



Системное программное обеспечение

Лекция 1. Операционные системы

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ

Что такое операционная система

Операцио́нная систéма, сокр. ОС (operating system, OS) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

В логической структуре типичной вычислительной системы операционная система занимает положение между устройствами с их микроархитектурой, машинным языком и, возможно, собственными (встроенными) микропрограммами — с одной стороны — и прикладными программами с другой.

Разработчикам программного обеспечения операционная система позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя минимально необходимый набор функций (см.: интерфейс программирования приложений).

Основные функции ОС

- Исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
- Загрузка программ в оперативную память и их выполнение.
- Стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).
- Управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
- Управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, оптические диски и др.), организованным в той или иной файловой системе.
- Обеспечение пользовательского интерфейса.
- Сохранение информации об ошибках системы.

Дополнительные функции ОС

- Параллельное или псевдопараллельное выполнение задач (многозадачность).
- Эффективное распределение ресурсов вычислительной системы между процессами.
- Разграничение доступа различных процессов к ресурсам.
- Организация надёжных вычислений (невозможности одного вычислительного процесса намеренно или по ошибке повлиять на вычисления в другом процессе), основана на разграничении доступа к ресурсам.
- Взаимодействие между процессами: обмен данными, взаимная синхронизация.
- Защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений.
- Многопользовательский режим работы и разграничение прав доступа (см.: аутентификация, авторизация).

Когда необходима ОС:

- если нужен универсальный механизм сохранения данных
- для предоставления системным библиотекам часто используемых подпрограмм
- для распределения полномочий
- необходима возможность имитации «одновременного» исполнения нескольких программ на одном компьютере
- для управления процессами выполнения отдельных программ

Когда необходима ОС:

Таким образом, современные универсальные операционные системы можно охарактеризовать, прежде всего, как:

- использующие файловые системы (с универсальным механизмом доступа к данным),
- многопользовательские (с разделением полномочий),
- многозадачные (с разделением времени).

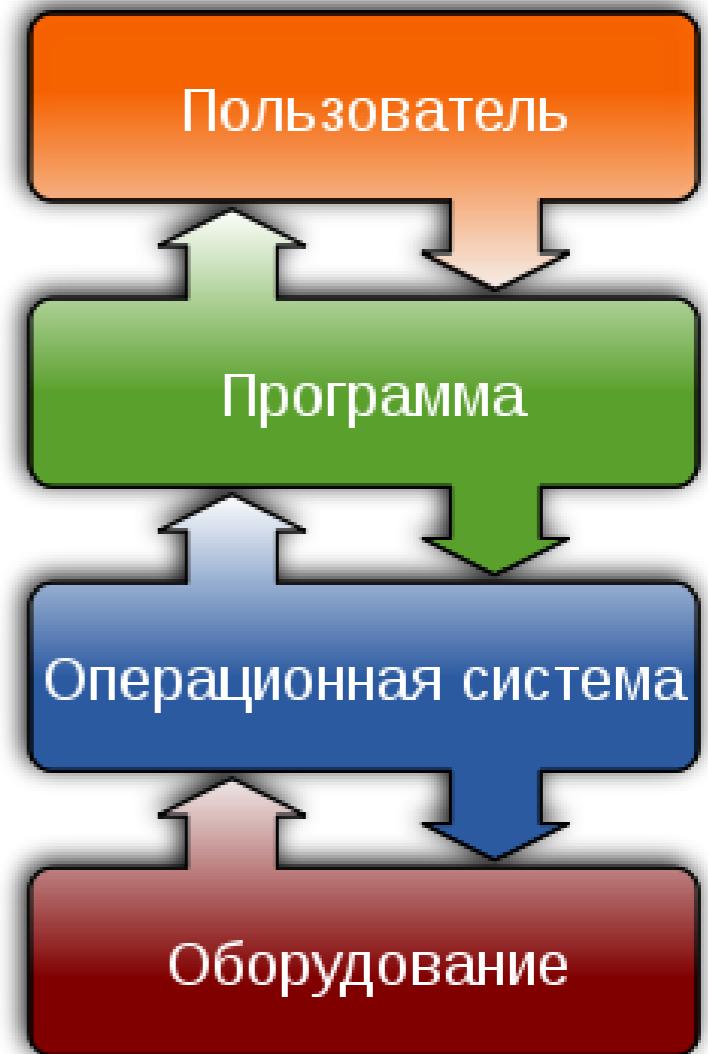
Применение универсальных компьютеров для управления производственными процессами потребовало реализации «масштаба реального времени» («реального времени») — синхронизации исполнения программ с внешними физическими процессами.

Из чего состоит ОС:

- Загрузчик ОС
- Ядро ОС
- Командный процессор
- Драйверы устройств
- Встроенное ПО

Компоненты ОС (4 уровня):

- Ядро ОС + драйвера + загрузчик + сетевая подсистема + файловая подсистема
- Системная библиотека + API к ядру (системные вызовы)
- Утилиты + оболочка
- Дополнительные библиотеки + User-oriented ПО



Файловая система

Файловая система — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах.

Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов.

Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

Однако файловая система не обязательно напрямую связана с физическим носителем информации. Существуют виртуальные файловые системы, а также сетевые файловые системы, которые являются лишь способом доступа к файлам, находящимся на удалённом компьютере.

Файловая система

Для носителей с произвольным доступом, (жёсткий диск): FAT32, HPFS, ext2 и др.

Поскольку доступ к дискам в несколько раз медленнее, чем доступ к оперативной памяти, для прироста производительности во многих файловых системах применяется асинхронная запись изменений на диск. Для этого применяется либо журналирование (хранение промежуточных состояний), например в ext3, NTFS

Файловая система

Для носителей с последовательным доступом (магнитные ленты): QIC и др.

Для оптических носителей — CD и DVD: ISO9660, HFS, UDF и др.

Виртуальные файловые системы: AEFS и др.

Сетевые файловые системы: NFS, CIFS, SSHFS, GmailFS и др.

Для флэш-памяти: YAFFS, ExtremeFFS, exFAT.

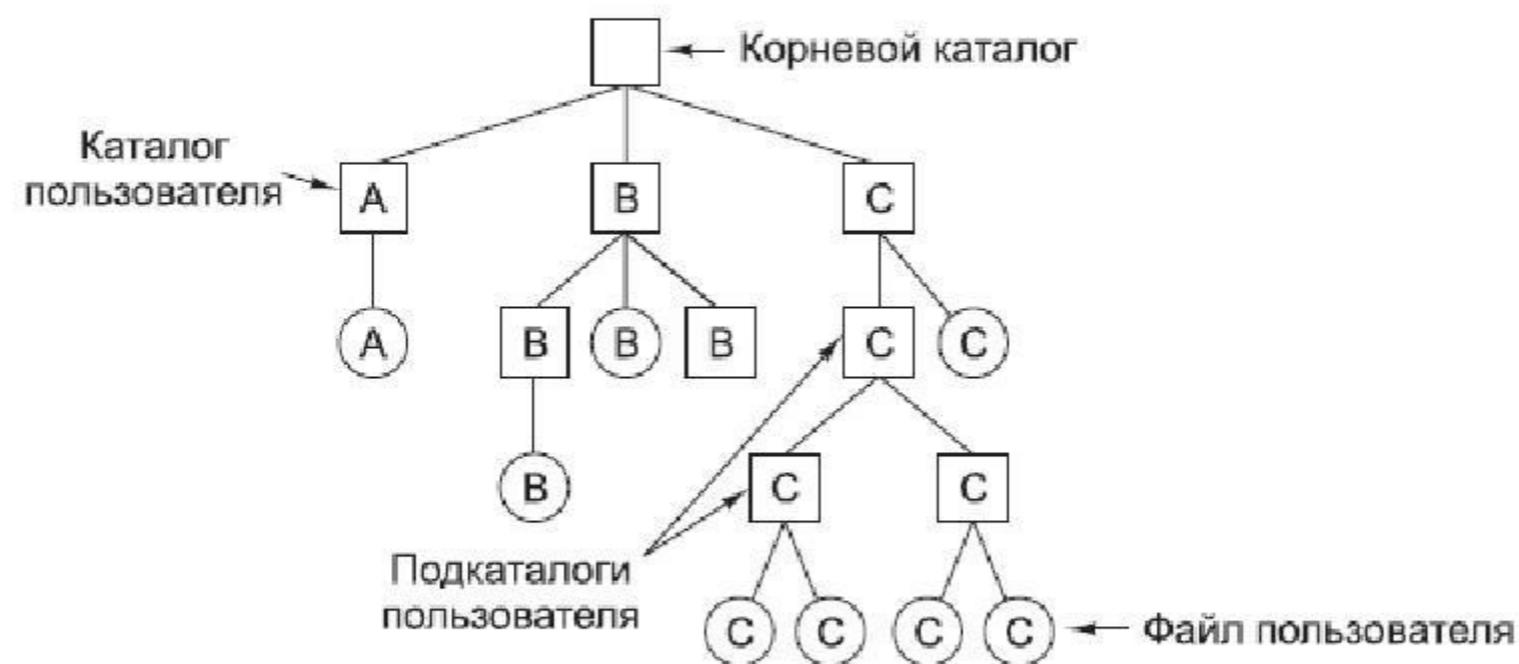
Специализированные файловые системы: ZFS (собственно файловой системой является только часть ZFS), VMFS (т. н. кластерная файловая система, которая предназначена для хранения других файловых систем) и др.

Иерархия файловых систем

Файлы на дисках объединяются в *каталоги*.

Одноуровневая файловая система - все файлы на данном диске хранятся в одном каталоге.

Иерархические системы каталогов



Иерархия файловых систем

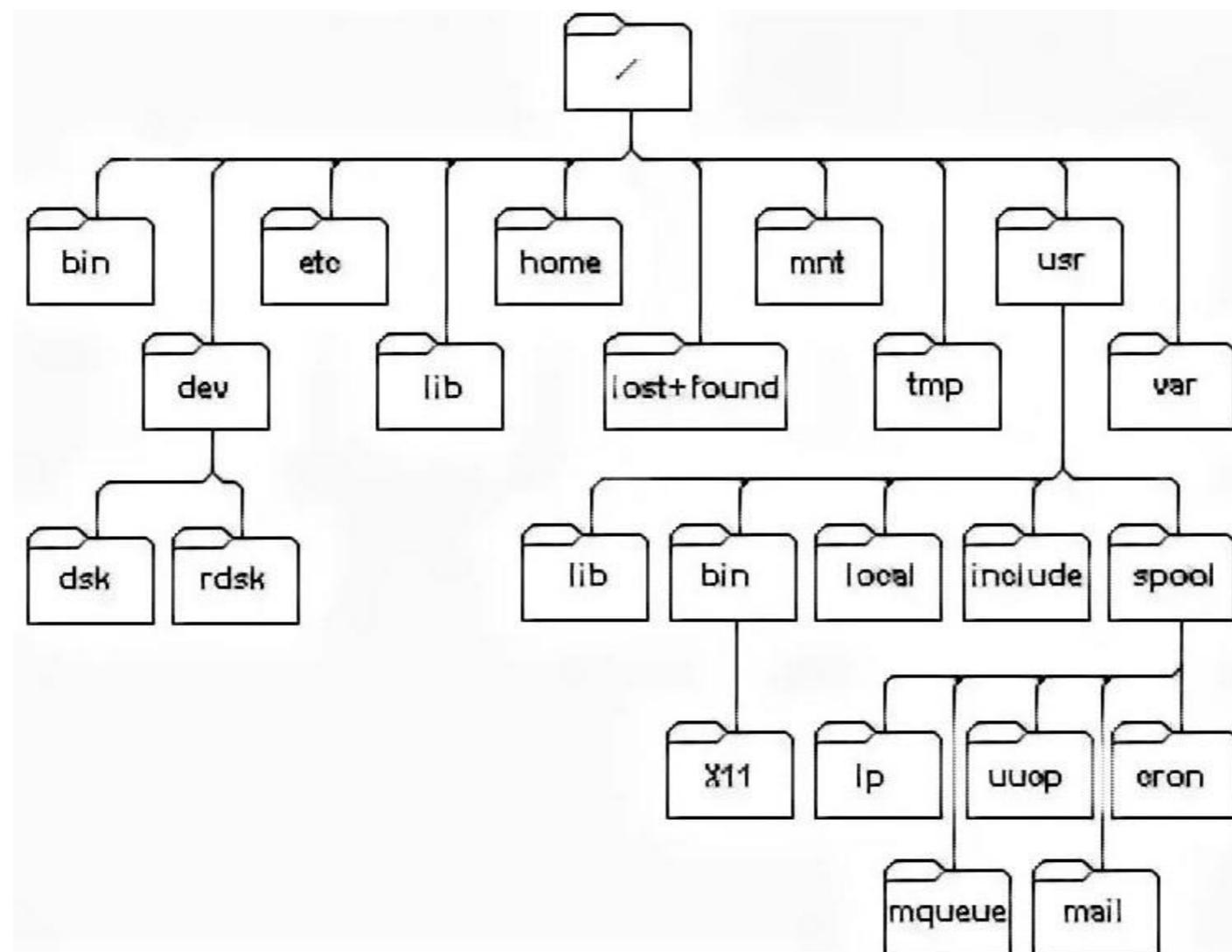
Иерархическая файловая система со вложенными друг в друга каталогами впервые появилась в Multics, затем в UNIX.

Каталоги на разных дисках могут образовывать **несколько отдельных деревьев, как в DOS/Windows**, или же объединяться в одно дерево, общее для всех дисков, как в UNIX-подобных системах.

Обратите внимание на использование слешей в файловых системах Windows, UNIX и UNIX-подобных операционных системах (В Windows используется обратный слеш «\», а в UNIX и UNIX-подобных операционных системах простой слеш «/»)

Иерархия файловых систем

В **UNIX** существует только один корневой каталог, а все остальные файлы и каталоги вложены в него. Чтобы получить доступ к файлам и каталогам на каком-нибудь диске, необходимо смонтировать этот диск командой `mount`. В большинстве **UNIX**-подобных систем съёмные диски (дискеты и CD), флеш-накопители и другие внешние устройства хранения данных монтируют в каталог `/mnt`, `/mount` или `/media`.



Иерархия файловых систем

В файловых системах NTFS и HFS каждый файл представляет собой набор атрибутов. Атрибутами считаются не только традиционные только для чтения, системный, но и имя файла, размер и даже содержимое. Таким образом, для NTFS и HFS то, что хранится в файле, — это всего лишь один из его атрибутов.

Один файл может содержать несколько вариантов содержимого. Таким образом, в одном файле можно хранить несколько версий одного.

Загрузчик ОС

Загрузчик операционной системы — системное программное обеспечение, обеспечивающее загрузку операционной системы непосредственно после включения компьютера.

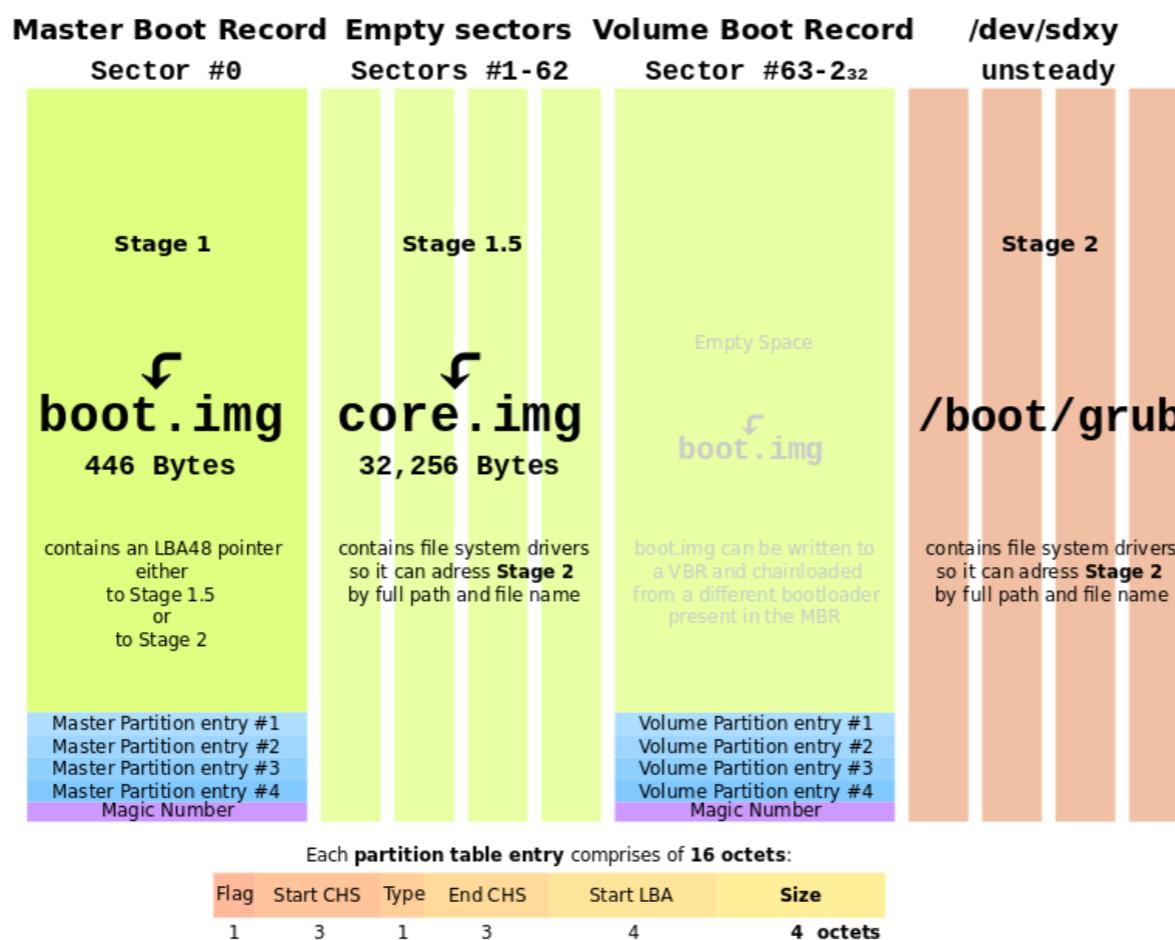
- обеспечивает необходимые средства для диалога с пользователем компьютера (например, загрузчик позволяет выбрать операционную систему для загрузки);
- приводит аппаратуру компьютера в состояние, необходимое для старта ядра операционной системы;
- загружает ядро операционной системы в ОЗУ. Загрузка ядра операционной системы не обязательно происходит с жесткого диска. Загрузчик может получать ядро по сети;
- формирует параметры, передаваемые ядру операционной системы (например, ядру Linux передаются параметры, указывающие способ подключения корневой файловой системы);
- передаёт управление ядру операционной системы.

Загрузчик ОС

- NTLDР — загрузчик ядра Windows NT
- Windows Boot Manager (bootmgr.exe, winload.exe) — загрузчик ядра Windows Vista, Windows 7 и Windows 8
- LILO (LInux LOader) — загрузчик, в основном применяемый для загрузки ядра Linux
- GRUB (Grand Unified Bootloader) — применяется для загрузки ядра Linux и Hurd (StartUp Manager)
- OS/2 BootManager — загрузчик ядра OS/2
- SILO (SPARC Improved bootLOader) — загрузчик Linux и Solaris для машин с архитектурой SPARC
- Loadlin — загружает Linux из-под DOS или Windows.
- BootX — загрузчик Mac OS X
- Acronis OS Selector — коммерческая графическая утилита прилагаемая к Acronis Disk Director, поддерживает Windows и Linux, появляется перед каждой загрузкой системы, умеет копировать системы

MBR

Главная загрузочная запись (*master boot record, MBR*) — код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах (чаще всего в самом первом) на устройстве хранения информации. MBR содержит небольшой фрагмент исполняемого кода, таблицу разделов (partition table) и специальную сигнатуру.



UEFI

Unified Extensible Firmware Interface (EFI) (*Расширяемый интерфейс прошивки*) — интерфейс между операционной системой и микропрограммами, управляющими низкоуровневыми функциями оборудования, его основное предназначение: корректно инициализировать оборудование при включении системы и передать управление загрузчику операционной системы. EFI предназначен для замены BIOS

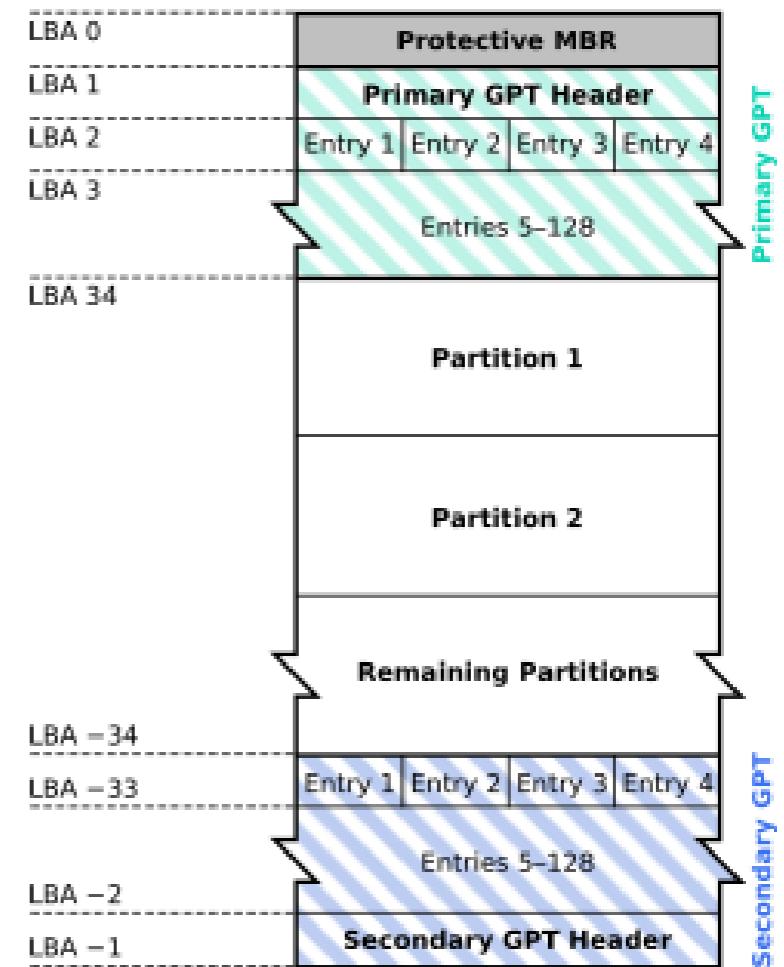
GPT

GUID Partition Table — стандарт формата размещения таблиц разделов на физическом жестком диске. Он является частью EFI — стандарта, предложенного Intel на смену BIOS. EFI использует GPT там, где BIOS использует MBR

Преимущества GPT:

- Использует LBA — адресацию, в отличие от CHS в MBR
- Обеспечивает дублирование — оглавление и таблица разделов записаны как в начале, так и в конце диска
- GPT позволяет создавать разделы диска размером до 9,4 ЗБ ($9,4 \times 1021$ байт), в то время как MBR может работать только до 2,2 ТБ ($2,2 \times 1012$ байт).

GUID Partition Table Scheme



Адресация жестких дисков

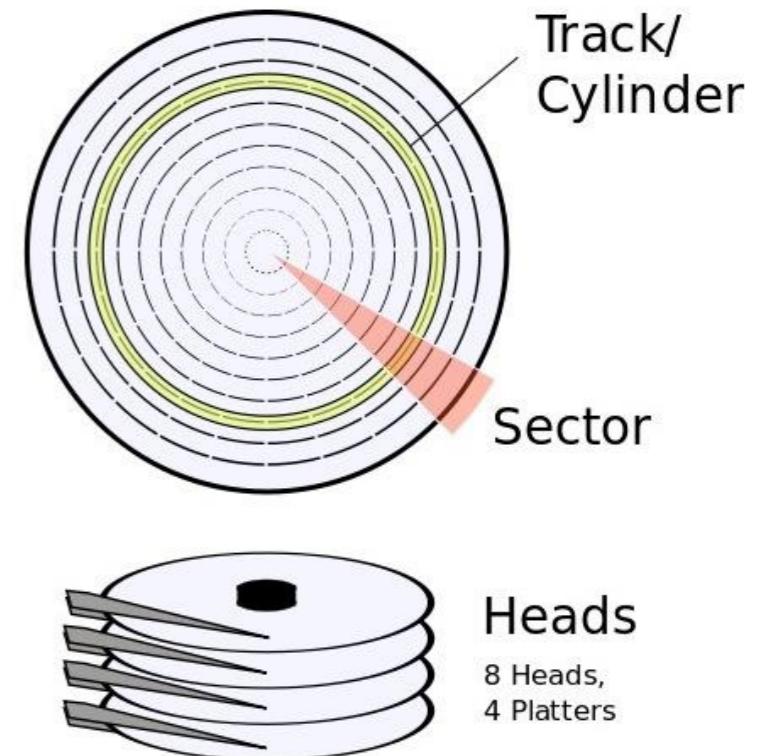
CHS - Сектор определяется на диске 3 координатами:

- Номером цилиндра
- Номером головки
- Номером сектора
- где c — номер цилиндра, h - номер головки, s - номер сектора, H — число головок, S — число секторов на дорожке.

LBA — современная схема адресации, определяется стандартом EIDE (Extended IDE)

$$LBA = [(Cylinder \cdot noofheads + heads) \cdot sectors/track] + (Sector - 1)$$

- LBA — адрес блока по LBA.
- Cylinder — номер цилиндра.
- noofheads — количество головок.
- heads — номер выбранной головки.
- sectors/track — количество секторов на одной дорожке.
- Sector — номер сектора.



$$LBA(c, h, s) = (c \cdot H + h) \cdot S + s - 1$$

$$s = (LBA \bmod S) + 1$$

$$h = \frac{LBA - (s - 1)}{S} \bmod H$$

$$c = \frac{LBA - (s - 1) - h \cdot S}{H \cdot S}$$

Ядро операционной системы

Ядро (kernel) — центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации. Также обычно ядро предоставляет сервисы файловой системы и сетевых протоколов.

Как основополагающий элемент ОС, ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам системы, необходимым для их работы. Как правило, ядро предоставляет такой доступ исполняемым процессам соответствующих приложений за счёт использования механизмов межпроцессного взаимодействия и обращения приложений к системным вызовам ОС.

Описанная задача может различаться в зависимости от типа архитектуры ядра и способа её реализации.

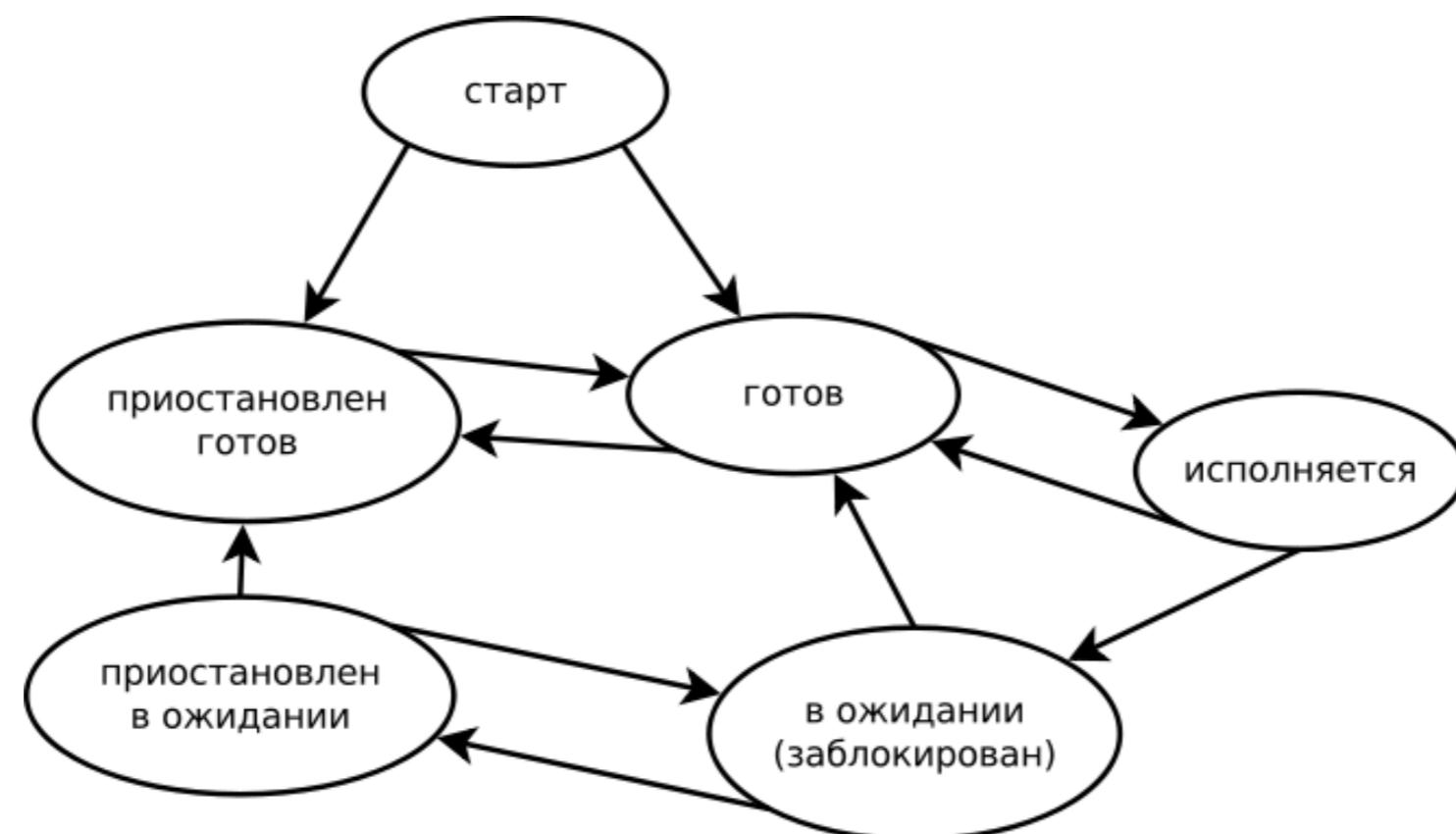
Объекты ядра ОС:

Процесс - команда, которая выполняется в текущий момент

Файлы - именованная область данных на носителе информации.

Событие – соответствует понятию в реальном мире

Поток (выполнения) - процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины.



Объекты ядра ОС:

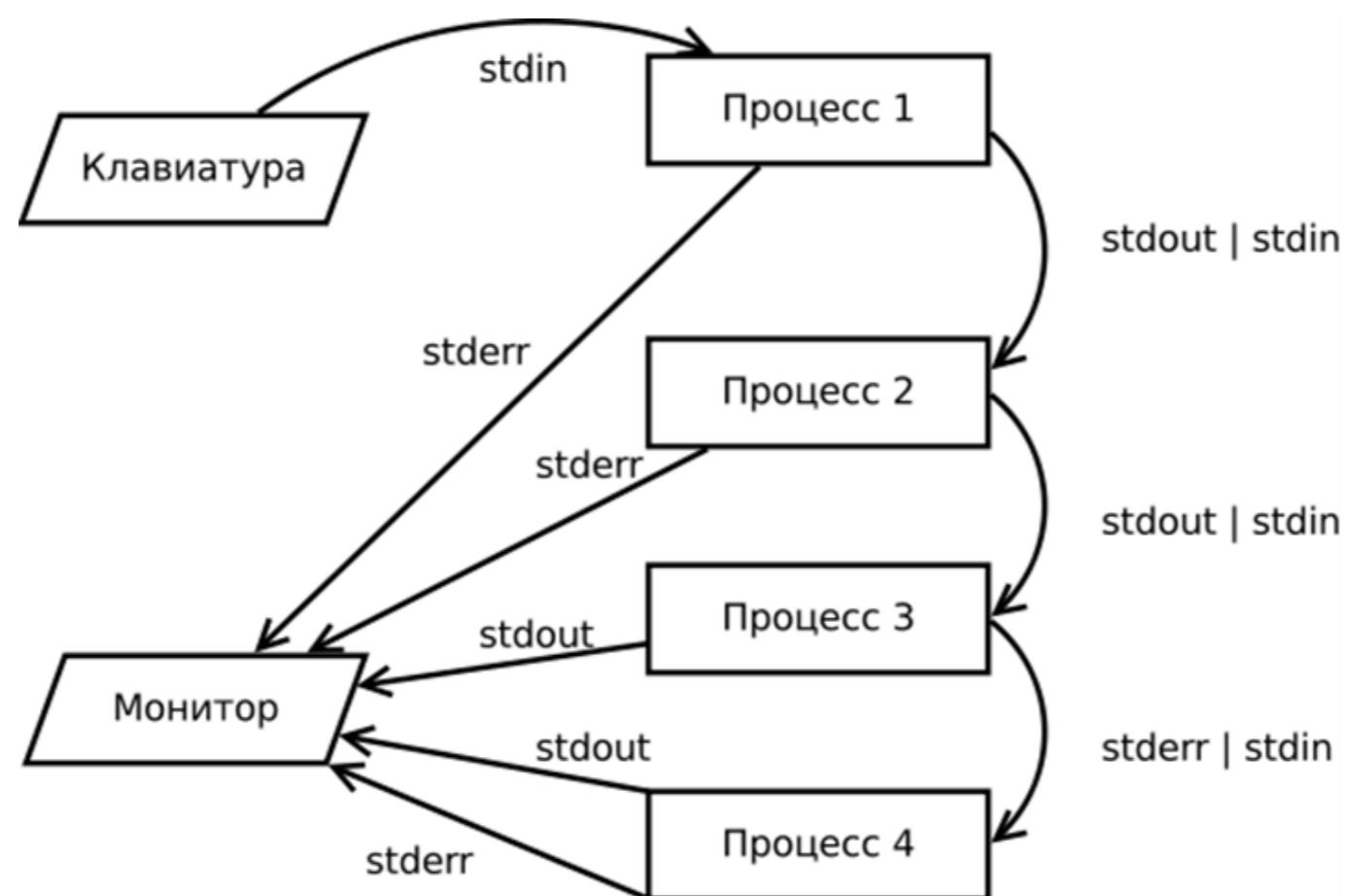
Семафор - объект, ограничивающий количество потоков, которые могут войти в заданный участок кода. Семафоры используются при передаче данных через разделяемую память.

Задачи семафоров:

запрет одновременного выполнения заданных участков кода;
поочерёдный доступ к критическому ресурсу

Объекты ядра ОС:

Мьютекс - одноместный семафор, служащий в программировании для синхронизации одновременно выполняющихся потоков.



Объекты ядра ОС:

Файлы, проецируемые в память это такой способ работы с файлами в некоторых операционных системах, при котором всему файлу или некоторой непрерывной части этого файла ставится в соответствие определённый участок памяти (диапазон адресов оперативной памяти). При этом чтение данных из этих адресов фактически приводит к чтению данных из отображенного файла, а запись данных по этим адресам приводит к записи этих данных в файл.

Объекты ядра ОС:

Канал - один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС. Именованный канал позволяет различным процессам обмениваться данными, даже если программы, выполняющиеся в этих процессах, изначально не были написаны для взаимодействия с другими программами.

Конвейер в терминологии UNIX — некоторое множество процессов, для которых выполнено следующее перенаправление ввода-вывода: то, что выводит на поток стандартного вывода предыдущий процесс, попадает в поток стандартного ввода следующего процесса. Запуск конвейера реализован с помощью системного вызова `pipe()`.

Командный процессор

Оболочка операционной системы — интерпретатор команд ОС, обеспечивающий интерфейс для взаимодействия пользователя с функциями системы.

В общем случае, различают оболочки с двумя типами интерфейса для взаимодействия с пользователем: текстовый пользовательский интерфейс (TUI) и графический пользовательский интерфейс (GUI).

Командный процессор

Командный интерпретатор исполняет команды своего языка, заданные в командной строке или поступающие из стандартного ввода или указанного файла.

В качестве команд интерпретируются вызовы системных или прикладных утилит, а также управляющие конструкции. Кроме того, оболочка отвечает за раскрытие шаблонов имен файлов и за перенаправление и связывание ввода-вывода утилит.

В совокупности с набором утилит, оболочка представляет собой операционную среду, язык программирования и средство решения как системных, так и некоторых прикладных задач, в особенности, автоматизации часто выполняемых последовательностей команд.

Командный процессор

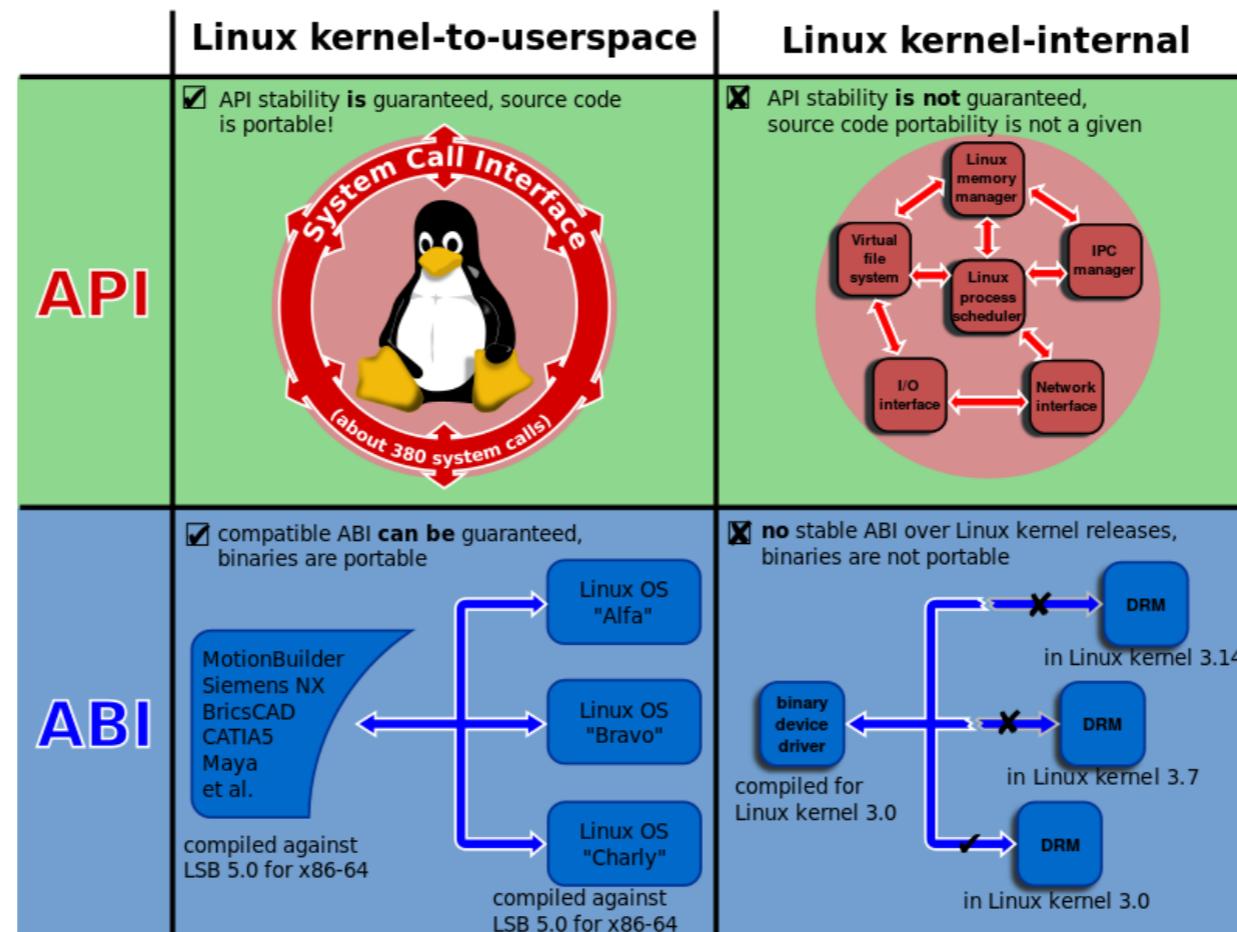
Windows	*nix
command.com – MS-DOS Win 9x	Bash
Windows NT - cmd.exe	Csh
Windows XP (SP2) – PowerShell	Ksh
с Windows 7 и Windows 2008 Server – PowerShell	Zsh
GUI - Проводник	

Многие пользователи и разработчики программного обеспечения пользуются для автоматизации часто выполняемых последовательностей команд операционной системы интерпретируемыми языками программирования, например, Perl или Python.

Системный вызов

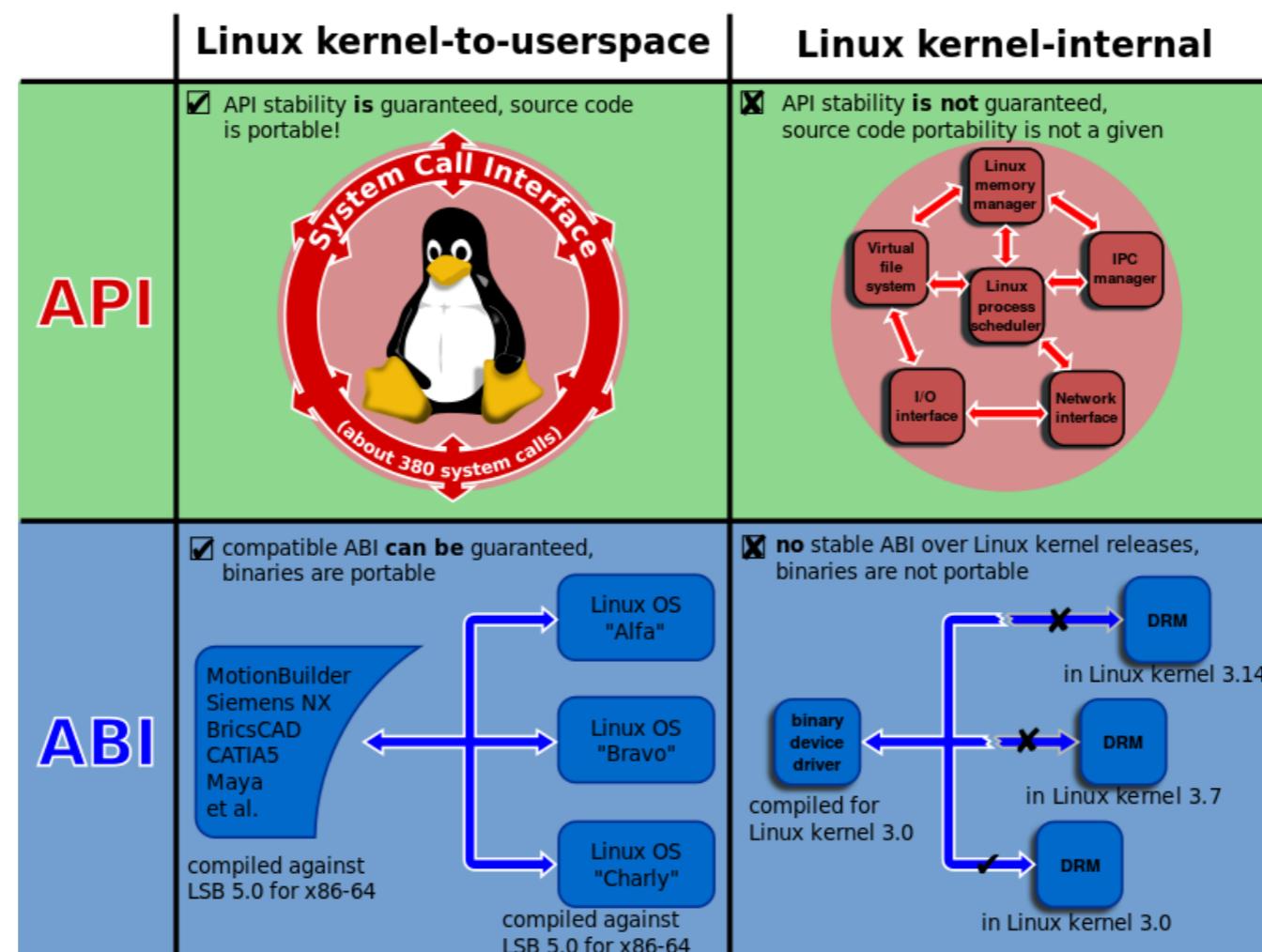
Системный вызов (*system call*) в программировании и вычислительной технике — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

Современные операционные системы (ОС) предусматривают разделение времени между выполняющимися вычислительными процессами и разделение полномочий, препятствующее исполняемым программам обращаться к данным других программ и оборудованию.



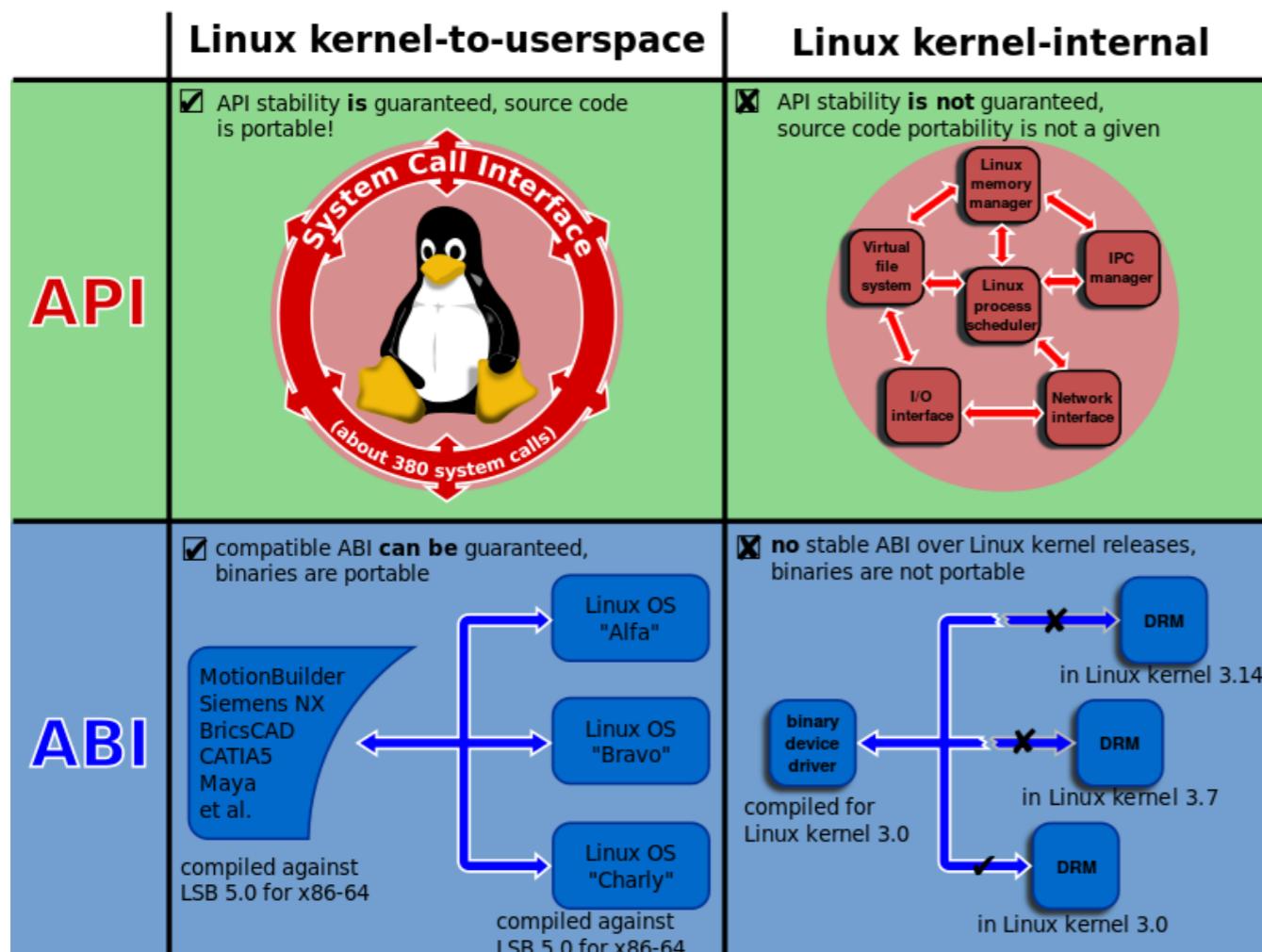
Системный вызов

Ядро ОС исполняется в привилегированном режиме работы процессора. Для выполнения межпроцессной операции или операции, требующей доступа к оборудованию, программа обращается к ядру, которое, в зависимости от полномочийзывающего процесса, исполняет либо отказывает в исполнении такого вызова.



Системный вызов

С точки зрения программиста, системный вызов обычно выглядит как вызов подпрограммы или функции из системной библиотеки. Однако системный вызов, как частный случай вызова такой функции или подпрограммы, следует отличать от более общего обращения к системной библиотеке, поскольку последнее может и не требовать выполнения привилегированных операций.



API - application programming interface
ABI – application binary interface

Драйвер

Драйвер (англ. *driver* — компьютерное программное обеспечение, с помощью которого другое программное обеспечение (операционная система) получает доступ к аппаратному обеспечению некоторого устройства.

Обычно с операционными системами поставляются драйверы для ключевых компонентов аппаратного обеспечения, без которых система не сможет работать. Однако для некоторых устройств могут потребоваться специальные драйверы, обычно предоставляемые производителем устройства.

В общем случае драйвер не обязан взаимодействовать с аппаратными устройствами, он может их только имитировать (например, драйвер принтера, который записывает вывод из программ в файл), предоставлять программные сервисы, не связанные с управлением устройствами (например, /dev/zero в Unix, который только выдаёт нулевые байты), либо не делать ничего (например, /dev/null в Unix и NUL в DOS/Windows).

Основные функции драйверов:

Драйвер состоит из нескольких функций, которые обрабатывают определенные события операционной системы. Обычно это 7 основных событий:

- Загрузка драйвера. Тут драйвер регистрируется в системе, производит первичную инициализацию и т. п.
- Выгрузка. Драйвер освобождает захваченные ресурсы — память, файлы, устройства и т. п.
- Открытие драйвера. Начало основной работы. Обычно драйвер открывается программой как файл, функциями CreateFile() / fopen() для win/*nix
- Запись: программа читает или записывает данные из/в устройство, обслуживаемое драйвером.
- Закрытие: операция, обратная открытию, освобождает занятые при открытии ресурсы и уничтожает дескриптор файла.
- Управление вводом-выводом (англ. IO Control, IOCTL). Зачастую драйвер поддерживает интерфейс ввода-вывода, специфичный для данного устройства. С помощью этого интерфейса программа может послать специальную команду, которую поддерживает данное устройство

Утилиты

Утилита — вспомогательная компьютерная программа в составе общего программного обеспечения для выполнения специализированных типовых задач, связанных с работой оборудования и операционной системы

Виды утилит по связи с ОС:

Независимые утилиты, не требующие для своей работы операционной системы,

Системные утилиты, входящие в поставку ОС и требующие её наличия

Виды утилит по функциям

- Диспетчеры файлов;
- Архиваторы;
- Просмотрщики;
- Утилиты для диагностики аппаратного или программного обеспечения;
- Утилиты восстановления после сбоев;
- Оптимизатор диска — вид утилиты для оптимизации размещения файлов на дисковом накопителе, например, путём дефрагментации диска;
- Деинсталлятор — программа для удаления программного обеспечения;
- Утилиты управления процессами (диспетчер задач).

Прикладная программа или приложение — программа, предназначенная для выполнения определенных пользовательских задач и рассчитанная на непосредственное взаимодействие с пользователем.

Программные средства общего назначения

- Текстовые редакторы
- Текстовые процессоры
- Системы компьютерной вёрстки
- Графические редакторы
- СУБД
- Электронные таблицы
- Веб-браузеры

программные средства специального назначения

- Экспертные системы
- Трансляторы
- Мультимедиа-приложения (медиаплееры, программы для создания и редактирования видео, звука, text-to-speech и пр.)
- Гипертекстовые системы (электронные словари, энциклопедии, справочные системы)
- Системы управления содержимым

профессиональные программные средства

- САПР
- АРМ
- АСУ
- АСУ ТП
- АСНИ
- Геоинформационные системы
- Биллинговые системы
- CRM – системы взаимодействия с клиентом
- СТРМ/ЕТРМ — системы управления складом

Прикладное ПО

профессиональные программные средства

- SRM (Supplier Relationship Management) — системы управления взаимоотношениями с поставщиками
- BI (Business Intelligence) — аналитические системы
- DMS (Document Management System) — СЭД (системы электронного документооборота)
- CMS (Content Management System) — системы управления содержанием (контентом)
- WMS (Warehouse Management System) — системы управления складом (СУС)
- ERP-системы — системы планирования ресурсов предприятия
- ЕАМ-системы — системы управления основными фондами предприятия
- MRM-системы — системы управления маркетинговыми ресурсами
- MES-системы — системы оперативного (цехового) управления производством и ремонтами
- Интеграционные шины данных (ESB)

Встроенное ПО (прошивка)

- Прошивкой (Firmware, fw) называют содержимое энергонезависимой памяти компьютера или любого цифрового вычислительного устройства — микрокалькулятора, сотового телефона, GPS-навигатора и т. д., в которой содержится его микропрограмма.
- Словом «прошивка» иногда называют образ ПЗУ, предназначенный для записи в память соответствующего устройства с целью обновления его микропрограммы, а также собственно процесс записи этого образа в энергонезависимую память устройства.
- Прошивка памяти осуществляется при изготовлении устройства различными способами — например, установкой микросхемы памяти с записанным содержимым («прошитой») или с помощью программатора.
- Большинство устройств допускает замену содержимого памяти («перепрошивку»). Способы «перепрошивки» могут быть самыми различными — от физической замены микросхемы памяти до передачи данных по беспроводным каналам.



Системное программное обеспечение

Лекция 1. Операционные системы

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ



Системное программное обеспечение
Лекция 2. Операционные системы *NIX

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ

open source



Лицензирование ПО

Виды ПО по правообладателю:

Проприетарное программное обеспечение — программное обеспечение, являющееся частной собственностью авторов или правообладателей и не удовлетворяющее критериям свободного ПО (наличия открытого программного кода недостаточно). Правообладатель проприетарного ПО сохраняет за собой монополию на его использование, копирование и модификацию, полностью или в существенных моментах.

Ограничения проприетарного ПО:

- Ограничение на коммерческое использование
- Ограничение на распространение
- Ограничение на изучение, модификацию

Виды ПО по правообладателю:

Свободное программное обеспечение (СПО, free software, также *libre software*), свободный софт — программное обеспечение, пользователи которого имеют права («свободы») на его неограниченную установку, запуск, а также свободное использование, изучение, распространение и изменение (совершенствование), и распространение копий и результатов изменения. Если на программное обеспечение есть исключительные права, то свободы объявляются при помощи свободных лицензий.

Под свободой ПО Столлман имеет в виду свободу

Использовать

Копировать

Распространять

Изменять

GNU GPL

GNU General Public License (переводят как Универсальная общественная лицензия GNU, Универсальная общедоступная лицензия GNU или Открытое лицензионное соглашение GNU) — лицензия на свободное программное обеспечение, созданная в рамках проекта GNU в 1988 г

Цель GNU GPL — предоставить пользователю права копировать, модифицировать и распространять (в том числе на коммерческой основе) программы, а также гарантировать, что и пользователи всех производных программ получат вышеперечисленные права. Принцип «наследования» прав называется «копилефт» и был придуман Ричардом Столлманом. По контрасту с GPL, лицензии проприетарного ПО «очень редко дают пользователю такие права и обычно, наоборот, стремятся их ограничить, например, запрещая восстановление исходного кода».

GNU GPL

Согласно подготовленным Фондом разъяснениям по применению лицензии GNU GPL к конкретным лицензируемым программам, лицензия должна в электронной форме присоединяться к компьютерной программе.

Лицензируя работу на условиях GNU GPL, автор сохраняет за собой авторство.

GNU GPL не позволяет включать программу в проприетарное ПО. Если данная программа является библиотекой, вероятно, лучшим будет разрешить проприетарному ПО линковаться с ней. Для данной цели необходимо использовать GNU Lesser General Public License вместо GPL

GNU GPL

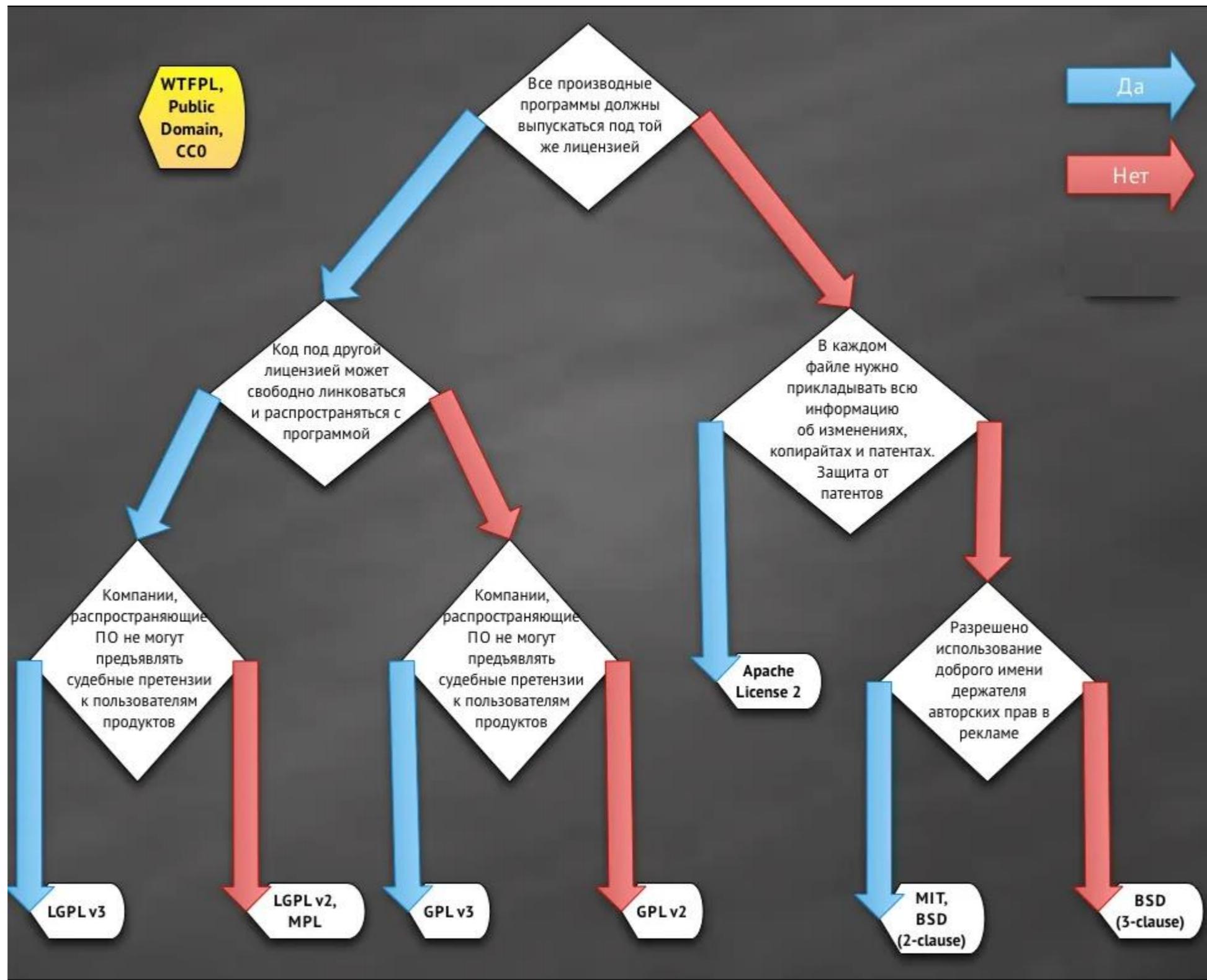
GPL предоставляет получателям компьютерных программ следующие права, или «свободы»:

- свободу запуска программы с любой целью;
- свободу изучения того, как программа работает, и её модификации (доступ к исходному коду);
- свободу распространения копий как исходного, так и исполняемого кода;
- свободу улучшения программы, и выпуска улучшений в публичный доступ

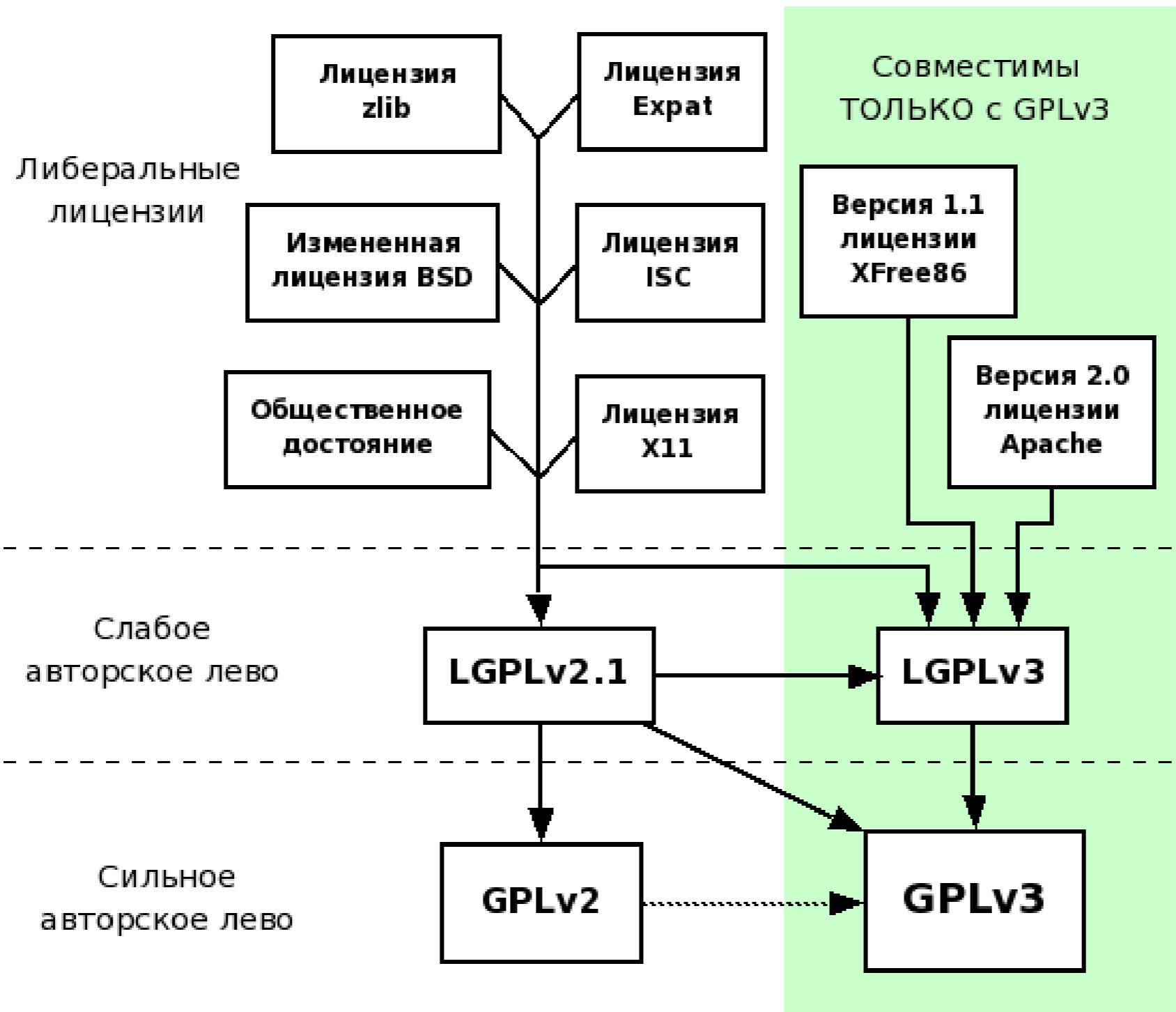


В общем случае распространитель программы, полученной на условиях GPL, либо программы, основанной на таковой, обязан предоставить получателю возможность получить соответствующий исходный код.

Open-source



Open-source



Виды ПО по распространению

Коммерческое программное обеспечение (*commercial software*) — программное обеспечение, созданное с целью получения прибыли от его использования другими, например, путем продажи экземпляров.

Freeware — программное обеспечение, лицензионное соглашение которого не требует каких-либо выплат правообладателю. Freeware обычно распространяется в бинарном виде, без исходных кодов и является проприетарным ПО.

Условно-бесплатное программное обеспечение (*Shareware*) — свободно распространяемое программное обеспечение, с возможным (или возможным при определенных условиях) использованием.

Виды ПО по распространению

Коммерческое ПО с открытым исходным кодом (COSS, синоним Open Core) представляет собой программный продукт, который содержит некоторые элементы свободного и открытого программного обеспечения для того, чтобы законно претендовать на статус «Open Source». Иногда в открытой и бесплатной версии исключаются некоторые возможности, присутствующие в коммерческой версии этого же продукта, которая распространяется по проприетарной лицензии. Открытие части исходного кода, созданного ранее под проприетарной лицензией, оставляет потенциальную возможность привязки такого решения к одному-единственному поставщику.

COSS имеет следующие особенности:

- Гарантия доступности системы в будущем
- Ограничение возможностей открытой версии
- Условия выхода обновлений для открытой версии
- Техническая документация открыта

Виды ПО по распространению

Коммерческое ПО с открытым исходным кодом (COSS, синоним Open Core) представляет собой программный продукт, который содержит некоторые элементы свободного и открытого программного обеспечения для того, чтобы законно претендовать на статус «Open Source». Иногда в открытой и бесплатной версии исключаются некоторые возможности, присутствующие в коммерческой версии этого же продукта, которая распространяется по проприетарной лицензии. Открытие части исходного кода, созданного ранее под проприетарной лицензией, оставляет потенциальную возможность привязки такого решения к одному-единственному поставщику.

COSS имеет следующие особенности:

- Гарантия доступности системы в будущем
- Ограничение возможностей открытой версии
- Условия выхода обновлений для открытой версии
- Техническая документация открыта

Виды ПО по распространению

Коммерческое ПО с открытым исходным кодом (COSS, синоним Open Core) представляет собой программный продукт, который содержит некоторые элементы свободного и открытого программного обеспечения для того, чтобы законно претендовать на статус «Open Source». Иногда в открытой и бесплатной версии исключаются некоторые возможности, присутствующие в коммерческой версии этого же продукта, которая распространяется по проприетарной лицензии. Открытие части исходного кода, созданного ранее под проприетарной лицензией, оставляет потенциальную возможность привязки такого решения к одному-единственному поставщику.

COSS имеет следующие особенности:

- Гарантия доступности системы в будущем
- Ограничение возможностей открытой версии
- Условия выхода обновлений для открытой версии
- Техническая документация открыта

Операционные системы UNIX

Операционные системы UNIX

История развития

- 1969 г. – Bell Labs AT&T создали операционную систему UNIX
- Система изначально создавалась как многопользовательская, т.е. поддерживающая в один момент времени одновременную работу нескольких пользователей.
- Предшественником UNIX стал Multics - одна из первых ОС с разделением времени исполнения программ

Роль UNIX:

- Так же, как и Multics, была написана не на ассемблере , а на языке высокого уровня (C)
- Она содержала упрощённую, файловую модель. Файловая система включала как службы, так и устройства (такие как принтеры, терминалы и жёсткие диски) и предоставляла внешне единообразный интерфейс к ним с дополнительными механизмами работы с устройствами
- Популяризовала предложенную в Multics идею иерархической файловой системы с произвольной глубиной вложенности (создание рекурсивных подкаталогов)
- Интерпретатор команд стал одной из пользовательских команд, а в качестве команд выступают отдельные программы
- Новые команды можно добавлять без перекомпиляции ядра

Роль UNIX

- Ориентация на текстовый ввод-вывод
- Размер машинного слова равен 8 бит (1 байт)
- Распространению регулярных выражений. Возможности, предоставляемые UNIX-программам, стали основой стандартных интерфейсов операционных систем (POSIX).
- Использование языка Си, первого ЯВУ, предоставляющим доступ ко всем возможностям процессора, таким как ссылки, таблицы, битовые сдвиги, инкременты
- Возможность использования протоколов TCP/IP на сравнительно недорогих компьютерах, что привело к быстрому росту Интернета

Регулярные выражения

- Регулярные выражения (*regular expressions*) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. По сути это строка-образец («шаблон», «маска»), состоящая из символов и метасимволов и задающая правило поиска.
- Пример:
 - найти все последовательности символов «кот» в любом контексте, как то: «кот», «котлета», «терракотовый»;
 - найти отдельно стоящее слово «кот» и заменить его на «кошка»;
 - найти слово «кот», которому предшествует слово «персидский» или «чеширский»;
 - убрать из текста все предложения, в которых упоминается слово кот или кошка.

Архитектурные особенности UNIX

- Мультиплатформенность ядра
- Файловая система древовидная, чувствительная к регистру символов в именах, очень слабые ограничения на длину имён и пути
- Нет поддержки структурированных файлов ядром ОС, на уровне системных вызовов файл есть поток байтов
- Командная строка находится в адресном пространстве запускаемого процесса, а не извлекается системным вызовом из процесса интерпретатора команд
- Понятие «переменных окружения» (Текстовых переменных, хранящих информацию о состоянии системы)
- Запуск процессов вызовом `fork()`, то есть возможность клонирования текущего процесса со всем состоянием.
- Понятия `stdin/stdout/stderr`.
- Ввод-вывод только через дескрипторы файлов.
 - **Файловый дескриптор** — это неотрицательное целое число. Когда создается новый поток ввода-вывода, ядро возвращает процессу, создавшему поток ввода-вывода, его файловый дескриптор.
 - файловый дескриптор 0 — с потоком стандартного ввода процесса (терминал)
 - файловый дескриптор 1 — с потоком стандартного вывода (терминал)
 - файловый дескриптор 2 — с потоком диагностики (куда обычно выводятся сообщения об ошибках).
- Интерпретатор команд есть обычное приложение, общающееся с ядром обычными системными вызовами
- Команда командной строки есть не более чем имя файла программы

Архитектурные особенности UNIX

- Пространство имен устройств на диске в каталоге /dev, поддающееся управлению администратором, в отличие от подхода Windows, где это пространство имен размещается в памяти ядра, и администрирование этого пространства (например, задание прав доступа) крайне затруднено из-за отсутствия его постоянного хранения на дисках
- Широкое использование текстовых файлов для хранения настроек, в отличие от двоичной базы данных настроек, как, например, в Windows
- Широкое использование утилит обработки текста для выполнения повседневных задач под управлением скриптов.
- Широкое использование именованных каналов (pipe)
- Все процессы, кроме init, равны между собой, не бывает «специальных процессов».
- Адресное пространство делится на глобальное для всех процессов ядро и на локальную для процесса части, нет «групповой» части адресного пространства, как и возможности загрузки туда кода и его исполнения там
- Использование двух уровней привилегий процессора вместо четырёх

Стандартные команды ОС UNIX

- Создание и навигация по файлам и каталогам: [touch](#), [ls](#), [mv](#), [rm](#), [cp](#), [ln](#), [pwd](#), [cd](#), [mkdir](#), [rmdir](#), [find](#), [du](#);
- Просмотр и редактирование файлов: [nano](#), [more](#), [less](#), [ed](#), [ex](#), [vi](#), [emacs](#);
- Обработка текста: [echo](#), [cat](#), [grep](#), [sort](#), [uniq](#), [sed](#), [awk](#), [tee](#), [head](#), [tail](#), [cut](#), [tr](#), [split](#), [printf](#);
- Сравнение файлов: [comm](#), [cmp](#), [diff](#), [patch](#);
- Разнообразные утилиты командного интерпретатора: [yes](#), [test](#), [xargs](#), [expr](#);
- Системное администрирование: [chmod](#), [chgrp](#), [chown](#), [ps](#), [su](#), [w](#), [who](#), [df](#), [mount](#), [umount](#);
- Коммуникации: [mail](#), [telnet](#), [ftp](#), [finger](#), [rsh](#), [ssh](#);
- Командные оболочки: [sh](#), [bash](#), [csh](#), [ksh](#), [tcsh](#), [zsh](#);
- Работа с исходным кодом и объектным кодом: [cc](#), [gcc](#), [ld](#), [nm](#), [yacc](#), [bison](#), [lex](#), [flex](#), [ar](#), [ranlib](#), [make](#);
- Сжатие и архивация: [compress](#), [uncompress](#), [gzip](#), [gunzip](#), [tar](#)
- Работа с двоичными файлами: [od](#), [strings](#)

POSIX

- **POSIX** (*Portable Operating System Interface for Unix* — Переносимый интерфейс операционных систем Unix) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем.
- Серия стандартов POSIX была разработана комитетом 1003 IEEE. Международная организация по стандартизации (ISO) совместно с Международной электротехнической комиссией (IEC) приняли данный стандарт (POSIX) под названием **ISO/IEC 9945**.
- **Задачи**
 - содействовать облегчению переноса кода прикладных программ на иные платформы;
 - способствовать определению и унификации интерфейсов заранее при проектировании, а не в процессе их реализации;
 - сохранять по возможности и учитывать все главные, созданные ранее и используемые прикладные программы;
 - определять необходимый минимум интерфейсов прикладных программ, для ускорения создания, одобрения и утверждения документов;
 - развивать стандарты в направлении обеспечения коммуникационных сетей, распределенной обработки данных и защиты информации;
 - рекомендовать ограничение использования бинарного (объектного) кода для приложений в простых системах.

Полностью POSIX совместимые

- [A/UX](#)
- [BSD/OS](#)
- [HP-UX](#)
- [IBM AIX](#)
- [Integrity](#) и её микроядро [μ-velocity](#)
- [IRIX](#)
- [LynxOS](#)
- [Mac OS X](#)
- [Minix](#)
- [MPE/iX](#)
- [OpenSolaris](#)
- [OpenVMS](#)
- [QNX](#)
- [RTEMS](#)
- [Solaris](#)
- [UnixWare](#)
- [VxWorks](#)

По большей части POSIX-совместимые

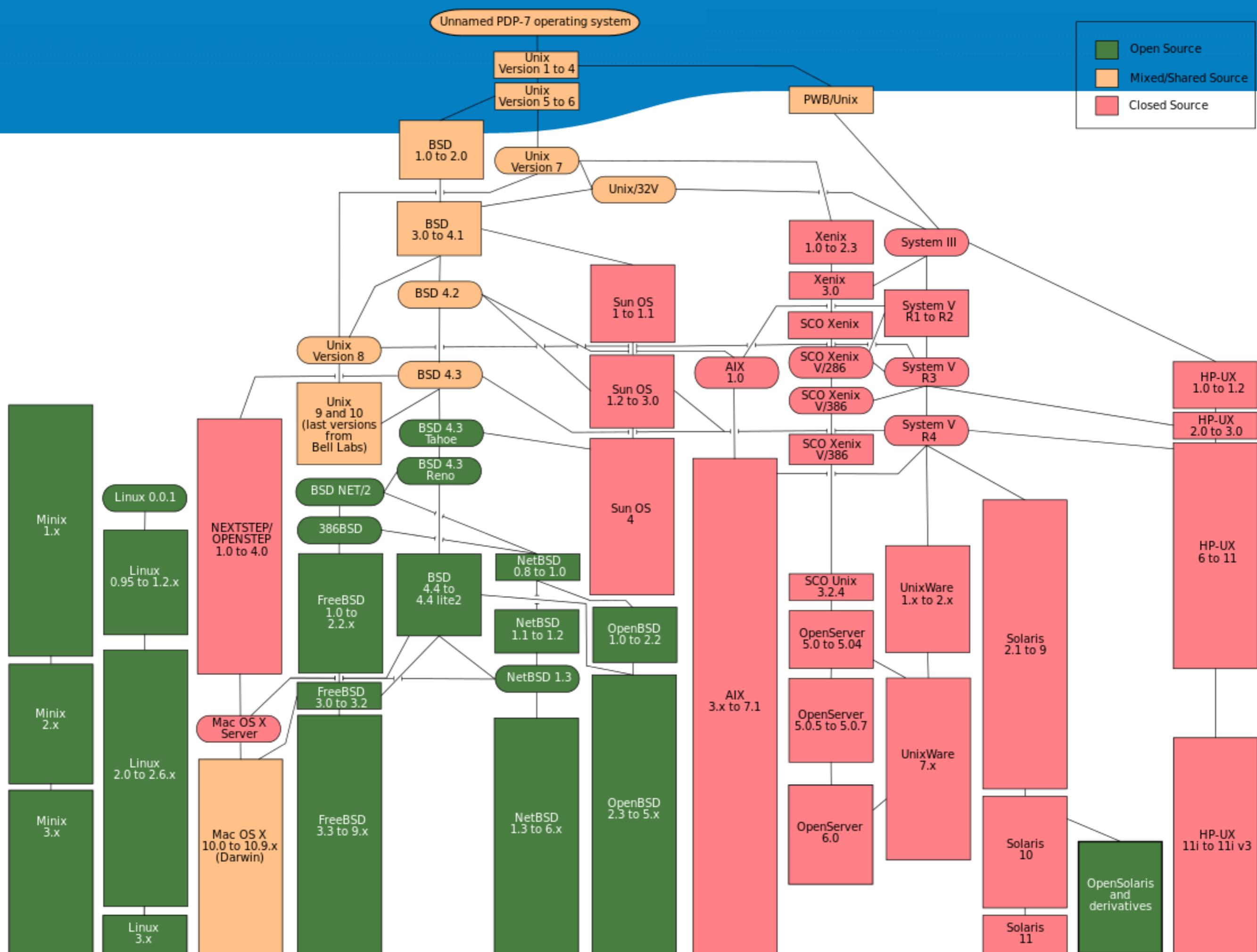
- [BeOS](#)
- [FreeBSD](#)
- [Linux NetBSD](#)
- [Nucleus RTOS](#)
- [OpenBSD](#)
- [Sanos](#)
- [SkyOS](#)
- [Syllable](#)
- [VSTa](#)
- [Symbian OS](#) (при помощи [PIPS](#))
- [DragonFlyBSD](#)

Операционные системы - последователи UNIX

- Linux
- Minix
- BSD
- FreeBSD
- Darwin -> Mac OS X -> Apple iOS
- Solaris

Философия UNIX

- Дуг Макилрой:
 - «Пишите программы, которые делают что-то одно и делают это хорошо.»
 - «Пишите программы, которые бы работали вместе.»
 - «Пишите программы, которые бы поддерживали текстовые потоки, поскольку это универсальный интерфейс». Обычно эти высказывания сводятся к одному «Делайте что-то одно, но делайте это хорошо».
- Майк Ганцарз:
 - Красиво — небольшое.
 - Пусть каждая программа делает что-то одно, но хорошо.
 - Страйте прототип программы как можно раньше.
 - Предпочитайте переносимость эффективности.
 - Храните данные в простых текстовых файлах.
 - Обратите преимущества программных средств себе на пользу.
 - Используйте сценарии командной строки для улучшения функциональности и переносимости.
 - Избегайте пользовательских интерфейсов, ограничивающих возможности пользователя по взаимодействию с системой.
 - Делайте каждую программу «фильтром».





Системное программное обеспечение
Лекция 2. Операционные системы *NIX

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ



Системное программное обеспечение
Лекция 3. Операционные системы *NIX

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ

Компоненты UNIX - систем

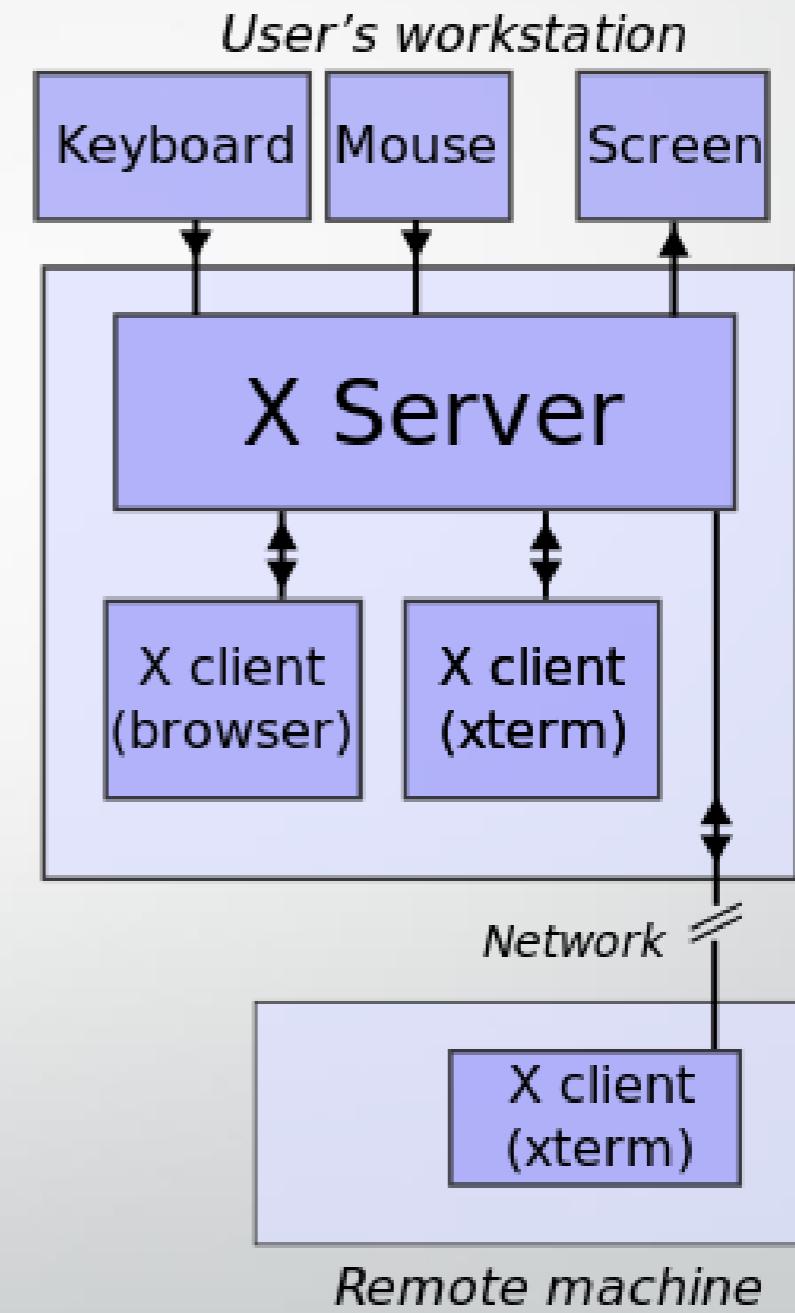
Сервер X11



- **X Window System** — оконная система, обеспечивающая стандартные инструменты и протоколы для построения графического интерфейса пользователя.
- В X Window System предусмотрена *сетевая прозрачность*: графические приложения могут выполняться на другой машине в сети, а их интерфейс при этом будет передаваться по сети и отображаться на локальной машине пользователя. В контексте X Window System термины «клиент» и «сервер» имеют непривычное для многих пользователей значение: «сервер» означает локальный дисплей пользователя (*дисплейный сервер*), а «клиент» — программу, которая этот дисплей использует (она может выполняться на удалённом компьютере).
- Система X Window System была разработана в Массачусетском технологическом институте (MIT) в 1984 году. Нынешняя версия протокола — X11 — появилась в сентябре 1987 года.

Пример:

- На пользовательской рабочей станции выполняются веб-браузер и эмулятор терминала.



Эмулятор терминала

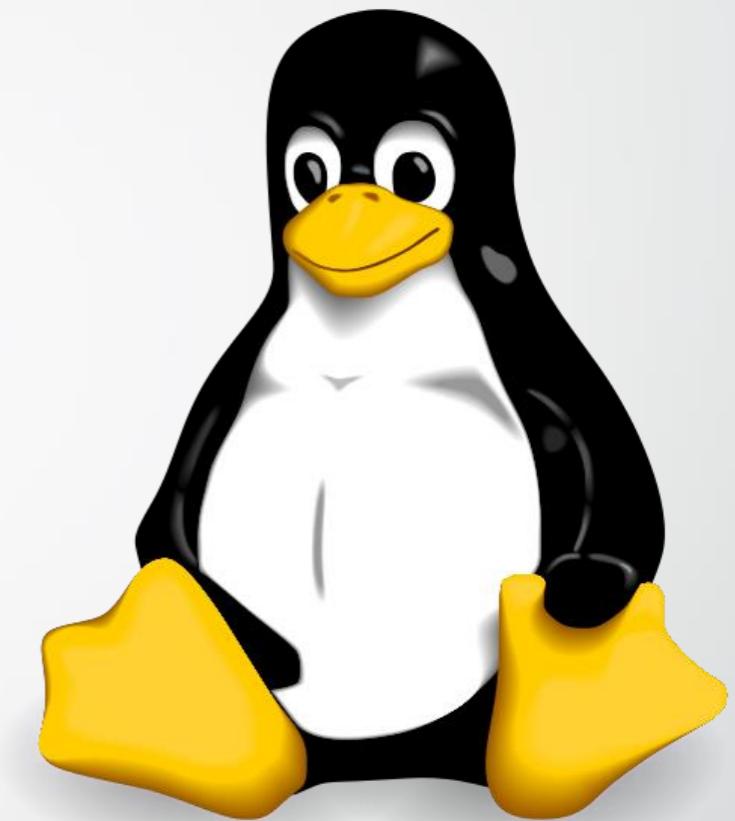
- Эмулятор терминала `term` или `tty` для краткости — это программа, которая эмулирует терминал компьютера внутри некоторой другой архитектуры вывода данных на экран.
- Несмотря на глубокую синонимичность с оболочкой командной строки или текстовым терминалом, термин *терминал* охватывает все удалённые терминалы, включая графические интерфейсы.
- Эмулятор терминала в оконном интерфейсе пользователя часто называется *окном терминала*.
- Примеры терминальных программ: PuTTY, Qt_comport, Tera Term.

bash

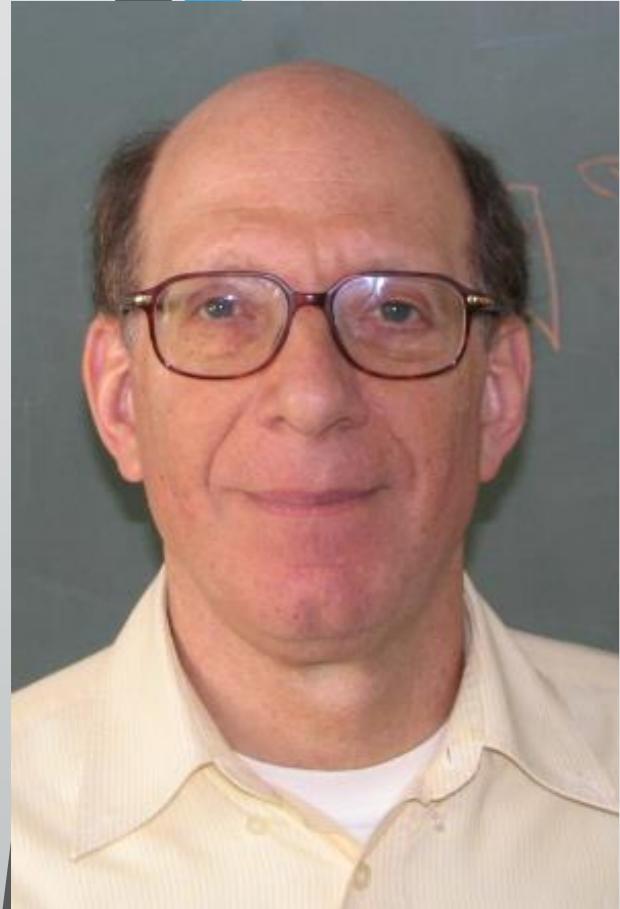
- Усовершенствованная вариация командной оболочки Bourne shell
- Bash — это командный процессор, работающий, как правило, в интерактивном режиме в текстовом окне.
- Bash также может читать команды из файла, который называется *скриптом* (или *сценарием*).
- Поддерживает автодополнение имён файлов и директорий, подстановку вывода результата команд, переменные, контроль за порядком выполнения, операторы ветвления и цикла.
- Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh.
- Другие функции, например, история, были скопированы из csh и ksh.
- Bash в основном удовлетворяет стандарту POSIX, но с рядом расширений

Команды SU и SUDO

- **su** (*Substitute User, Set UID, Switch User* замена пользователя, переключение пользователя) — команда Unix-подобных операционных систем, позволяющая пользователю войти в систему под другим именем, не завершая текущий сеанс
- **sudo** (*substitute user and do*, дословно «подменить пользователя и выполнить») — программа для системного администрирования UNIX-систем, позволяющая делегировать те или иные привилегированные ресурсы пользователям с ведением протокола работы. Основная идея — дать пользователям как можно меньше прав, при этом достаточных для решения поставленных задач.
- По умолчанию пользователь root



Операционные системы Linux



История создания

- MINIX является недорогой минимальной UNIX-подобной операционной системой, предназначенней для образовательных целей в области компьютерных наук, написанной Эндрю Таненбаумом.
- В 1991 году, во время обучения в Хельсинкском университете Торвальдс заинтересовался операционными системами и был разочарован лицензией MINIX, вследствие чего начал работать над своей собственной операционной системой, которая в итоге стала Linux.
- Торвальдс начал разработку ядра Linux на MINIX, и приложения, написанные для MINIX, были также использованы в Linux. Позже, когда Linux достиг определённой зрелости, появилась возможность продолжать разработку уже на базе самого Linux.
- Для того чтобы сделать Linux доступным для коммерческого использования, Торвальдс начал переходить от своей первоначальной лицензии (которая запрещала коммерческое распространение) на GNU GPL.

Текущее состояние:

- В настоящее время системы Linux лидируют на рынках смартфонов (Android занимает 64,1 % рынка),
- интернет-серверов (60 %)
- самых мощных суперкомпьютеров (97 %),
- в данных центрах и на предприятиях, занимают половину рынка встраиваемых систем, имеют значительную долю рынка нетбуков (32 % на 2009 год).
- На рынке домашних компьютеров Linux прочно занимает 3 место (по разным данным, от 1 до 5 %).
- Согласно исследованию Goldman Sachs, в целом, рыночная доля Linux среди электронных устройств составляет около 42 %
- В 2008 году расчёты показывали, что для того, чтобы «с нуля» разработать систему, аналогичную Fedora 9, потребовалось бы затратить 10,8 млрд долл. Совокупная себестоимость ядра Linux оценена в более чем 1 млрд евро (около 1,4 млрд долл.). Только за 2008 год себестоимость ядра Linux увеличилась на 225 млн евро. В системе Linux воплощён труд в эквиваленте 73 тыс. человеко-лет

Применение:

- Встраиваемые системы
- Серверы, требующие высокого аптайма.
- Компьютеры нестандартной архитектуры (например, суперкомпьютеры) — из-за возможности быстрой адаптации ядра операционной системы и большого количества ПО под нестандартную архитектуру.
- Системы военного назначения (например, МСВС РФ) — по соображениям безопасности.
- Компьютеры, встроенные в различные устройства (банкоматы, терминалы оплаты, мобильные телефоны, маршрутизаторы, стиральные машины и даже беспилотные военные аппараты) — из-за широких возможностей по конфигурированию Linux под задачу, выполняемую устройством, а также отсутствия платы за каждое устройство.
- Массовые специализированные рабочие места (например, тонкие клиенты, нетбуки) — также из-за отсутствия платы за каждое рабочее место и по причине их ограниченной вычислительной мощности, которой может не хватать для проприетарных ОС.
- Старые компьютеры с ограниченными ресурсами быстродействия и оперативной памяти, для них используются быстрые рабочие окружения или оконные менеджеры, не требовательные к ресурсам (например, LXDE, Openbox, Xfce, Fluxbox).

Загрузчик GRUB

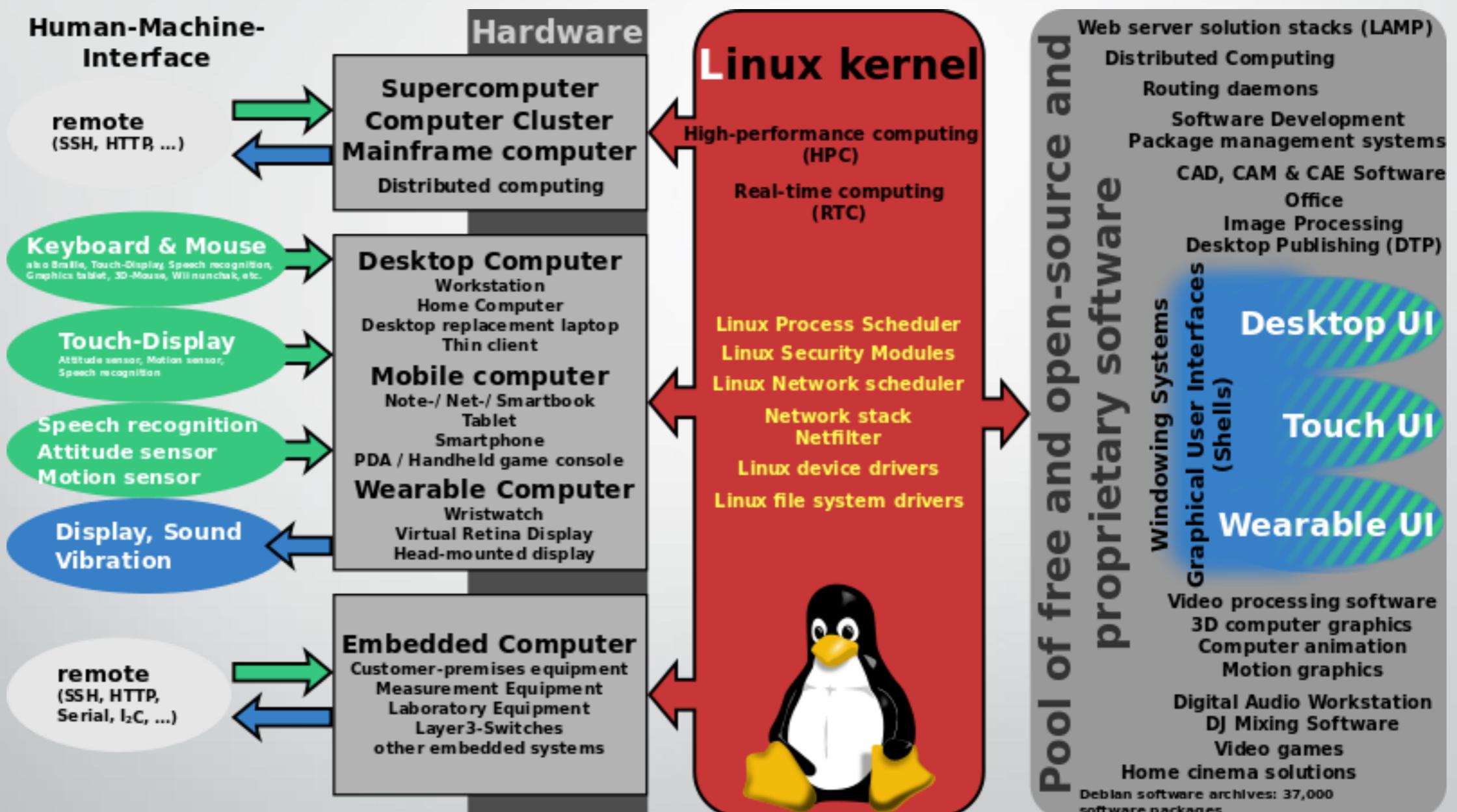
- **GRUB** — загрузчик операционной системы от проекта GNU. GRUB позволяет пользователю иметь несколько установленных операционных систем и при включении компьютера выбирать одну из них для загрузки.
- Загрузка Linux, OpenSolaris, *BSD ядер и других Multiboot-совместимых ОС;
- Передача управления другим загрузчикам, возможность загрузки Windows-систем;
- Защита паролем пунктов меню;
- Поддержка BOOTP и TFTP для сетевой загрузки;
- Интерактивная командная строка загрузки;
- Поддержка файловых систем: NTFS, ISO, UFS, UFS2, FFS, FAT16, FAT32, Minix, ext2/ext3/ext4, ReiserFS, JFS и XFS, и чтение файлов конфигурации, ядер, initrd и других файлов прямо с файловой системы.
- Поддержка загрузчика EFI, начиная с версии 1.98 (загрузка операционных систем семейства Mac OS).
- Помимо всего прочего, GRUB может достаточно гибко менять внешний вид, например показывать красивую картинку в загрузочном меню.
- GRUB2 имеет модульную структуру. Это означает, что туда легко добавляется всё, что угодно, вплоть до игр, также как и выбрасывается, если необходимо уменьшить размер.



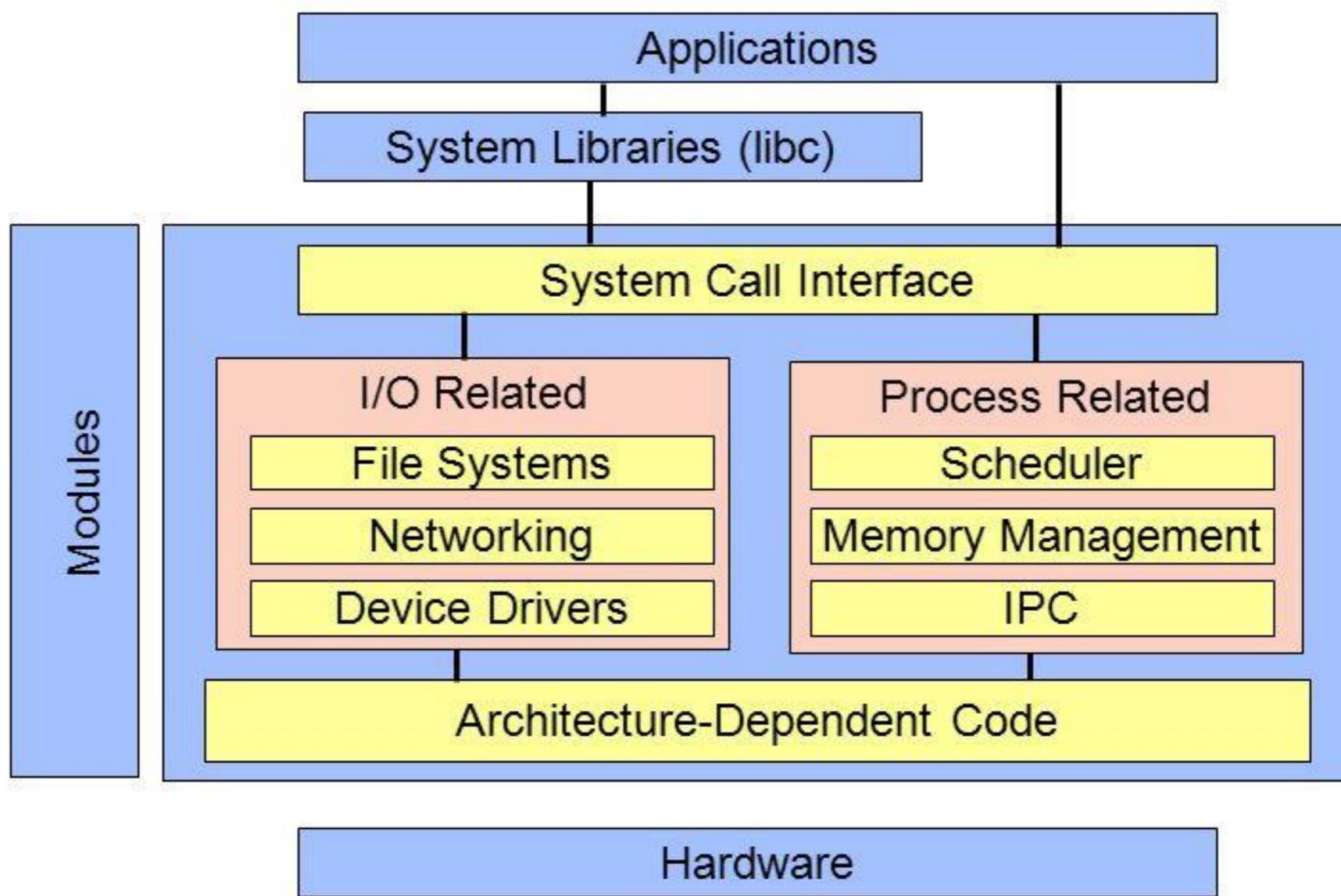
```
Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+
```

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

Ядро



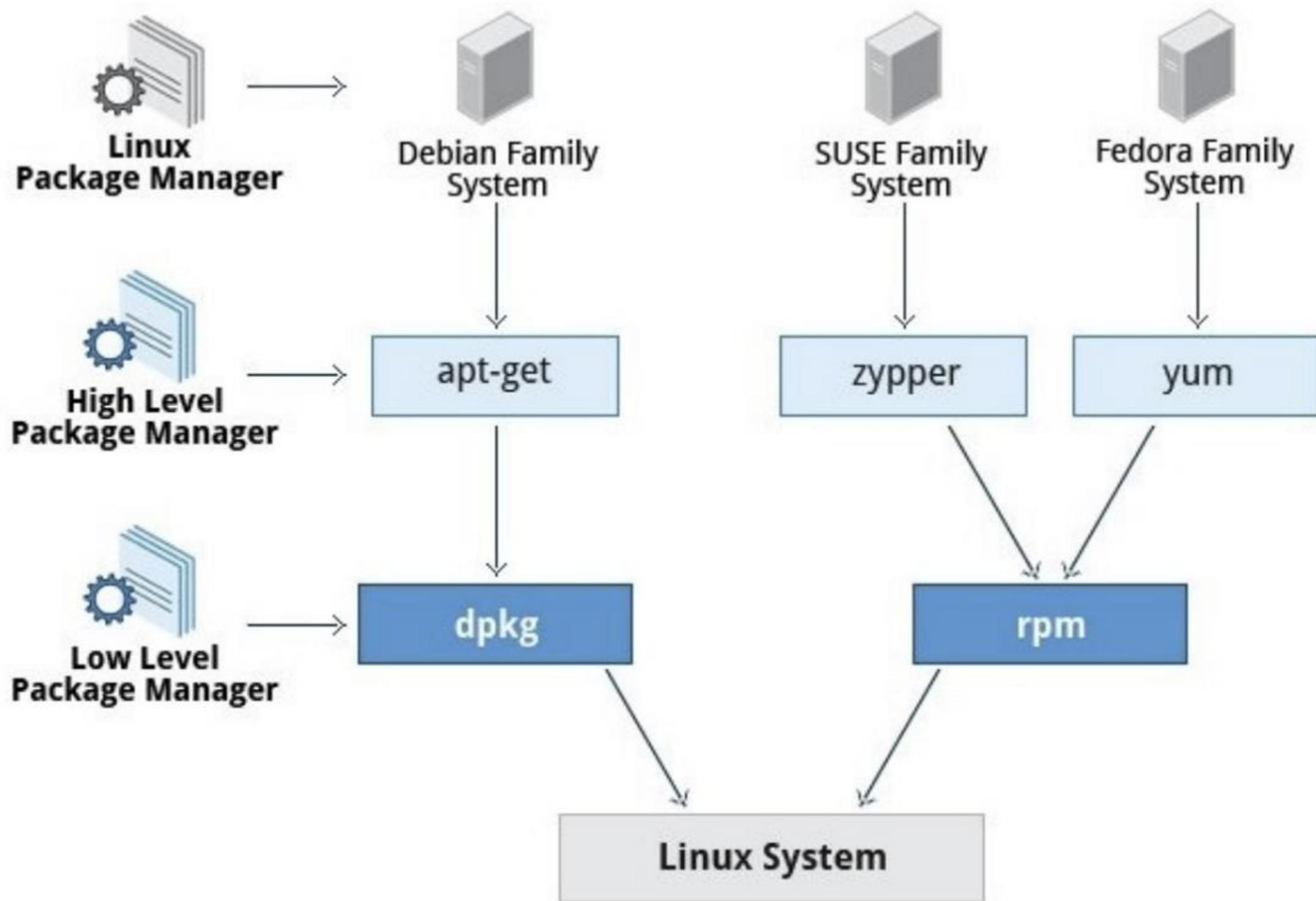
Linux Architecture



Пакеты ПО

- Пакет прикладных программ (аббр. ППП, англ. application package) или пакет программ — набор взаимосвязанных модулей, предназначенных для решения задач определённого класса некоторой предметной области. По смыслу ППП было бы правильнее называть пакетом модулей вместо устоявшегося термина пакет программ

Пакеты ПО



Пакеты ПО

- **Репозиторий, хранилище** — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.
- **apt** (*advanced packaging tool*) — программа для установки, обновления и удаления программных пакетов в операционных системах Debian и основанных на них (Ubuntu, Mint и т. п.), иногда также используется в дистрибутивах, основанных на Mandrake, например Mandriva, ALTLinux и PCLinuxOS. Способна автоматически устанавливать и настраивать программы для UNIX-подобных операционных систем как из предварительно откомпилированных пакетов, так и из исходных кодов.
- **Yellow dog Updater, Modified (YUM)** — открытый консольный менеджер RPM (Red Hat Package Manager)-пакетов. Позволяет облегчить каскадное обновление Linux систем с отслеживанием взаимосвязей RPM-пакетов. Распространяется под лицензией GNU.
- Системные администраторы могут автоматизировать обновление ПО используя ряд инструментов, таких как yum-updatesd, yum-updateonboot, yum-cron, PackageKit.

Среда рабочего стола

- Среда рабочего стола — это разновидность графических интерфейсов пользователя, основанная на *метафоре рабочего стола*.
- Такая среда обеспечивает пространство, называемое рабочим столом, на котором появляются окна, пиктограммы, панели и другие элементы. Обычно поддерживаются механизмы, объединяющие разные части среды, — например, drag-n-drop (перенос данных между окнами с помощью указательного устройства). Назначение рабочего окружения — создание интуитивного способа взаимодействия пользователя с компьютером.

GNOME

- **GNOME** — свободная среда рабочего стола для Unix-подобных операционных систем. GNOME является частью проекта GNU.
- Разработчики GNOME ориентируются на создание полностью свободной среды, доступной всем пользователям вне зависимости от их уровня технических навыков, физических ограничений и языка, на котором они говорят. В рамках проекта GNOME разрабатываются как приложения для конечных пользователей, так и набор инструментов для создания новых приложений, тесно интегрируемых в рабочую среду.
- GNOME — *GNU Network Object Model Environment* («сетевая среда объектной модели GNU»). Под GNU в данном случае подразумевается не проект, а операционная система, официальной средой рабочего стола которой он является.

Activities

Mon 21:45



Type to search...



AisleRiot Solit...



Books



Brasero



Calendar



Cheese



Contacts



Documents



Empathy



Evolution



Firefox Web ...



gedit



LibreOffice



LibreOffice C...



LibreOffice Dr...



LibreOffice Im...



LibreOffice M...



LibreOffice W...



Mahjongg



Maps



Mines



Music



Photos



Settings

Frequent

All



KDE/Plasma

- **KDE Software Compilation (KDE SC)** — свободная среда рабочего стола и набор программ от проекта KDE. До начала 2010 года была известна как KDE (сокращение от K Desktop Environment). Построена на основе кроссплатформенного инструментария разработки пользовательского интерфейса Qt. Работает преимущественно под UNIX-подобными операционными системами, которые используют графические подсистемы X Window System и Wayland. Новое поколение технологии KDE 4 частично работает на Microsoft Windows и Mac OS X.



KDE/Plasma

Информация о системе

Справка по модулю Справка

Поиск

Сведения об этой системе

Arch Linux

<https://www.archlinux.org/>

Программы

Версия KDE Plasma: 5.18.0
Версия KDE Frameworks: 5.67.0
Версия Qt: 5.14.1
Версия ядра: 5.5.3-arch1-1
Архитектура: 64-битная

Оборудование

Процессоры: 4 x Intel® Pentium® CPU N3710 @ 1.60GHz
Память: 7,7 ГиБ ОЗУ

Копировать в буфер обмена Копировать текст на английском в буфер обмена

ArchLinux

5.5.3-arch1-1 kernel -- arch x86_64 Arch uptime 0h 50m 29s

cpu

[22%] CPU1 | [16%] CPU2 -- used
[17%] CPU3 | [23%] CPU4 -- freq
1023mhz | cpu1 --
782mhz | cpu2 --
922mhz | cpu3 --
849mhz | cpu4 --

load
plasmashell 0,35 0,51 0,71 5,57
Xorg 3,04
conky 2,28
kwin_x11 1,77

ram
883 MiB / 7,68 GiB -- used
0 B write --

root
6,84 GiB / 49,0 GiB -- used
7,48 GiB / 51,7 GiB home --

net
192.168.0.107 ethernet ip --
192.168.0.198 wifi ip --
0 B up -- down --
0 B down -- up --

lan
192.168.0.107 lan ip --
192.168.0.198 lan ip --
0 B up -- down --
0 B down -- up --

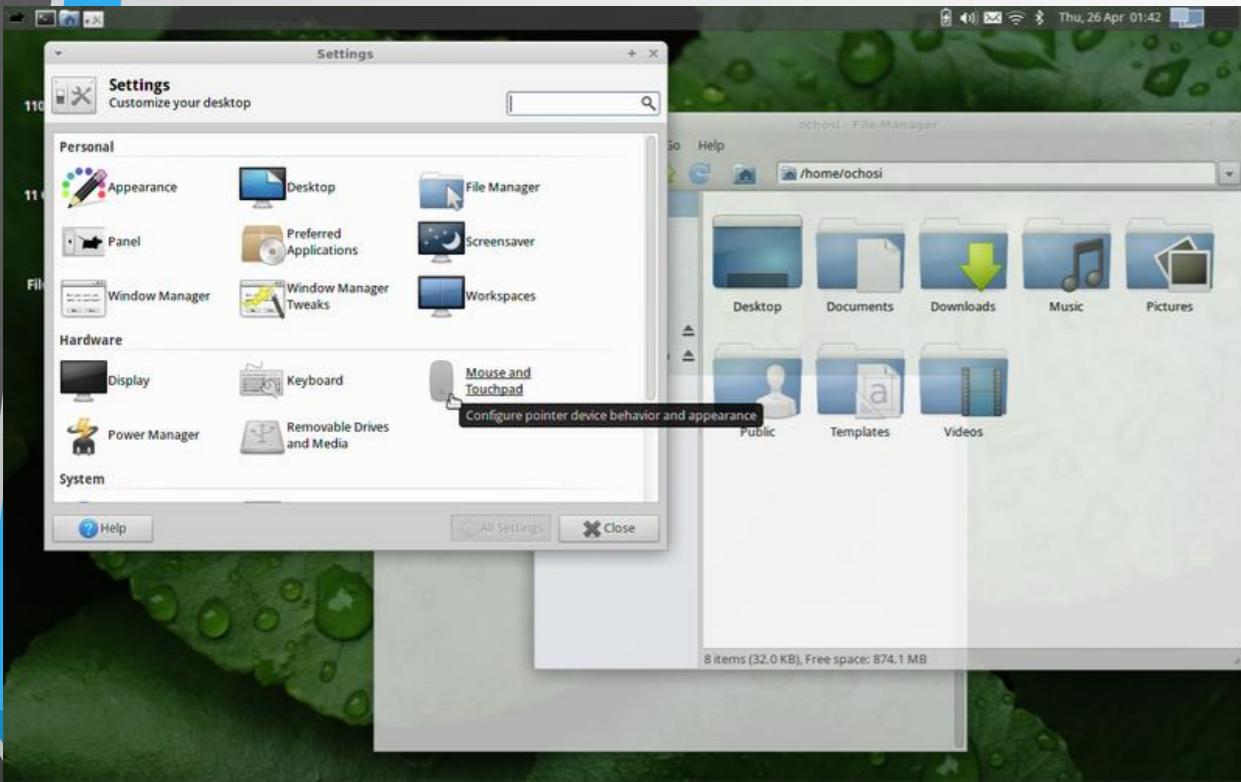
CALENDAR

Февраль 2020

Пн	Вт	Ср	Чт	Пт	Сб	Вс
				1	2	
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

XFCE/LXDE

- **Xfce**— свободная среда рабочего стола для UNIX-подобных операционных систем, таких как GNU/Linux, NetBSD, OpenBSD, FreeBSD, Solaris и т. п. Конфигурация данной среды полностью управляет мышью, конфигурационные файлы скрыты от пользователя
- **LXDE** (англ. *Lightweight X11 Desktop Environment*) — свободная среда рабочего стола для UNIX и других POSIX-совместимых систем, таких как Linux или BSD.



Дистрибутивы Linux

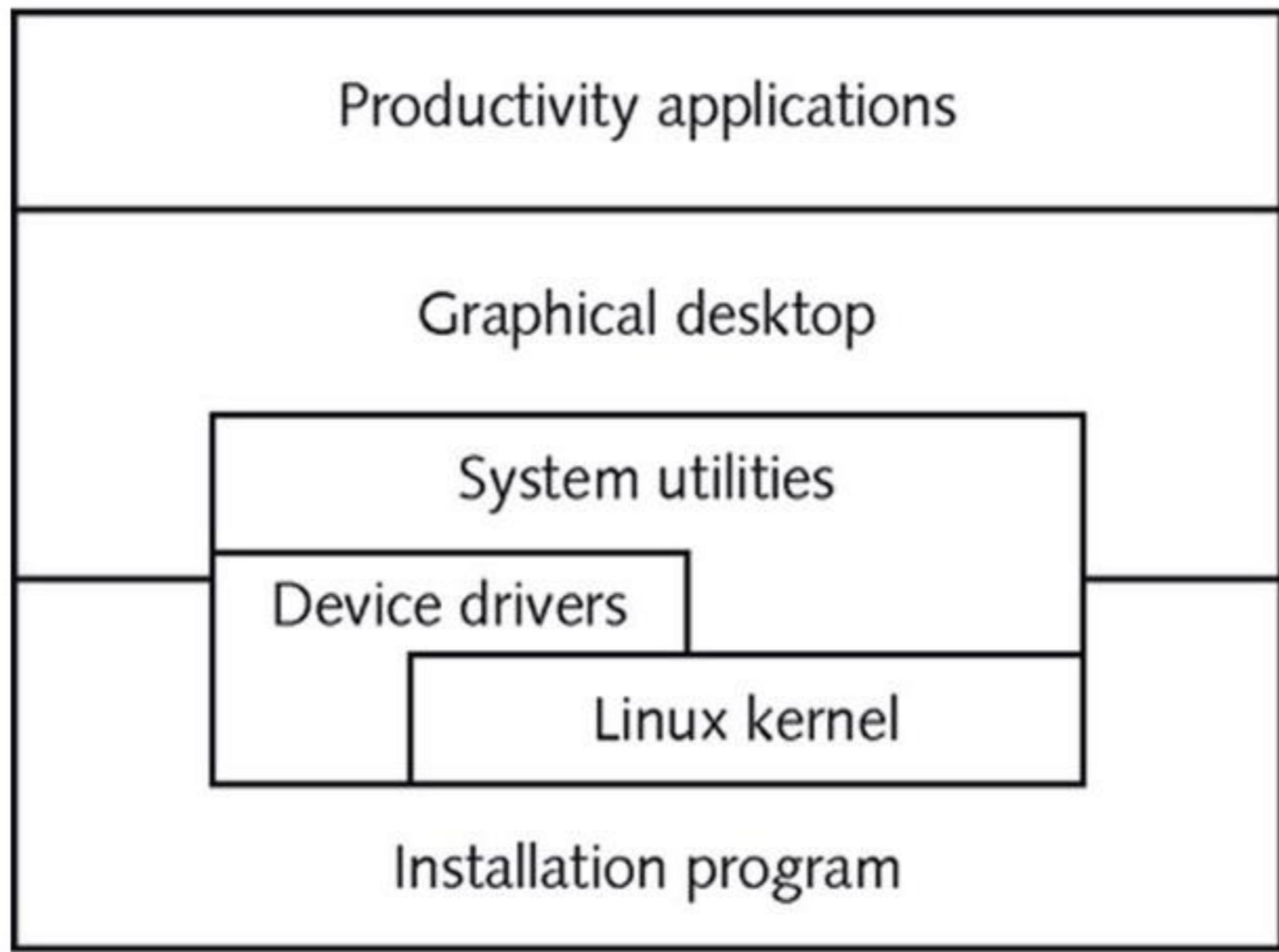
Что такое дистрибутив

- **Дистрибутив операционной системы** — это форма распространения системного программного обеспечения.
- Дистрибутив обычно содержит программы для начальной инициализации системы (инициализация аппаратной части, загрузка урезанной версии системы и запуск программы-установщика), программу-установщик (для выбора режимов и параметров установки) и набор специальных файлов, содержащих отдельные части системы (так называемые пакеты).
- Программа установки позволяет также произвести первичную настройку системы.

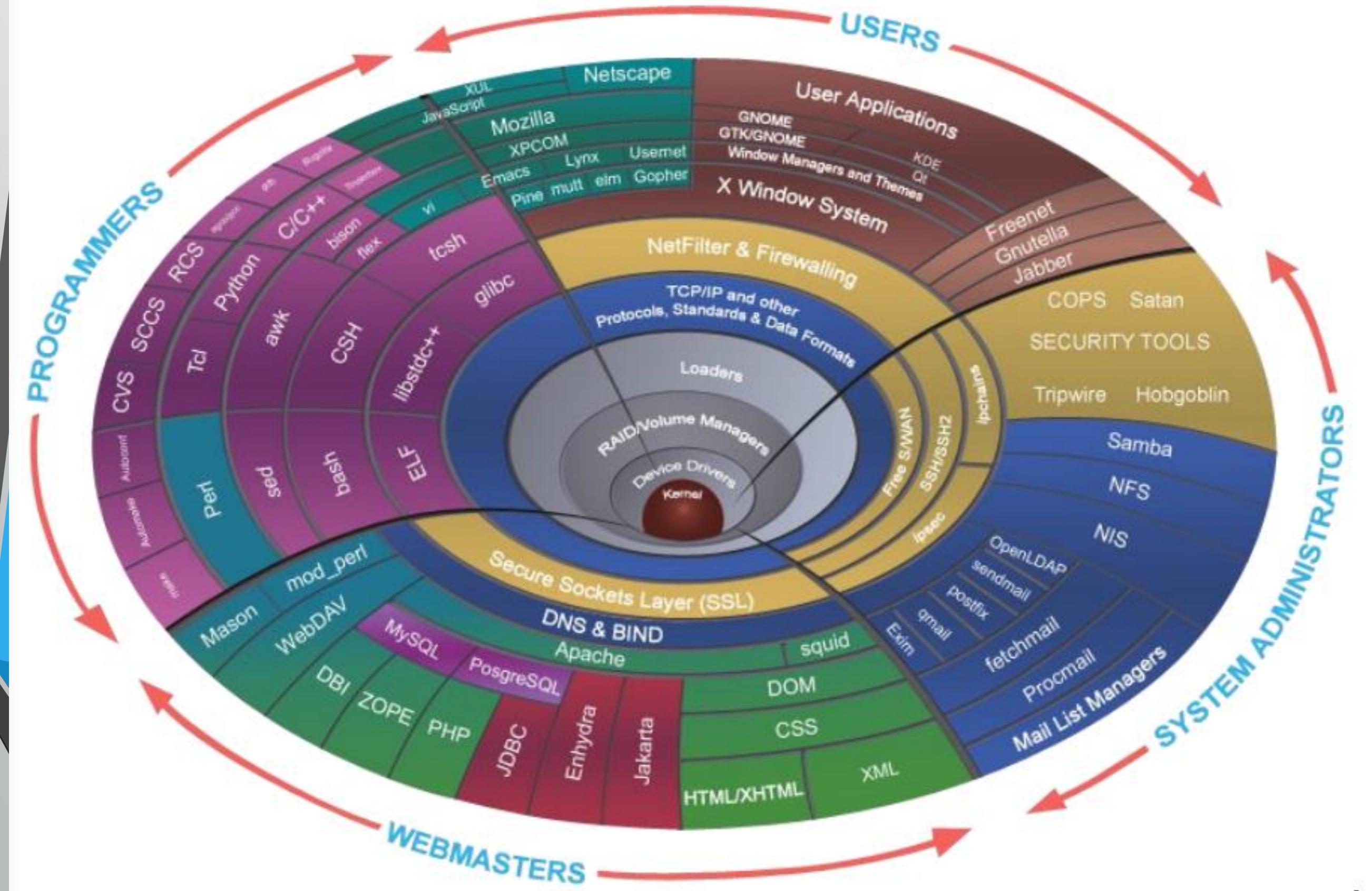
Что такое дистрибутив

- **Дистрибутив операционной системы** — это форма распространения системного программного обеспечения.
- Дистрибутив обычно содержит программы для начальной инициализации системы (инициализация аппаратной части, загрузка урезанной версии системы и запуск программы-установщика), программу-установщик (для выбора режимов и параметров установки) и набор специальных файлов, содержащих отдельные части системы (так называемые пакеты).
- Программа установки позволяет также произвести первичную настройку системы.

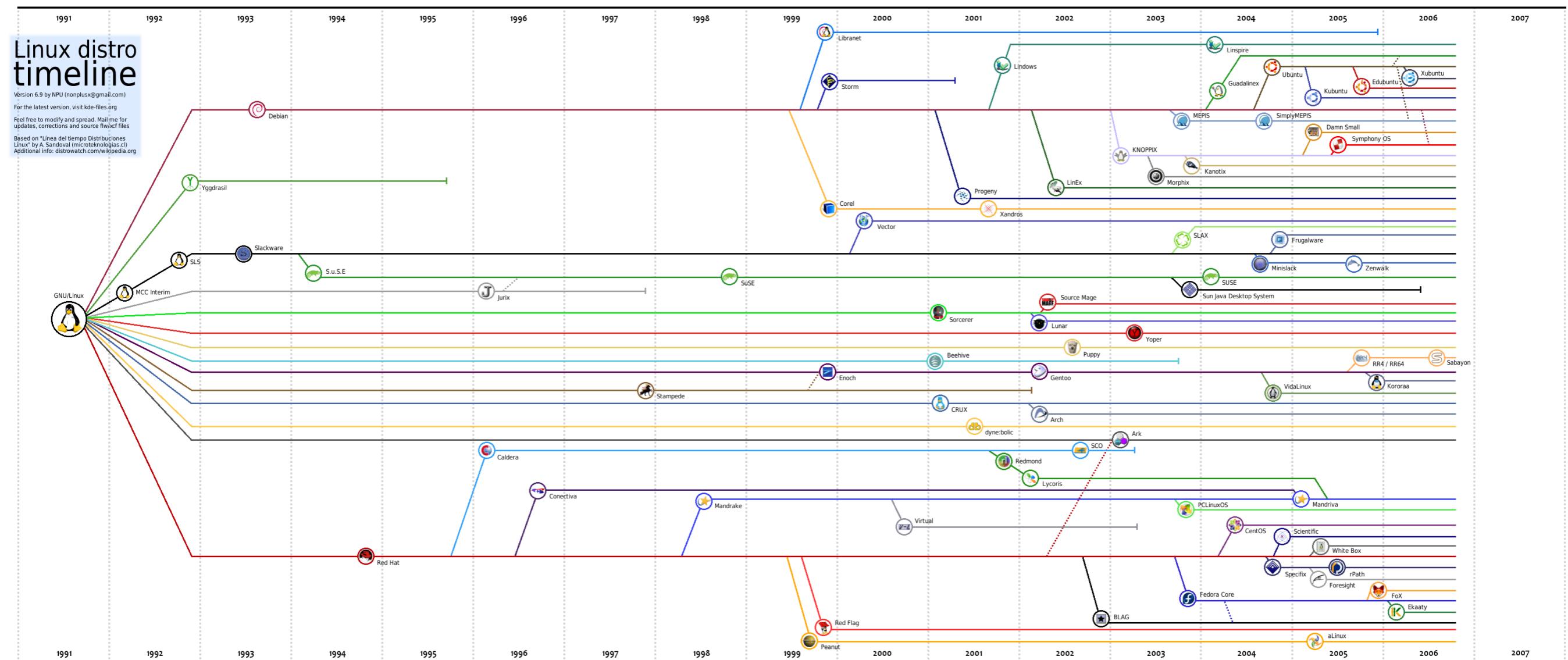
Linux Distribution



Anatomy of a Linux System



Что такое дистрибутив



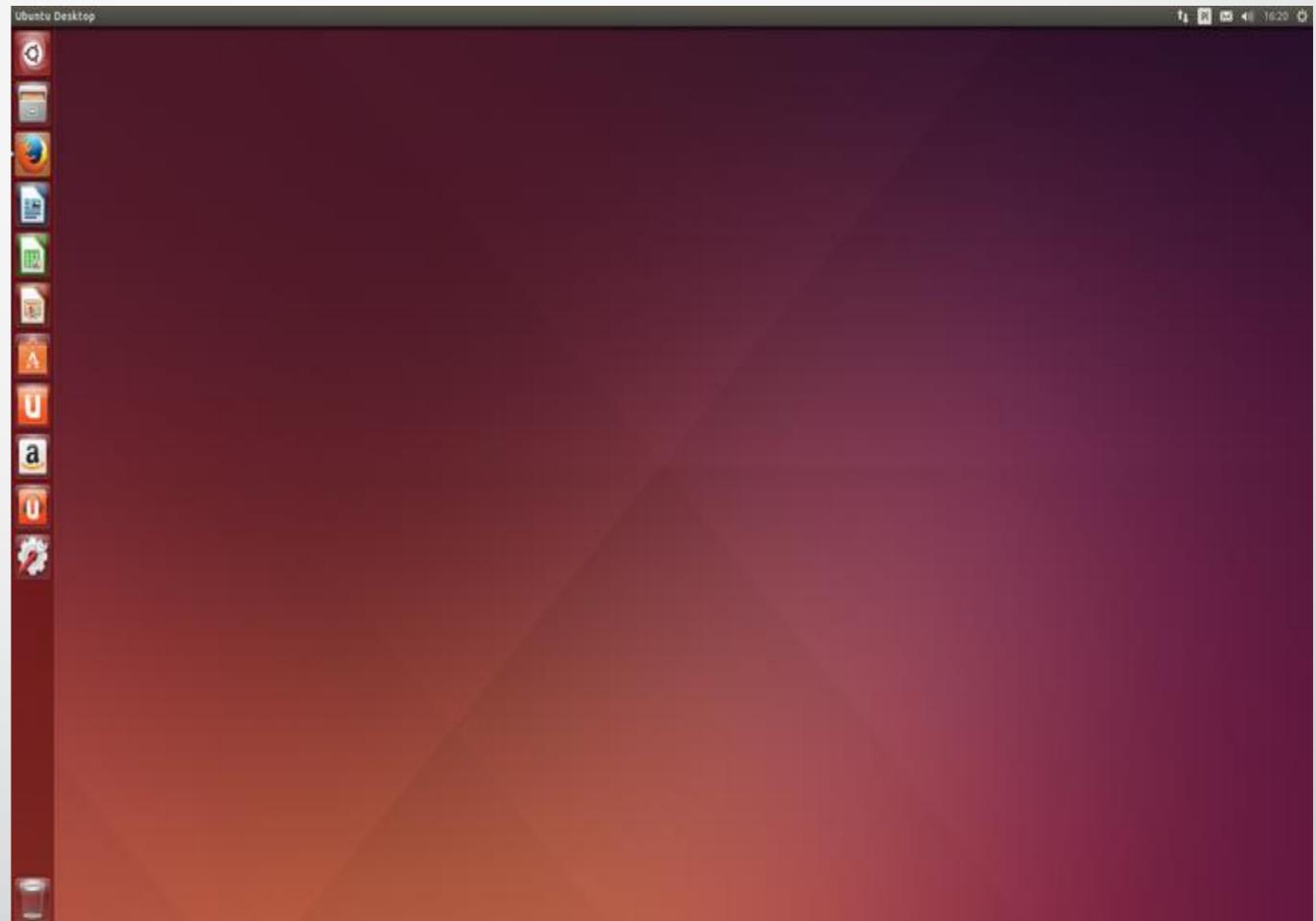
Debian

- <https://www.debian.org/>
- Установщик пакетов apt
- Графическая оболочка по умолчанию - Gnome

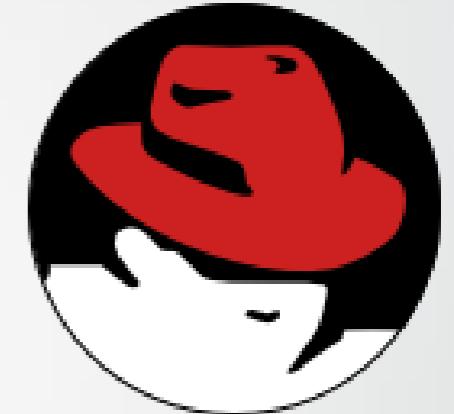


Ubuntu/Kubuntu/Xubuntu/Lubuntu

- <https://ubuntu.ru/>
- Установщик пакетов apt
- Графическая оболочка по умолчанию – Unity (Lomiri) /GNOME



RedHat



- **Red Hat Enterprise Linux** — дистрибутив Linux компании Red Hat.
- Данный дистрибутив позиционируется для корпоративного использования. Новые версии выходят с периодичностью около 3 лет.
- Основная особенность дистрибутива — наличие коммерческой поддержки на протяжении 10 лет, с возможностью продления до 13 лет. Многие производители программного и аппаратного обеспечения включили RHEL в число поддерживаемых ими дистрибутивов Linux.
- Отсутствие поддержки MP3 и DivX (по лицензионным соображениям)
- Платный доступ к бинарным пакетам обновлений (исходные коды доступны)



- **Fedora**— дистрибутив операционной системы GNU/Linux. Этот дистрибутив спонсируется фирмой Red Hat и поддерживается сообществом. Проект служит для тестирования новых технологий, которые в дальнейшем включаются в продукты Red Hat и других производителей. Компания Red Hat не предоставляет поддержку пользователям Fedora, поддержка осуществляется открытым сообществом.



CentOS

- CentOS (Community ENTerprise Operating System) — дистрибутив Linux, основанный на коммерческом Red Hat Enterprise Linux компании Red Hat и совместимый с ним. Согласно жизненному циклу Red Hat Enterprise Linux (RHEL, CentOS 5, 6 и 7 будут поддерживаться «до 10 лет», поскольку они основаны на RHEL. Ранее версия CentOS 4 поддерживалась семь лет.
- Red Hat Enterprise Linux состоит из свободного ПО с открытым кодом, но доступен в виде дисков с бинарными пакетами только для платных подписчиков. Как требуется в лицензии GPL и других, Red Hat предоставляет все исходные коды. Разработчики CentOS используют данный исходный код для создания окончательного продукта, очень близкого к Red Hat Enterprise Linux и доступного для загрузки. Существуют и другие клоны Red Hat Enterprise Linux, созданные на основе этого кода.



Slackware

- **Slackware Linux** — один из старейших дистрибутивов Linux.
- Его иногда называют «самым UNIX'овым». Поклонникам этого дистрибутива приписывают высказывание: «*Если вы выучили Red Hat, то всё, что вы знаете, — это Red Hat, если вы выучили Slackware — вы знаете Linux.*»
- Принцип KISS (Keep it simple stupid, keep it short and simple — «делай короче и проще») — это относится к простоте построения системы, а не к простоте использования.



Gentoo

- **Gentoo Linux** — популярный дистрибутив GNU/Linux с мощной и гибкой технологией Portage, совмещающей в себе возможности конфигурирования и настройки, а также автоматизированную систему управления пакетами.
- Последняя создавалась под влиянием системы управления пакетами во FreeBSD.
- Отличительной особенностью Gentoo является наличие оптимизации под конкретное аппаратное обеспечение.

Имеет пакетный менеджер Portage

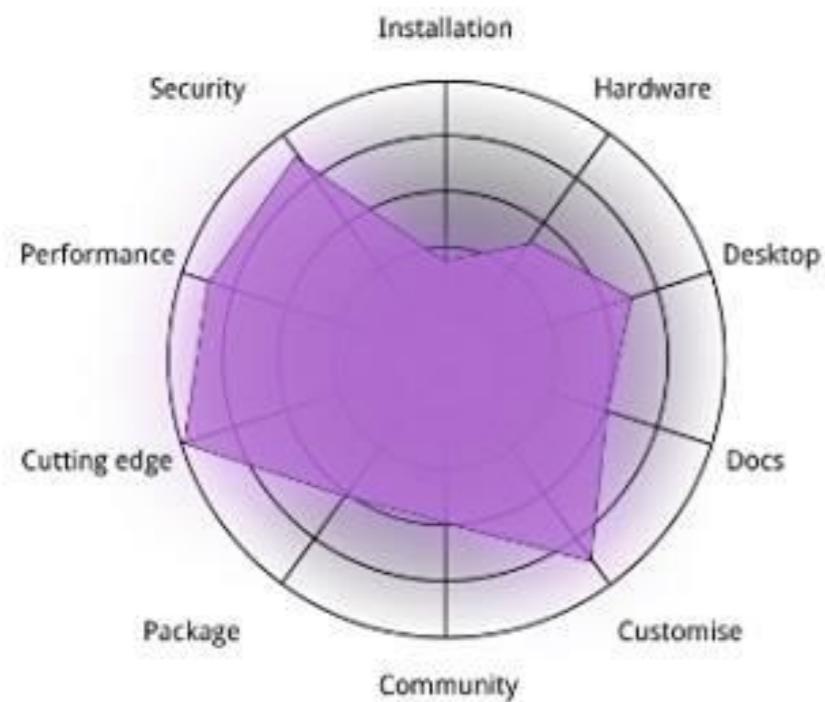
Производные - Chromium OS



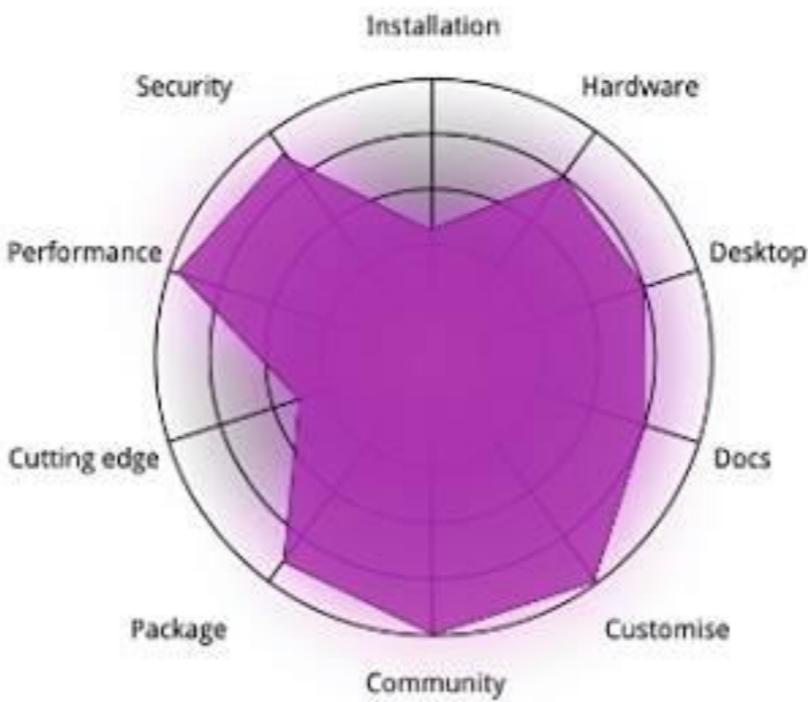
Arch Linux

- **Arch Linux** — независимый дистрибутив GNU/Linux общего назначения, оптимизированный для архитектуры x86-64, который стремится предоставить последние «стабильные» версии программ, следуя модели *rolling release*. По умолчанию пользователю предоставляется минималистичная базовая система, в которую пользователь может добавить то, что ему требуется. Для установки, удаления и обновления пакетов используется пакетный менеджер Растман
- Arch Linux распространяется, в основном, в виде готовых пакетов двоичных файлов, хотя допускает сборку из исходного кода, а пакеты из AUR могут как собираться как на машине пользователя, так и конвертироваться из deb/rpm пакетов в пакеты расстан.

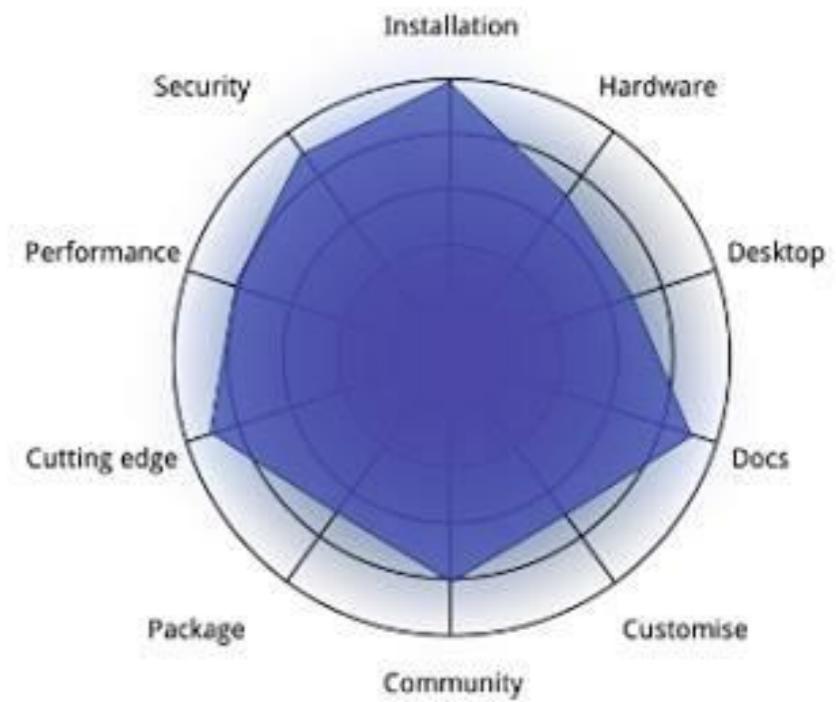
Arch Linux



Debian



Fedora



Linux Mint



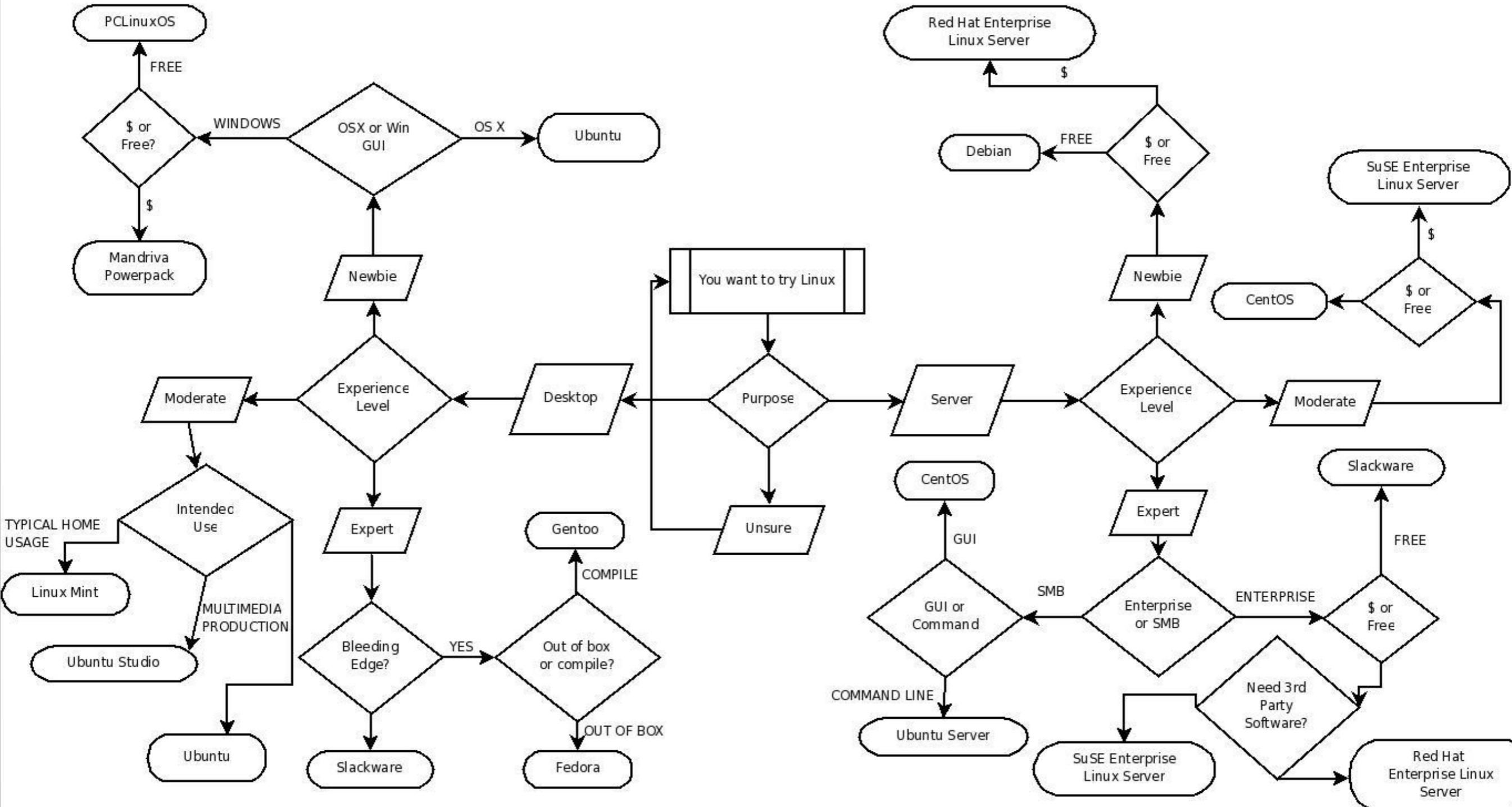
OpenSUSE



Ubuntu



LINUX DISTRIBUTION CHOICE FLOW CHART





Системное программное обеспечение
Лекция 3. Операционные системы *NIX

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ



Системное программное обеспечение

Лекция 4. Конфигурирование ПО

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ

Зависимости пакетов

Если пакет А **зависит** от другого пакета В, то В требуется для правильной работы А. Например, пакет gimp зависит от пакета gimp-data, поскольку необходимо гарантировать, что графическому редактору GIMP доступны необходимые файлы данных.

Если пакет А **рекомендует** другой пакет В, то пакет В предоставляет важную дополнительную функциональность пакету А, которая желательна в большинстве случаев. Например, пакет mozilla-browser рекомендует пакет mozilla-psm, который добавляет поддержку безопасной передачи данных веб-браузеру Mozilla. В то время как пакет mozilla-psm не строго требуется для работы Mozilla, большинство пользователей хотят, чтобы Mozilla поддерживал безопасную передачу конфиденциальных данных (таких как номера кредитных карт).

Зависимости пакетов

Если пакет А **предлагает** другой пакет В, то пакет В предоставляет функциональность, которая может улучшить пакет А, но не является необходимой в большинстве случаев. Например, пакет kmail предлагает пакет gnuPG, содержащий программу для шифрования, которая может использоваться KMail.

Если пакет А **конфликтует** с другим пакетом В, то эти два пакета не могут быть установлены одновременно. Например, пакет fb-music-hi конфликтует с пакетом fb-music-low, поскольку они предоставляют альтернативные наборы музыки для игры Frozen Bubble.

Apt-get

Основные задачи - обеспечение сообщения с удаленными репозиториями пакетов, которыми управляют разработчики ОС, и выполнение базовых операций с ними. Также при помощи этой команды можно обновить локальный кэш пакетов, установить новые или удалить старые пакеты.

Чаще всего apt-get применяется для установки и удаления пакетов, для обновления списка пакетов и самой системы, для обновления кэша и выполнения операций в рабочей системе. В Ubuntu 16.04 все команды можно выполнять при помощи одной команды apt: apt install, apt search и так далее.

Apt-cache

Другой незаменимый инструмент из семейства apt - apt-cache. С его помощью можно получить информацию о доступных пакетах из локального кэша. Например, если необходимо установить определенный пакет или найти подходящий инструмент, то начать поиск стоит именно здесь. Вы легко найдете информацию о доступной версии пакета, прямых и обратных зависимостях.

Aptitude

Объединяет в себе функционал двух предыдущих инструментов. Он может работать не только в командном режиме, но и имеет интерфейс, построенный на базе библиотеки ncurses. При работе в командном режиме aptitude почти полностью копирует команды apt-get и apt-cache. Поэтому мы не будем отдельно обсуждать этот инструмент. По идеи вы всегда можете заменить apt-get и apt-cache на aptitude.

Инструменты работы с пакетами

Dpkg

Основное отличие dpkg от рассмотренных инструментов - это возможность работы напрямую с пакетами deb, т.е. без удаленных репозиториев. По идее, все вышеописанные инструменты используют dpkg для своей работы.

В отличии от инструментов apt-* dpkg не устанавливает автоматически требуемые зависимости. Основные задачи dpkg - работа с .deb пакетами напрямую и получение полной информации о пакете и его структуре. Хотя при помощи dpkg можно получить информацию об установленных пакетах в системе, его первостепенная задача - это работа с отдельными пакетами.

Инструменты работы с пакетами

Tasksel

Tasksel стоит отдельно в этом ряду. Эта программа объединяет пакеты, основываясь на выполнении определенной задачи.

Например, LAMP можно установить одной командой при помощи Tasksel (для установки LAMP с помощью APT требуется установить сначала Apache, потом MySQL, потом PHP). Для этого необходимо выполнить команду:

```
sudo tasksel install lamp-server
```

Виды репозиториев:

Base repository - основное хранилище, содержит все пакеты, но обновляется достаточно редко, обычно одновременно с выходом минорной версии дистрибутива, например, Debian 9.6, Debian 9.7 и т.д.

Security updates - обновления безопасности, являются критически важными для функционирования системы и крайне желательны к установке.

Stable updates - стабильные обновления, не являющиеся обновлениями безопасности, могут исправлять некритические ошибки в ПО или незначительно расширять его функционал в рамках текущей версии.

Stable backports - стабильное ПО с обратной совместимостью, содержит прекомпилированные для текущего дистрибутива самые свежие версии ПО, которые разрабатываются в рамках новой версии, позволяет использовать более новые версии программ, не подвергая стабильность дистрибутива угрозам при использовании пакетов тестируемой или нестабильной ветки.

Разделы репозиториев (Debian):

main - содержит пакеты, которые полностью совместимы с "Критериями Debian по определению Свободного ПО"

non-free - распространяемое без ограничений ПО, которое не соответствует или не полностью соответствует принципам свободного ПО по версии FSF (*Free Software Foundation*, Фонд свободного ПО)

contrib - свободное ПО, которое требует для работы несвободные компоненты, например, бинарные модули драйверов, прошивок ROM и т.д., либо требует ПО, имеющее собственника, скажем несвободную версию Java от Oracle.

Разделы репозиториев (Ubuntu):

main - также, как и в Debian содержит свободные пакеты, поддерживаемые компанией Canonical

restricted - несвободное ПО, поддерживаемое Canonical

universe - свободное ПО, поддерживаемое сообществом

multiverse - несвободное ПО, поддерживаемое сообществом.

Управление репозиториями

Список подключенных репозиториев хранится в **/etc/apt/sources.list**

Типичная запись репозитория выглядит как строка со следующей структурой:

```
deb http://deb.debian.org/debian buster main contrib  
non-free
```

Строка начинается с **deb**, который обозначает репозиторий с двоичными пакетами или **deb-src** для репозиториев с исходным кодом,

Далее идет адрес репозитория и имя выпуска, для Debian это **buster**, для Ubuntu - **bionic**, затем следует перечисление подключенных разделов, в указанном примере подключены все три.

Управление репозиториями

Также в Debian можно использовать вместо имен классы выпусков:

- **stable** - текущий выпуск дистрибутива,
- **oldstable** - предыдущий выпуск,
- **testing** - разрабатываемый выпуск,
- **unstable** - нестабильный выпуск, он же имеет собственное имя **sid**.

Можно везде прописать **testing**, получив аналог популярных ныне **rolling release** дистрибутивов, которые не имеют закрепленной версии и всегда предоставляют наиболее свежий срез пакетов, зачастую ценой стабильности.

Для добавления собственных источников пакетов предназначена директория **/etc/apt/sources.list.d/** в которой следует располагать файлы с адресами источников и обязательным расширением **.list**. Хотя их можно прописать и в основной файл, но это считается дурным тоном.

Низкоуровневый менеджер пакетов Dpkg

Основой системы управления пакетами в Debian является **dpkg** - достаточно низкоуровневая утилита, предназначенная для получения информации, установки и удаления пакетов. Назвать ее менеджером пакетов в современном смысле этого слова достаточно затруднительно, **dpkg** не умеет работать с репозиториями и разрешать зависимости. Но тем не менее этот инструмент достаточно широко используется по сей день и навыки работы с ним необходимо иметь каждому администратору.

Среда окружения

Среда окружения (англ. Environment) — в информатике совокупность значений системных переменных, путей, открытых файловых дескрипторов и других ресурсов операционной системы, передаваемые процессу (программе) при его запуске.

В различных операционных системах состав среды окружения может сильно различаться.

Среда окружения

К объектам среды окружения обычно относят:

- системные переменные
- текущие пути на различных дисках (в случае поддержки нескольких дисков операционной системой)
- точка монтирования каталогов (в том числе корневого), используется в unix-подобных операционных системах для обеспечения режима "тюрьмы" (англ. jail)
- связь стандартных потоков ввода-вывода с файловыми хэндлерами или устройствами (используется для перенаправления ввода-вывода)

Среда окружения

К объектам среды окружения обычно относят:

- ограничения на количество одновременно открытых файлов, стеков и т.д.
- набор прав (обычно соответствует правам пользователя, запустившего процесс, но может изменяться как в сторону большего набора прав, так и в сторону ужесточения)
- дисковые квоты, ограничение на максимальный объём оперативной памяти, загрузки процессоров и т.д.
- значения показателей использования ресурсов, получаемые от родительского процесса (на некоторых системах)

Переменные среды

Операционная система поддерживает специальный вид ресурсов, называемых переменные среды (environment variables). Эти переменные представляют собой пару ИМЯ - ЗНАЧЕНИЕ. Имя может начинаться с буквы и состоять из букв, цифр и символов подчеркивания.

Для подстановки значения переменной в командную строку перед именем переменной ставится знак \$:

```
$ echo $USER  
guest
```

В случае, если переменная не установлена, возвращается пустая строка.

Переменные среды

Для установки значения переменной используется оператор присваивания (в случае Bourne-подобных оболочек):

```
$ TEST=test
```

или встроенный оператор set (в случае C-подобных):

```
$ set TEST=test
```

Команда set без аргументов выводит список значений всех переменных, установленных в среде:

```
$ set
```

```
COLUMNS=197
```

```
CVS_RSH=ssh
```

```
HOSTNAME=myhost
```

```
HOSTTYPE=i686
```

```
IFS=$' \t\n'
```

```
INPUTRC=/etc/inputrc
```

```
KDEDIR=/usr
```

```
KDEDIRS=/home/guest/.local/
```

```
LOGNAME=guest
```

```
....
```

Переменные среды

Переменные могут носить характер локальный для данного процесса или глобальный для сессии. Задать локальные значения для переменных можно, предварив ими вызов команд:

```
$ TEST=test1 sh -c 'echo $TEST'  
test1
```

Переменные среды

Оценить содержимое набора переменных для сессии можно, вызвав встроенную команду интерпретатора env, в случае Bourne-подобных интерпретаторов (sh, ksh, bash, zsh, pdksh...), и printenv в случае использования интерпретаторов клона C-Shell (csh, tcsh...):

```
$ env
HOSTNAME=myhost
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
KDE_NO_IPV6=1
SSH_CLIENT=172.16.0.9 50487 22
QTDIR=/usr/lib/qt-3.3
QTINC=/usr/lib/qt-3.3/include
SSH_TTY=/dev/pts/6
USER=guest
MOZILLA_CERTIFICATE_FOLDER=/home/guest/.evolution/
KDEDIR=/usr
```

Переменные среды

Наборы команд Shell могут компоноваться в командные файлы, называемые скриптами, где в первой строке в специального вида комментарии указывается командный интерпретатор для исполнения этого набора. Например, создадим в текстовом редакторе файл с названием `test`, следующего содержания:

```
#!/bin/sh
```

```
echo Переменная TEST:  
echo $TEST
```

Данная программа будет выводить на стандартный вывод текстовое сообщение «Переменная TEST:» и значение переменной TEST, если оно задано. Запустить его из командной строки можно, передав его в качестве параметра командному интерпретатору:

```
$ sh test
```

```
Переменная TEST:
```

Переменные среды

Придать переменной глобальный характер можно при помощи оператора **export** (Bourne) или **setenv** (C-SHELL):

```
$ export TEST=test1  
$ sh test
```

Переменная TEST:

```
test1
```

Задать локальные значения переменных для выполнения данной программы можно, предварив ими вызов команд:

```
$ TEST=test2 sh test
```

Переменная TEST:

```
test2
```

Удаление переменных среды производится при помощи оператора **unset**:

```
$ unset TEST
```

Переменные среды

- SHELL — домашняя оболочка.
- PATH — список каталогов, просматриваемых при поиске исполняемых файлов.
- MANPATH — список каталогов, просматриваемых при поиске файлов системного руководства man
- IFS — разделители полей.
- LPDEST — принтер, используемый по умолчанию, если данная переменная не установлена, используются установки системы.
- EDITOR — редактор, используемый по умолчанию.
- VISUAL — режим редактирования командной строки.
- PS1 или prompt — первичное приглашение shell, выдаваемое в поток стандартного вывода в интерактивном режиме.
- PS2 — вторичное приглашение, выдаваемое в поток стандартного вывода в интерактивном режиме при вводе символа перевода строки в незавершенной команде.

Переменные среды

- TERM — тип используемого терминала.
- PAGER — команда, используемая man для просмотра страниц руководства.
- TZ — часовой пояс.
- LINES — количество строк, помещающихся на экране.
- COLUMNS — количество символов, помещающихся в столбце.
- HOME — домашний каталог, используемый, в частности, командой cd
- LOGNAME — ваше входное имя.

Окружение

В разёртывании программного обеспечения, **окружение** или **ярус** является компьютерной системой в которой компьютерная программа или компонент программного обеспечения развертывается и выполняется.

В простом случае, такое развертывание и немедленное выполнение программы на той же машине, может выполняться в единственном окружении, однако при промышленной разработке используется разделение на development окружение ('окружение разработчика', где делаются исходные изменения) и production окружение (которое используют конечные пользователи); часто с промежуточными этапами ('stages') посередине. Этот структурированный процесс управления релизами может иметь фазы deployment (rollout, 'развертывание', 'выкатка'), testing ('тестирование'), и rollback ('откат') в случае проблем.

Окружение

Окружения могут существенно отличаться в размерах: development окружение это обычно рабочая станция отдельного разработчика, в то время как production окружение может быть сетью множества географически разнесённых машин в случае с дата-центрами, или виртуальными машинами в случае с облачными решениями.

Код, данные и конфигурация могут быть развёрнуты параллельно, и нет необходимости связи с соответствующим ярусом — например, pre-production код может подсоединяться к production БД.

Архитектуры окружения

Архитектуры разёртывания существенно разнятся, но в целом, ярусы начинаются с development (DEV) и заканчиваются production (PROD). Распространённой 4-х ярусной архитектурой является каскад ярусов deployment, testing, model, production (DEV, TEST, MODL, PROD) с деплоем софта на каждом ярусе по очереди.

Quality Control (QC), для приёмочного тестирования; песочница или экспериментальное окружение (EXP), для экспериментов не предназначенных для передачи в продакшн; и Disaster Recovery ('аварийное восстановление'), для предоставления возможности немедленного отката в случае проблемы с продакшеном.

deployment, testing, acceptance and production (DTAP)

Архитектуры окружения

Такая разбивка в частности подходит для серверных программ, когда сервера работают в удаленных дата-центрах; для кода который работает на конечных устройствах пользователя, например приложений (apps) или клиентов, последний ярус обозначают как окружение пользователя (USER) или локальное окружение (LOCAL).

Архитектуры окружения

- Local - Компьютер разработчика
- Development/Thunk - Сервер разработки, выступающий как песочница, где разработчик может выполнить unit-тестирование
- Integration - Основа для построения CI или для тестирования сайд-эффектов разработчиком
- Testing/Test/QC/Internal Acceptance - Окружение, в котором выполняется тестирование интерфейса. Команда контроля качества проверяет, что новый код не будет иметь влияния на существующую функциональность системы после деплоя нового кода в тестовое окружение.
- Staging/Stage/Model/Pre-production/External-Client Acceptance/Demo - Зеркало продакшен-окружения
- Production/Live - Серверы конечных пользователей/клиентов



Системное программное обеспечение

Лекция 4. Конфигурирование ПО

Миронов Антон Николаевич
старший преподаватель кафедры МОСИТ