# UNIX и UNIX-подобные системы

# История UNIX и Linux

- **ОС UNIX** появилась в конце 60-х годов как операционная система для мини-ЭВМ PDP-7. Активное участие в разработке приняли Кеннет Томсон и Деннис Ритчи.
- Особенностями ОС UNIX стали: многопользовательский режим, новая архитектура файловой системы и др.
- **В 1973** году большая часть ядра ОС была переписана на новом языке С.
- **С 1974** года ОС UNIX распространяется в университетах США в исходных кодах


# Бесплатные версии ОС семейства UNIX

- Существует большое количество бесплатных версий UNIX.
- FreeBSD, NetBSD, OpenBSD – варианты, разрабатываемые на основе ОС BSD.
- Наиболее популярное семейство бесплатных UNIX-систем – это системы семейства Linux. Первый вариант Linux был разработан Линусом Торвальдсом в 1991 г. В настоящее время существует несколько вариантов Linux: Ubuntu, Mandriva и др.

# Задачи LINUX

Простота и совместимость.

Например, на самом нижнем уровне файл должен представлять собой просто набор байтов. Наличие различных классов файлов для последовательного и произвольного доступа, доступа по ключу, удаленного доступа и т. д. (как это реализовано на мэйнфреймах) просто является помехой. А если команда

**ls A***

означает вывод списка всех файлов, имя которых начинается с буквы «А», то команда

**rm A***

должна означать удаление всех файлов, имя которых начинается с буквы «А», а не одного файла, имя которого состоит из буквы «А» и звездочки. Эта характеристика иногда называется *принципом наименьшей неожиданности {principle of least surprise)*.

# Задачи LINUX

**Мощность и гибкость**

Это означает, что в системе должно быть небольшое количество базовых элементов, которые можно комбинировать бесконечным числом способов, чтобы приспособить их для конкретного приложения. Одно из основных правил системы Linux заключается в том, что каждая программа должна выполнять всего одну функцию — и делать это хорошо. То есть компиляторы не занимаются созданием листингов, так как другие программы могут лучше справиться с этой задачей.

# Задачи LINUX

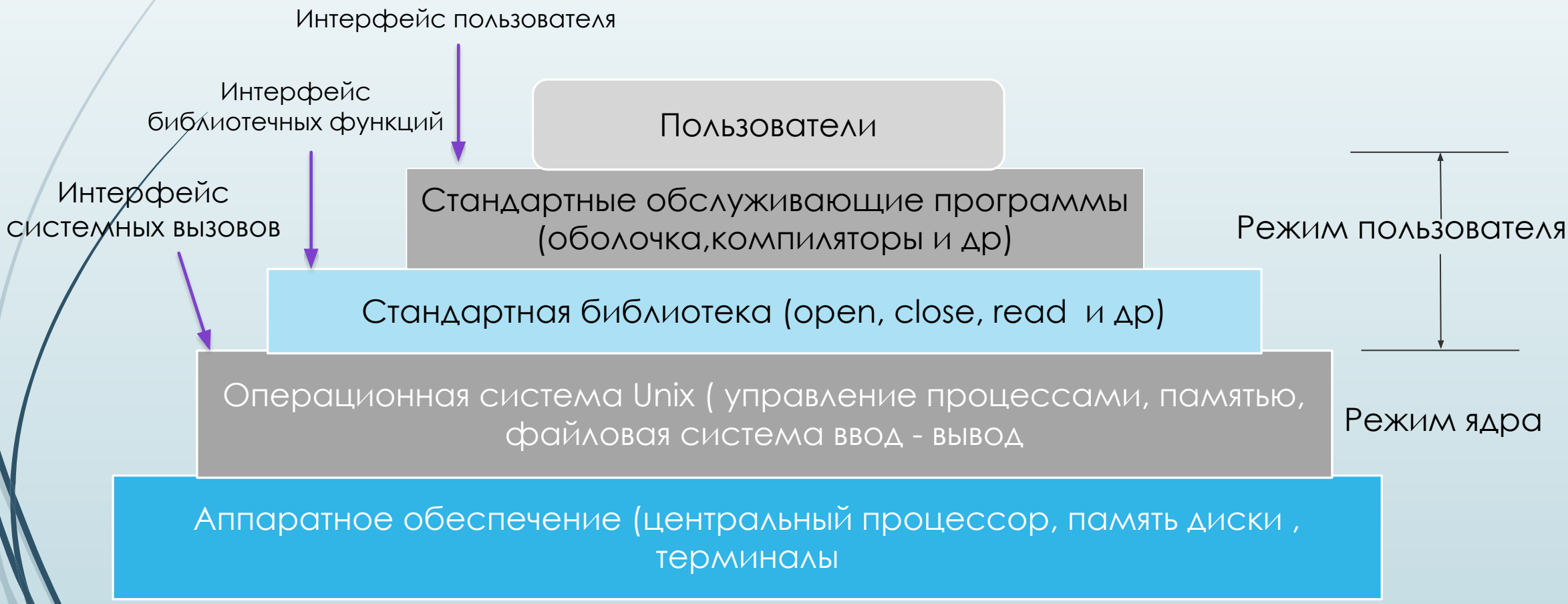
3. Отсутствие избыточности

Зачем писать **copy**, когда достаточно **cp**? Чтобы получить список всех строк, содержащих строку «ard» из файла/, программист в операционной системе Linux вводит команду

**grep ard f**

Противоположный подход состоит в том, что программист сначала запускает программу grep (без аргументов), после чего программа grep приветствует программиста фразой: «Здравствуйте, я grep. Я ищу шаблоны в файлах. Пожалуйста, введите ваш шаблон». Получив шаблон, программа grep запрашивает имя файла. Затем она спрашивает, есть ли еще какие-либо файлы. Наконец, она выводит  резюме того, что она собирается делать, и спрашивает, все ли верно. Хотя такой тип пользовательского интерфейса может быть удобен для начинающих пользователей, он бесконечно раздражает опытных программистов.

# Интерфейсы системы Linux



# Интерфейсы системы Linux

Графический интерфейс  пользователя создает среду рабочего стола — знакомую нам метафору с окнами, значками, каталогами, панелями инструментов, а также возможностями перетаскивания. Популярными средами рабочего стола для Linux являются GNOME (GNU Network Object Model Environment) и KDE (K Desktop Environment).

Графические интерфейсы пользователя в Linux поддерживает оконная система X Windowing System, которую обычно называют XII (или просто X).

# Оболочка

- Несмотря на то что Linux имеет графический интерфейс пользователя, большинство программистов и продвинутые пользователи по-прежнему предпочитают интерфейс командной строки, называемый оболочкой (**shell**). Они часто запускают одно или несколько окон с оболочками из графического интерфейса пользователя и работают в них. Интерфейс командной строки оболочки значительно быстрее в использовании, существенно мощнее, прост в расширении и не грозит пользователю туннельным синдромом из-за необходимости постоянно пользоваться мышью

Оболочка **bash** основана на оригинальной оболочке системы UNIX, которая называется оболочкой Бурна (Bourne shell), и фактически даже ее название вляется сокращением от Bourne Again SHell. Используется и множество других оболочек (ksh, csh и т. д.), но bash является оболочкой по умолчанию в большинстве Linux-систем.

Когда оболочка запускается, она инициализируется, а затем выводит на экран символ приглашения к вводу (обычно это знак процента или доллара) и ждет, когда пользователь введет командную строку.

# Оболочка bash

У команд могут быть аргументы, которые передаются запускаемой программе в виде текстовых строк. Например, командная строка

**cp src dest**

запускает программу **cp** с двумя аргументами: **src** и **dest**. Эта программа интерпретирует первый аргумент как имя существующего файла. Она копирует этот файл и называет эту копию **dest**.

Не все аргументы являются именами файлов. В строке

**head -20 file**

первый аргумент **-20** дает указание программе head напечатать первые 20 строк файла **file** (вместо ринятых по умолчанию 10 строк). Управляющие работой команды или указывающие дополнительные значения аргументы называются флагами и по соглашению обозначаются знаком тире. Тире требуется, чтобы избежать двусмысленности — поскольку, например, команда **head 20 file** вполне законна. Она дает указание программе head вывести первые 10 строк файла с именем **20**, а затем вывести первые 10 строк второго файла **file**. Большинство команд Linux-систем могут принимать несколько флагов и аргументов.

Чтобы было легче указывать группы файлов, оболочка принимает так называемые **волшебные символы (magic charecters),** иногда называемые также **групповыми (wild cards).** Например, символ звездочка означает все возможные текстовые строки, так что строка

**ls *.c**

дает указание программе **ls** вывести список всех файлов, имя которых оканчивается на **.c**. Если существуют файлы x.c, y.c и z.c, то данная команда эквивалентна команде

**ls x.c y.c z.c**

Другим групповым символом является вопросительный знак, который заменяет один любой символ. Кроме того, в квадратных скобках можно указать множество символов, из которых программа должна будет выбрать один. Например, команда

**ls [ape]***

выводит все файлы, имя которых начинается с символов «**a**», «**p**» или «**e**».

Такая программа, как оболочка, не должна открывать терминал (клавиатуру и монитор), чтобы прочитать с него или сделать на него вывод. Вместо этого запускаемые программы автоматически получают доступ для чтения к файлу, называемому **стандартным устройством ввода (standard input)**, а для записи — к файлу, называемому **стандартным устройством вывода (standard output)**, и к файлу, называемому **стандартным устройством для вывода сообщений об ошибках (standard error).**

# Утилиты Linux

Пользовательский интерфейс командной строки (оболочки) Linux состоит из большого числа стандартных служебных программ, называемых также утилитами. Эти программы можно разделить на шесть следующих категорий:

**1. Команды управления файлами и каталогами.**

**2. Фильтры.**

**3. Средства разработки программ, такие как текстовые редакторы и компиляторы.**

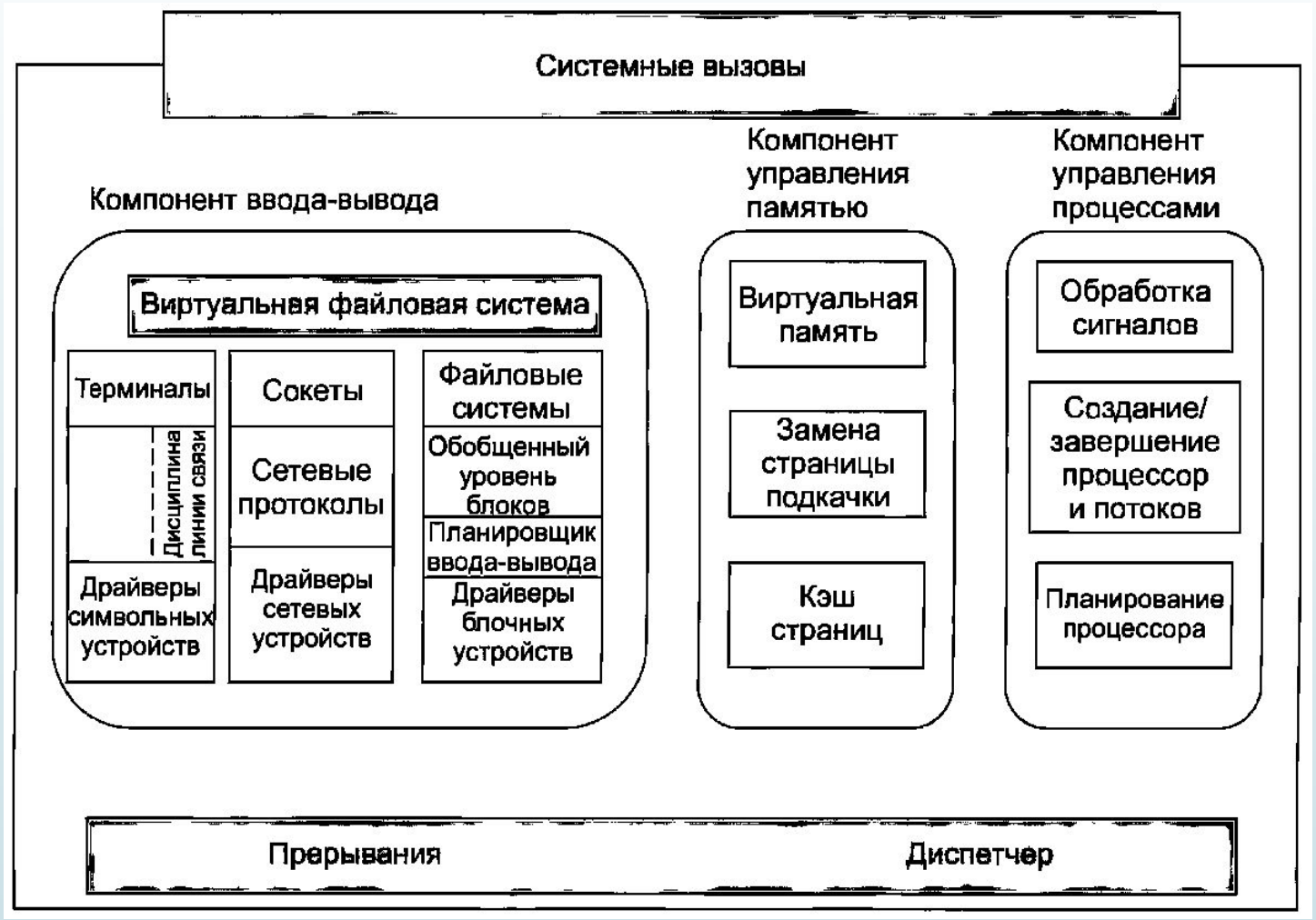
**4. Текстовые процессоры.**

**5. Системное администрирование.**

**6. Разное.**

- **cat** - Конкатенация нескольких файлов в стандартный выходной поток
- **chmod** - Изменение режима защиты файла
- **cp** - Копирование файлов
- **cut** - Вырезание колонок текста из файла
- **grep** - Поиск определенного шаблона в файле
- **head** - Извлечение из файла первых строк
- **ls** - Распечатка каталога
- **make** - Компиляция файлов для создания двоичного файла

- **mkdir** - Создание каталога
- **od** - Восьмеричный дамп файла
- **paste** - Вставка колонок текста в файл
- **pr** - Форматирование файла для печати
- **rm** - Удаление файлов
- **rmdir** - Удаление каталога
- **sort** - Сортировка строк файла по алфавиту
- **tail** - Извлечение из файла последних строк
- **tr** - Преобразование символов из одного набора в другой


Системные вызовы
Компонент ввода-вывода
Виртуальная файловая система
Терминалы
Дисциплина линии связи
Драйверы символьных устройств
Сокеты
Сетевые протоколы
Драйверы сетевых устройств
Файловые системы
Обобщенный уровень блоков
Планировщик ввода-вывода
Драйверы блочных устройств
Компонент управления памятью
Виртуальная память
Замена страницы подкачки
Кэш страниц
Компонент управления процессами
Обработка сигналов
Создание/ завершение процессор и потоков
Планирование процессора
Прерывания
Диспетчер

# Структура ядра

- Ядро работает непосредственно с аппаратным обеспечением и обеспечивает взаимодействие с устройствами ввода-вывода и блоком управления памятью, а также управляет доступом процессора к ним. Нижний уровень ядра состоит из обработчиков прерываний (которые являются основным средством  взаимодействия с устройствами) и механизма диспетчеризации на низком уровне.

# Процессы в Linux

- Процессы могут создавать  дочерние процессы, в результате чего формируется дерево процессов. Управление процессами в Linux отличается от других UNIX-систем в том плане, что Linux рассматривает каждую исполняемую сущность — однопоточный процесс, каждый поток многопоточного процесса или ядро — как отдельную задачу. Процесс (или задача в общем случае) представляется двумя основными компонентами — структурой задачи и дополнительной информацией (описывающей адресное пространство пользователя).

- Первый постоянно находится в памяти, а данные второго могут выгружаться на диск. Процесс создается посредством дублирования структуры задачи процесса, после чего производится настройка информации образа памяти (ставится указатель на образ памяти родителя). Настоящие копии страниц образа памяти создаются только в том случае, когда совместное использование не разрешено, а модификация памяти требуется. Этот механизм называется копированием при записи. Для планирования применяется алгоритм, основанный на приоритетах, который отдает предпочтение интерактивным процессам.
- Модель памяти состоит из трех сегментов для каждого процесса: текста, данных и стека. Для управления памятью применяется страничная подкачка. Состояние каждой страницы отслеживается в карте памяти, а страничный демон поддерживает достаточное количество свободных страниц при помощи модифицированного алгоритма часов.

# Ввод-вывод в Linux

- Доступ к устройствам ввода-вывода осуществляется при помощи специальных файлов, у каждого из которых есть старший номер устройства и младший номер устройства. Для снижения числа обращений к диску в блочных устройствах ввода-вывода применяется кэширование дисковых блоков. Символьный ввод-вывод может осуществляться в необработанном режиме, потоки символов можно модифицировать при помощи дисциплин линий связи. Сетевые устройства работают несколько иначе, с ними связываются модули сетевых протоколов (для обработки потока сетевых пакетов по дороге к процессу пользователя и обратно).

# Файловая система Linux

- Файловая система иерархическая, с файлами и каталогами. Все диски монтируются в единое дерево каталогов, начинающееся в едином корне. Отдельные файлы могут быть связаны с любым каталогом файловой системы. Чтобы пользоваться файлом, его нужно сначала открыть — при этом выдается дескриптор файла,  который затем используется при чтении этого файла и записи в него.
- Внутри файловая система использует три основные таблицы: таблицу дескрипторов файлов, таблицу описания открытых файлов и таблицу i-узлов. Таблица i-узлов является наиболее важной из этих таблиц. В ней содержится вся административная информация о файле и местоположении его блоков.