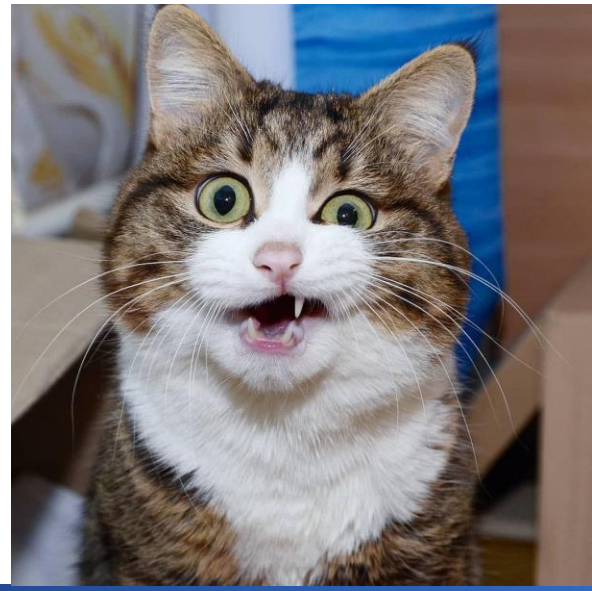
Утилиты systemd

- **systemctl**
  - systemctl get-default или systemctl set-default
  - systemctl list-units --type service
  - systemctl list-units --type service --state failed
  - systemctl list-unit-files
  - systemctl stop application (start status)
  - systemctl disabled application (enabled is-enabled)
  - systemctl mask firewalld (unmask)
  - systemctl cat mask firewalld
  - systemctl poweroff
  - systemctl halt \ systemctl reboot \ systemctl rescue
  - systemctl daemon-reload
  - systemctl kill
  - И т.д.

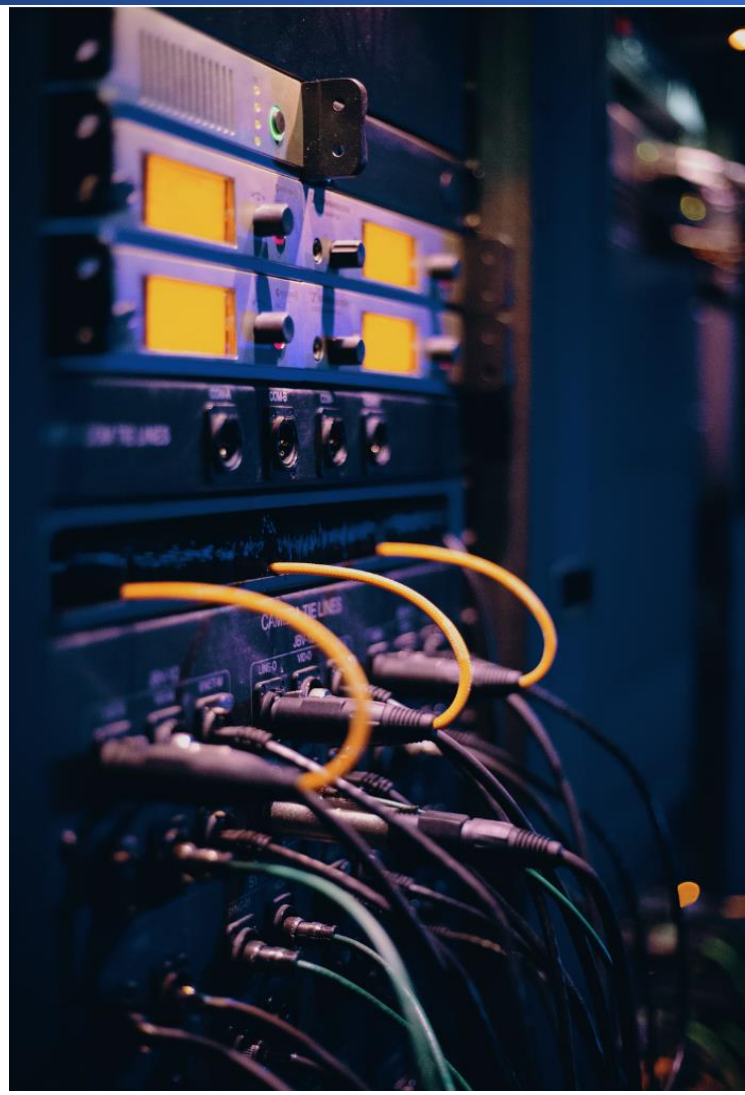


Утилиты systemd

- systemd-analyze (текущие данные и анализ)
  - systemd-analyze
  - systemd-analyze blame
- journalctl (/run/log/journal/<machine-id>)
  - journalctl
  - journalctl -b
  - journalctl --list-boots
  - journalctl -u sshd *или* journalctl -f -u sshd
  - journalctl --disk-usage
  - journalctl --vacuum-size=1G
  - И т.д



Итоги



Итоги

- Вспомним задачи слоя ОС
- Задачи ОС
- Обобщенная архитектура ОС
- Классификация
- Опять про Linux
- Ядро и дистрибутивы
- Загрузка ОС
- SYSTEMD