

# Система Linux

# История

- Linux – современная, свободно распространяемая ОС, основанная на стандартах UNIX.
- 1983: Ричард Столмен (Richard Stallman) начал проект GNU, а в 1985 г. основал Free Software Foundation. Основная цель – разработка UNIX-подобной системы, которая состояла бы только из свободно распространяемого программного обеспечения
- Впервые Linux разработана как небольшое, но самодостаточное ядро ОС в 1991 Линусом Торвальдсом (Linus Torvalds), с основной целью добиться совместимости с UNIX.
- История Linux – это история многолетнего (удаленного) взаимодействия пользователей всего мира, которое осуществляется почти исключительно через Интернет.
- Система была спроектирована с целью эффективного и надежного использования на распространенных персональных компьютерах, но она также используется и на многих других аппаратных платформах.
- Основная часть ОС Linux – полностью оригинальна, но на ней может также исполняться значительная часть свободно распространяемого программного обеспечения для UNIX, и в результате имеется оригинальная свободно распространяемая совместимая с UNIX система, в которой нет ведомственного (proprietary) кода.

# Ядро Linux

- Версия 0.01 (май 1991) не содержала сетевых средств, выполнялась только на 80386-совместимых Intel – процессорах, имела очень ограниченный набор драйверов устройств и поддерживала только файловую систему MINIX (MINIX – операционная система типа UNIX, разработанная Andrew Tannenbaum).
- Linux 1.0 (март 1994) включала следующие новые возможности:
  - Поддержку стандартных для UNIX сетевых протоколов TCP/IP
  - BSD-совместимый интерфейс сокетов для сетевого программирования
  - Поддержку драйверов устройств для использования IP в сетях типа Ethernet
  - Расширенную файловую систему
  - Поддержку большого диапазона SCSI – контроллеров для высокопроизводительного доступа к дискам
- Версия 1.2 (март 1995) была последней версией ядра Linux только для PC.

# Linux 2.0

- Выпущена в июне 1996, со следующими новыми возможностями:
  - Поддержкой нескольких аппаратных архитектур, включая полный 64-разрядный перенос на рабочие станции Digital Alpha (первые 64-разрядные рабочие станции в мире).
  - Поддержкой многопроцессорной архитектуры
- Другие новые возможности:
  - Улучшенный код для управления памятью
  - Улучшенная производительность TCP/IP
  - Поддержку внутренних потоков (threads) ядра ОС, для обработки зависимостей между загрузочными модулями, и для автоматической загрузки модулей по требованию.
  - Стандартизованный конфигурационный интерфейс
- Доступна на процессорах Motorola 68000, Sun SPARC, PC (x86) и PowerMac.

# Система Linux

- Linux использует многие инструменты, разработанные как части Berkeley BSD UNIX, системы X Window разработки MIT, а также проекта GNU некоммерческой ассоциации Free Software Foundation (FSF).
- Минимальный набор системных библиотек был разработан как часть проекта GNU, с улучшениями, разработанными сообществом Linux.
- Средства сетевого администрирования Linux были разработаны на основе 4.3 BSD UNIX; недавние производные от BSD (например, Free BSD), в свою очередь, заимствовали код из Linux.
- Система Linux поддерживается слабо связанной сетью разработчиков, взаимодействующих через Internet. Небольшое число публично доступных ftp-серверов используются как хранилища информации о де-факто стандартах.

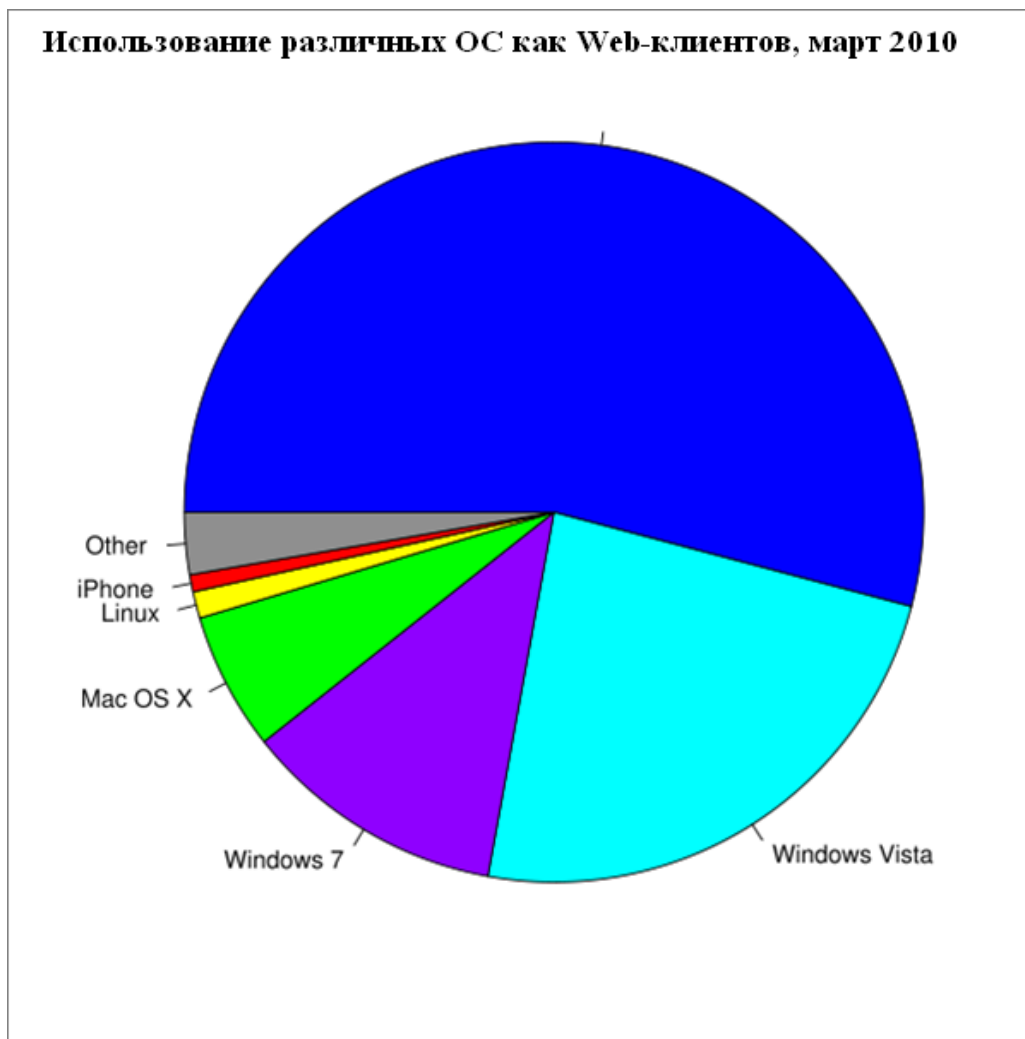
# Дистрибутивы Linux

- Стандартный предварительно откомпилированный набор пакетов, или *дистрибутивов*, включает базовую систему Linux, утилиты для инсталляции системы и управления системой, а также готовые к инсталляции пакеты инструментов для UNIX.
- Ранние дистрибутивы включали SLS и Slackware. *Red Hat* и *Debian* – популярные дистрибутивы, соответственно, основанный на коммерческих и некоммерческих исходных текстах.
- Единый формат файла пакета - RPM обеспечивает совместимость между различными дистрибутивами Linux
- Личный опыт: При частичных инсталляциях Linux в различных конфигурациях и последующих “доинсталляциях” до полной версии возможны проблемы: Инсталлятор путает фактический состав инсталлируемых пакетов (Linux Red Hat, 2003)

# Лицензирование Linux

- Ядро Linux распространяется на условиях GNU General Public License (GPL), которые установлены организацией Free Software Foundation.
- Программист, использующий Linux, либо создающий свои собственные системы на базе Linux, не имеет права превращать свой продукт в коммерческий (ведомственный);
- программное обеспечение, распространяемое на основе GPL, *не может распространяться только в виде двоичного кода* (т.е. в поставку Linux должен быть включен исходный код)

Linux в основном используется как *серверная* ОС.  
Использование различных ОС как web-клиентов

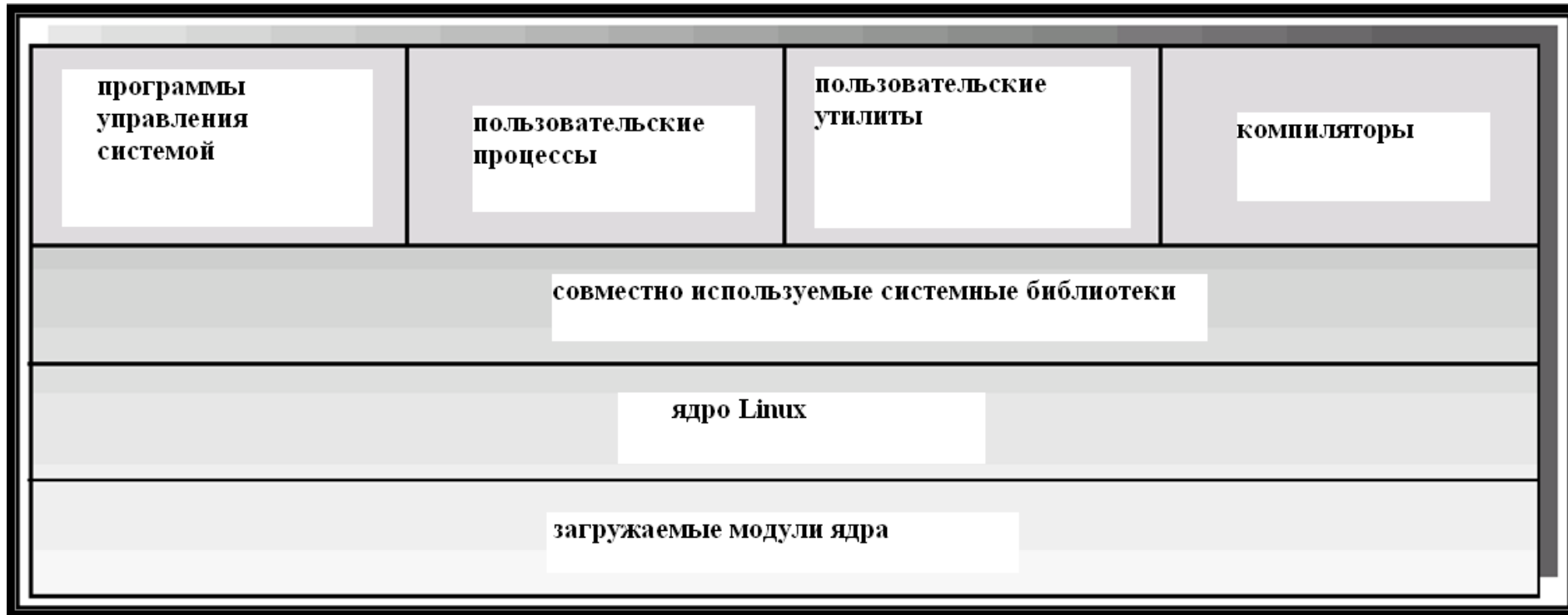




# Принципы проектирования

- Linux – многопользовательская и многозадачная ОС с полным набором UNIX-совместимых инструментов.
- Ее файловая система соответствует традиционной семантике UNIX. Она полностью реализует стандартную сетевую модель UNIX.
- Основные цели проектирования Linux – скорость, эффективность и стандартизация.
- Linux разработан как система, совместимая со стандартами POSIX по крайней мере два дистрибутива Linux были официально сертифицированы как совместимые с POSIX.
- Программный интерфейс Linux соответствует семантике *SVR4 UNIX*, но не *BSD UNIX*.

# Компоненты системы Linux



# Компоненты системы Linux (прод.)

- Как большинство реализаций UNIX, Linux состоит из трех основных групп кода – ядро, системные библиотеки и системные утилиты; наиболее важно различие между ядром и всеми остальными компонентами.
- Ядро отвечает за поддержку основных концепций (абстракций) ОС.
  - Код ядра исполняется в *привилегированном режиме*, и ему полностью доступны все аппаратные ресурсы компьютера.
  - Весь код и структуры данных ядра хранятся и исполняются в едином адресном пространстве.

# Компоненты системы Linux (прод.)

- Системные библиотеки определяют стандартный набор функций, с помощью которого приложения взаимодействуют с ядром, и которые реализуют основную часть функциональности ОС, не требующую исполнения в привилегированном режиме.
- Системные утилиты выполняют индивидуальные специфические задачи.

# Модули ядра

- Одним из важнейших новшеств в ядре Linux являются загружаемые модули ядра (loadable kernel modules, LKM), появившиеся в версии 1.2. Они обеспечивают ядру гибкость и функциональность
- Части (секции) кода ядра могут компилироваться, загружаться и выгружаться, независимо от остальной части ядра.
- Модуль ядра может реализовывать драйвер устройства, файловую систему или сетевой протокол.
- Модульный интерфейс позволяет третьим сторонам разрабатывать и распространять на своих собственных условиях драйверы или файловые системы, которые не могут распространяться на основе GPL.
- Модули ядра позволяют устанавливать Linux в виде стандартного, минимального ядра, без использования каких-либо встроенных устройств.
- Три компоненты модуля Linux поддерживают:
  - Управление модулем
  - Регистрацию драйвера
  - Разрешение конфликтов

# Управление модулем

- Управляет загрузкой модуля в память и его взаимодействием с остальной частью ядра.
- Управление модулем разбито на две части:
  - Управление частями кода модуля в памяти ядра
  - Управление символами, на которые модуль разрешает ссылаться
- Module requestor – управляет загрузкой запрошенных, но еще не загруженных модулей; он также регулярно опрашивает ядро, чтобы убедиться, что модуль до сих пор используется, и выгрузит его, если он долгое время активно не использовался.

# Схема исходного кода загружаемого модуля Linux

```
#include <linux/module.h>
#include <linux/init.h>

MODULE_LICENSE( "GPL" );
MODULE_AUTHOR( "Module Author" );
MODULE_DESCRIPTION( "Module Description" );

static int __init mod_entry_func( void )
{
    return 0;
}

static void __exit mod_exit_func( void )
{
    return;
}

module_init( mod_entry_func );
module_exit( mod_exit_func );
```

Макросы  
модуля

Конструктор/  
деструктор  
модуля

Макросы  
входа/  
выхода

# Регистрация драйверов

- Предоставляет модулю возможность сообщить ядру, что новый драйвер доступен.
- Ядро поддерживает динамическую таблицу всех известных драйверов и обеспечивает набор подпрограмм для добавления драйверов в эти таблицы или удаления из них в любое время.
- Таблицы регистрации включают следующие элементы:
  - Драйверы устройств
  - Файловые системы
  - Сетевые протоколы
  - Двоичные форматы



# Разрешение конфликтов

- Механизм, который позволяет различным драйверам устройств резервировать аппаратные ресурсы и защищать эти ресурсы от случайного использования другими драйверами
- Цели модуля разрешения конфликтов:
  - Предотвратить конфликты, связанные с использованием аппаратуры
  - Предотвратить *автопроверки (autoprobес)* от пересечения с уже существующими драйверами устройств
  - Разрешить конфликты различных драйверов, пытающихся иметь доступ к одной и той же аппаратуре

# Управление процессами в Linux

- Средства управления процессами в UNIX разделяют создание процесса и запуск новой программы на две различные операции.
  - Системный вызов `fork` создает новый процесс.
  - Новая программа запускается с помощью вызова `execve`.
- В UNIX процесс содержит всю информацию, которую ОС должна поддерживать для реализации концепции отдельного исполнения отдельной программы.
- В Linux свойства процесса делятся на три группы: идентификация процесса, его окружение и контекст.

# Идентификация процесса

- Идентификатор процесса (PID). Уникальный идентификатор процесса (число); используется для указания процессов в операционной системе, когда приложение выполняет системный вызов `signal`, `modify` или `wait` для другого процесса.
- Полномочия (Credentials). Каждый процесс должен иметь связанный с ним идентификатор пользователя и один или более идентификаторов групп, определяющих права процесса для доступа к системным ресурсам и файлам.
- Идентификация личности (Personality). Нетрадиционно для систем типа UNIX, но в Linux каждый процесс имеет уникальный идентификатор личности, с помощью которого возможна некоторая модификация семантики ряда системных вызовов.  
Используется главным образом в библиотеках эмуляции, для запроса о совместимости системных вызовов с тем или иным специфическим диалектом UNIX.

# Окружение процесса

- Окружение процесса получается из процесса-родителя, состоит из двух векторов, завершающихся нулями:
  - Вектор аргументов содержит список аргументов командной строки, использованный при вызове исполняемой программы; традиционно начинается с имени самой программы
  - Вектор окружения – это список пар “NAME=VALUE”, которые связывают переменные окружения с заданными именами и их произвольные текстовые значения.
- Передача переменных окружения между процессами и наследование этих переменных дочерними процессами – гибкие средства передачи информации компонентам системного программного обеспечения, работающим в непривилегированном режиме.
- Механизм переменных окружения обеспечивает средства настройки ОС, которые могут устанавливаться для каждого процесса отдельно, а не путем конфигурирования системы в целом.

# Контекст процесса

- (Постоянно изменяющееся) состояние исполняемой программы в любой момент времени.
- Контекст планирования – наиболее важная часть контекста процесса; эту информацию использует планировщик для приостановки и запуска процесса.
- Ядро поддерживает хранение статистической информации о ресурсах, потребляемых в каждый момент каждым процессом, и общем объеме ресурсов, использованных каждым процессом с момента его создания по настоящий момент.
- Таблица файлов – это вектор указателей на системные файловые структуры. При выполнении системных вызовов для ввода-вывода процессы ссылаются на эти структуры с помощью индексов в таблице файлов.

# Контекст процесса

- В то время как таблица файлов содержит список открытых файлов, контекст файловой системы применяется для запросов об открытии новых файлов. Здесь хранятся ссылки на текущую корневую (root) директорию и рабочую (default) директорию для поиска файлов.
- Таблица обработчиков сигналов определяет подпрограммы в адресном пространстве процесса, которые должны быть вызваны при возникновении соответствующих сигналов.
- Контекст виртуальной памяти процесса определяет все содержимое его персонального адресного пространства.

# Процессы и потоки

- Linux использует одно и то же внутреннее представление для процессов и потоков; поток – это новый процесс, который использует общее адресное пространство с процессом-родителем.
- Различие проявляется только в случае, когда новый поток создается системным вызовом `clone`.
  - `fork` создает новый процесс со своим полностью новым контекстом
  - `clone` создает новый процесс со своим новым идентификатором личности, но такой, которому разрешено совместно использовать структуры данных со своим родителем
- Использование `clone` дает процессам возможность явного контроля над тем, какие ресурсы совместно используются потоками.

# Планирование

- Распределение операционной системой процессорного времени между различными задачами.
- В то время как обычно под планированием понимается запуск и приостановка процессов, в Linux планирование также включает выполнение различных задач ядра.
- Выполнение задач ядра включает как задания, запрошенные данным процессом, так и задания, исполняемые в процессе работы драйверов.



# Синхронизация в ядре

- Запрос на исполнение в режиме ядра может возникнуть в двух случаях:
  - Исполняемая программа может запросить сервис ОС, как явно с помощью системного вызова, так и неявно, например, при отказе страницы.
  - Драйвер устройства может сгенерировать аппаратное прерывание, в результате которого процессор начнет исполнять в режиме ядра обработчик данного прерывания.
- Синхронизация в ядре требует, чтобы критические секции ядра исполнялись без их прерывания другими критическими секциями.

# Синхронизация в ядре

- Linux использует два метода для защиты критических секций:
  1. Обычный код ядра - не прерываемый
    - Если получено прерывание по времени, в момент, когда процесс исполняет подпрограмму системного сервиса ядра, флаг *need\_resched* служит для указания того, чтобы запустился планировщик, когда завершится системный вызов и управление должно быть передано непривилегированному коду.
  2. Второй метод применяется к критическим секциям ядра, которые исполняются в сервисах обработки прерываний.
    - Используя аппаратуру процессора для управления прерываниями для отключения прерываний во время исполнения критической секции, ядро гарантирует, что оно может исполняться без риска одновременного обращения к общим структурам данных.

# Синхронизация в ядре

- Во избежание потери производительности, ядро Linux использует архитектуру синхронизации, которая позволяет большим критическим секциям исполняться без необходимости отключения прерываний на все время исполнения критической секции.
- Службы обработки прерываний делятся на верхнюю половину (*top half*) и нижнюю половину (*bottom half*).
  - Верхняя половина – это обычная процедура обработки прерываний, исполняемая с отключением рекурсивных прерываний.
  - Нижняя половина исполняется при включенном режиме прерываний, с использованием мини-планировщика, который обеспечивает, чтобы нижние половины не прерывали друг друга.
  - Эта архитектура дополняется механизмом для выбора нижних половинок при исполнении обычного кода ядра.

# Уровни защиты прерываний

обработчики прерываний верхней половины

обработчики прерываний нижней половины

сервисные системные подпрограммы ядра (не прерываемые)

программы пользовательского уровня (прерываемые)



увеличение приоритетов



- Код каждого уровня может быть прерван кодом более высокого уровня, но никогда не будет прерван кодом того же или более низкого уровня.
- Пользовательский процесс может быть всегда прерван другим процессом, если происходит прерывание для планирования в режиме разделения времени.

# Планирование процессов

- Linux использует два алгоритма планирования процессов:
  - Алгоритм деления времени для равноправного планирования с прерываниями между различными процессами
  - Алгоритм реального времени для случая, когда абсолютные приоритеты более важны, чем равноправность
- Класс планирования процесса определяет, какой именно алгоритм применить.
- Для процессов с делением времени Linux использует алгоритм на основе доверия с приоритетами.
  - Правило учитывает как историю процесса, так и его приоритет.
  - Эта система автоматически определяет приоритеты интерактивных процессов или исполняющих ввод-вывод.

$$\text{credits} := \frac{\text{credits}}{2} + \text{priority}$$

# Планирование процессов

- Linux реализует классы планирования: FIFO и round-robin; в обоих случаях каждый процесс имеет приоритет, а не только определенный класс планирования.
  - Планировщик запускает процесс с наивысшим приоритетом; для процессов с одним и тем же приоритетом, он исполняет процесс, который дольше всего ждал
  - FIFO – процессы исполняются до их завершения или блокировки
  - round-robin – процесс будет прерван через некоторое время и помещен в конец очереди планирования, так что RR-процессы одинакового приоритета автоматически разделяют время между собой.

# Поддержка симметричного мультипроцессирования (SMP)

- Linux 2.0 была первым ядром Linux, поддерживающим SMP-оборудование; различные процессы или потоки могут выполняться параллельно на нескольких процессорах
- Для соблюдения требований ядра об исполнении без прерываний, SMP накладывает следующее ограничение: не более чем один процесс в каждый момент может исполнять код в режиме ядра.

# Операционные системы для мобильных устройств



# ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

- *Мобильные устройства*: мобильные телефоны, смартфоны, коммуникаторы
- Особенности ОС для мобильных устройств:
  - учет более жестких ограничений по памяти
  - учет более низкой скорости процессора
  - учет особенностей экранов и экранных навигаторов конкретных моделей мобильных устройств
  - совместимость с основными форматами файлов: .doc/docx, .ppt/.pptx, .pdf, .jpg и др.
  - мультимедийные возможности: рисунки, видео, аудио, обмен мультимедийными сообщениями
  - поддержка коммуникационных и сетевых технологий: Wi-Fi / WiMAX, Bluetooth, GPRS, EVDO, GSM, CDMA

# Рынок ОС для мобильных устройств

- Nokia Symbian OS
- Google Android
- Windows Mobile
- Blackberry OS
- Apple iOS
- Samsung Bada
- PalmOS

# Windows Mobile

- ОС для мобильных устройств фирмы Microsoft
- Относится к семейству Windows CE (Consumer and Embedded) – Windows для встроенных систем; ядро Windows Mobile – система Windows CE
- Текущая версия (2011) – Windows Phone
- В новых версиях все более полно воспроизводятся возможности Windows и улучшается пользовательский интерфейс
- В США – третья по популярности ОС для мобильных устройств (после Blackberry OS и iPhone OS)
- Поддерживаемые мобильные устройства: PocketPC, смартфоны, коммуникаторы (например, Qtek)
- Первая версия была выпущена в 1996 г.

# Windows Mobile: Возможности и ПО

- *Office Mobile* – аналог Microsoft Office для мобильных устройств; полная совместимость по форматам
- *Windows Media Player* – мультимедийный проигрыватель
- *Internet Explorer Mobile*
- Поддержка Bluetooth (2.1), Wi-Fi
- Пользовательский интерфейс – с помощью стайлуса и пальцев
- Поддержка .NET Compact Framework
- *Microsoft ActiveSync* – для синхронизации данных с ПК

# Версии Windows Mobile 6

- Windows Mobile 6 Classic — для КПК
- Windows Mobile 6 Professional — для коммуникаторов
- Windows Mobile 6 Standard — для смартфонов

# Новые возможности Windows Mobile 6.1

- Увеличена скорость работы интерфейса
- SMS-сообщения отображаются в виде чата
- В Internet Explorer добавлены функции «Масштаб» и «Обзор страницы»
- Добавлен полноценный менеджер задач
- Новый элемент меню «Managed Programs»
- Улучшена работа по Wi-Fi и Bluetooth
- Добавлена поддержка Bluetooth 2.1
- Переработан алгоритм синхронизации с ПК
- Добавлена поддержка MS Exchange Server 2007
- Улучшен органайзер

# Интерфейс Windows Mobile 6.5



# Windows Mobile: Перспективы

- Конкуренция с Apple (iPhone OS) и Google (Android)
- Новый проект – Microsoft KIN: новый смартфон, конкурирующий с Apple iPhone



# Symbian OS

- Популярная ОС для мобильных устройств, разработанная консорциумом Symbian (Nokia, Ericsson, Psion, Motorola), основанным в 1998 г.
- Фирма Nokia финансирует объединение Symbian Foundation (цель – мобильная платформа на основе Symbian OS)
- Разработана на основе ОС Psion EPOC32
- Язык реализации – C++; имеется поддержка Java
- Наиболее распространенные версии – Symbian OS Series 60 2<sup>nd</sup> edition; 3<sup>rd</sup> edition
- На конец 2009 г., 47% смартфонов используют Symbian OS (Blackberry OS – 20%, Windows Mobile – 9%, Google Android – 5%)

# Symbian OS: Возможности

- Меню с иконками приложений
- Список контактов
- Поддержка встроенной фото- и видеокамеры, галереи изображений и видеоклипов
- Обработка файлов, управление памятью (SmartMedia)
- Web-браузер
- Email
- Поддержка GPS-навигации
- Редакторы фото и видео
- Поддержка Java - загрузки и исполнения мидлетов
- Библиотека приложений фирмы Nokia

# Google Android

- *Android* – стек приложений для мобильных устройств, включающий операционную систему (на базе ядра Linux), промежуточное программное обеспечение (middleware) и сервисные программы
- Разработана фирмой Android, Inc., приобретенной компанией Google (2007)
- Четвертая по популярности ОС для смартфонов в США (2010)
- Сервисные программы и библиотеки написаны на Java

# Google Android: Возможности

- Двумерная и трехмерная графика (OpenGL)
- СУБД SQLite для хранения данных
- Поддерживаемые сетевые технологии: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, WiMAX, Bluetooth 2.0
- Обмен сообщениями: SMS, MMS
- Web-браузер на базе WebKit Application Framework
- *Поддержка Java*: собственная реализация – Dalvik Virtual Machine, разработанная специально для мобильных устройств. Стандарт J2ME не поддерживается (устарел)
- *Поддержка мультимедиа*: кодеки, обработка файлов, взаимодействие с видео- и аудиоустройствами
- *Поддержка разработки приложений*: Android SDK; эмулятор устройств, отладка, профилирование, plug-in к среде Eclipse для разработки Java-приложений

# Интерфейс Google Android



# BlackBerry OS

- ОС для мобильных устройств с базовым набором приложений
- Работает на смартфонах и коммуникаторах, разработанных фирмой Research and Motion (RIM) – например, BlackBerry Torch 9800
- Современная версия (2010) – BlackBerry OS 6.0

# BlackBerry OS: Возможности

- Пометка сообщений и установка времени напоминаний на смартфоне BlackBerry
- Просмотр вложенных папок персональных контактов и редактирование контактов. BES (BlackBerry Enterprise Server) вставит все пользовательские контакты в приложение Contacts, даже если они находятся в различных папках
- Просмотр и использование контактов, расположенных в общих папках, и копирование их в локальный список контактов пользователя, при наличии разрешения
- Программа просмотра файлов для доступа в общие сетевые ресурсы с возможностью открывать, добавлять и сохранять документы. Возможность просмотра информации о документе, в том числе типа файла, размера и даты
- Отправка приглашения на встречи и записи календаря со смартфона BlackBerry
- Возможность добавлять, удалять, перемещать и переименовывать персональные папки
- Возможность просматривать личный список рассылки в контактах Outlook и опрашивать письма по нему
- RIM также работает над решением, которое позволит письмам, пришедшим со смартфона, выглядеть так же, как если бы они были отправлены из Microsoft Outlook

# Новые возможности BlackBerry OS 6.0

- Новый пользовательский интерфейс предназначенный для широкого использования Multitouch-жестов, но при этом сохраняющий возможности управления с помощью трекбола
- Структура рабочего стола - среднее между рабочими столами операционных систем Apple iOS и Android.
- Улучшенные мультимедийные возможности ОС.
- Улучшенные возможности для веб-серфинга. В новой ОС используется мобильный браузер на основе WebKit, что позволяет запускать веб-приложения, написанные на языке HTML 5.
- Упрощенный доступ к мобильным сообщениям, электронной почте и социальным сетям.



# Перспективы ОС для мобильных устройств

- Улучшение и упрощение пользовательского интерфейса
- Улучшенная графика
- Более широкие мультимедийные возможности
- Развитие набора сервисных и игровых программ
- Совместимость с ПК и с форматами файлов
- Продолжение и развитие использования платформы Java для мобильных устройств
- Развитие самих мобильных устройств: улучшение разрешения экранов, ускорение процессоров, увеличение объема памяти, реализация новых быстрых коммуникационных технологий

# Вопросы?