

# Операционные системы, среды и оболочки

Архитектура операционной  
системы

# Ядро и вспомогательные модули операционной системы

- ▶ При функциональной декомпозиции ОС модули разделяются на две группы:
  - ядро – модули, выполняющие основные функции ОС;
  - модули, выполняющие вспомогательные функции ОС.

# Модули ядра ОС

- ▶ Модули ядра ОС выполняют следующие базовые функции ОС:
  - управление процессами
  - управление памятью
  - управление устройствами ввода-вывода
- ▶ Ядро обеспечивает решение задачи **организации вычислительного процесса**: переключение контекстов, загрузка/выгрузка страниц, обработка прерываний и т.п.
- ▶ Другая задача – поддержка приложений, создание для них **прикладной программной среды**. Приложения обращаются к ядру с запросами (**системными вызовами**) для выполнения базовых операций (открытие и чтение файла, вывод информации на дисплей и т.п.)
- ▶ Функции выполняемые ядром ОС требуют высокой скорости выполнения и для этого размещаются постоянно в оперативной памяти (**резидентные модули**).

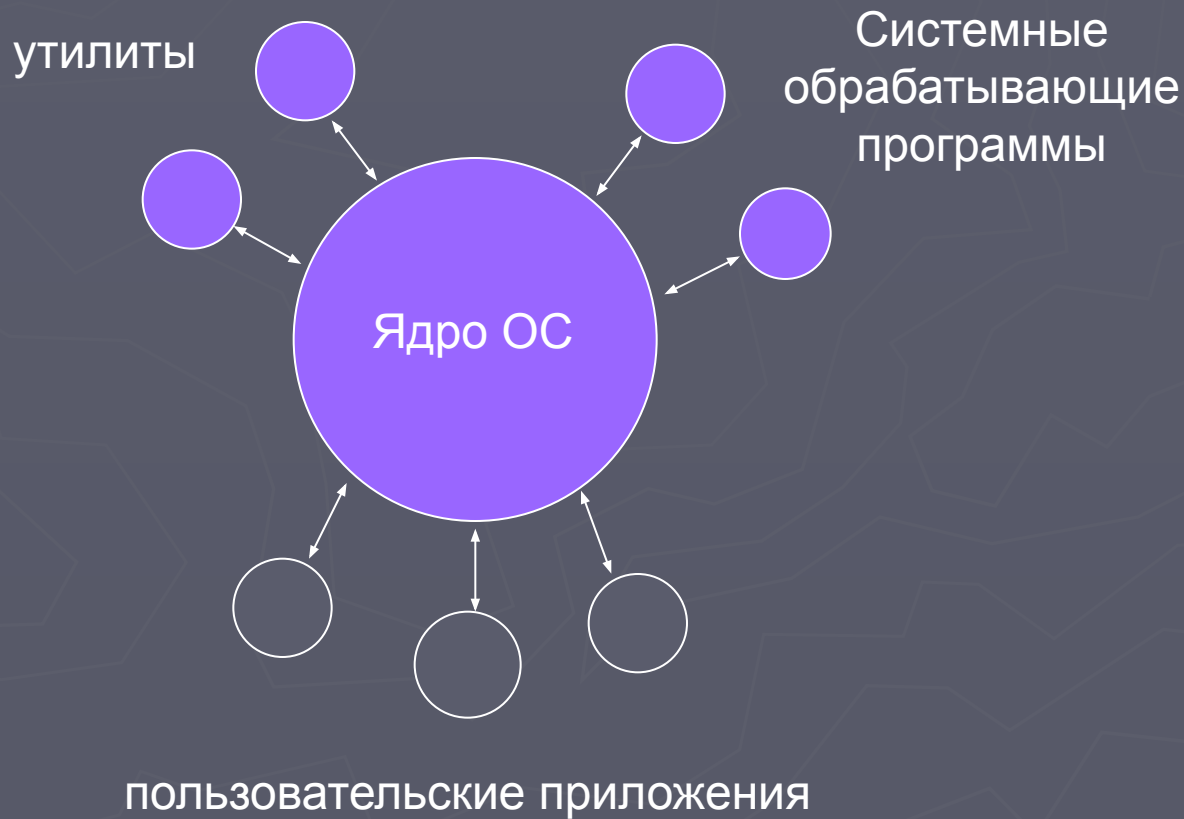
# Вспомогательные модули операционной системы

- ▶ Вспомогательные модули выполняют полезные, но менее обязательные функции. Например:
  - архивирование информации;
  - дефрагментация данных на диске;
  - поиск необходимого файла и т.п.
- ▶ Вспомогательные модули часто оформляются как обычные приложения и провести границу между ними и обычными приложениями сложно.
- ▶ Деление на основные и вспомогательные модули ОС условно. Некоторые программы переходят из разряда вспомогательных модулей в основные и наоборот.

# Вспомогательные модули операционной системы

- ▶ Вспомогательные модули ОС условно разделяются на следующие группы:
  - *Утилиты* – приложения, решающие отдельные задачи управления и сопровождения ОС
  - *Системные обрабатывающие программы* – текстовые и графические редакторы, компиляторы, компоновщики и т.п.
  - *Программы предоставления пользователю дополнительных услуг* – специальный вариант пользовательского интерфейса, калькулятор, игры и т.п.
  - *Библиотеки процедур* – модули различного назначения, упрощающие разработку приложений.
- ▶ Вспомогательные модули обращаются к функциям ядра ОС посредством системных вызовов.

# Ядро и вспомогательные модули операционной системы



# Привилегированный режим процессора

- ▶ Для надежного управления работой приложений ядро ОС должно обладать некоторыми привилегиями по отношению к остальным приложениям.
- ▶ Обеспечивается привилегированный режим специальными средствами аппаратной поддержкой. Процессор компьютера поддерживает как минимум два режима работы – **пользовательский** (user mode) и **привилегированный** (kernel mode).
- ▶ Приложения в пользовательском режиме не могут выполнять некоторые критичные команды (переключение процессора с задачи на задачу, доступ к механизму выделения и защиты областей памяти и т. п.).

# Привилегированный режим работы

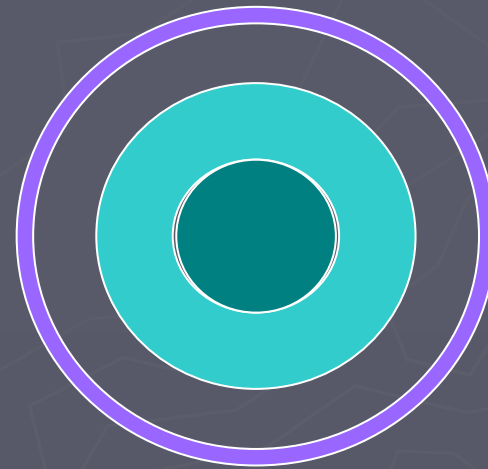
- ▶ Между числом привилегий, поддерживаемых аппаратурой и операционной системой нет однозначного соответствия:
  - процессор Intel поддерживает 4 режима работы процессора – операционные системы Windows используют два из них.
- ▶ Для реализации привилегированного режима достаточно поддержки двух режимов работы
- ▶ Повышение устойчивости ОС, обеспечивающееся использованием работы в привилегированном режиме, достигается за счет некоторого замедления, вызванного необходимостью переключения работы ядра.
- ▶ Архитектура ОС, основанная на разделении привилегированного режима для ядра и пользовательского режима для приложений – стала классической.



# Многослойная структура ОС

► Вычислительная система под управлением ОС можно рассматривать как состоящую из нескольких слоев:

- Нижний слой – аппарататура;
- Средний – ядро ОС;
- Верхний – утилиты, приложения и т.п.



Аппара  
тура  
Ядро  
ОС  
Прилож  
ения

# Детализация структуры ядра

- ▶ Ядро, являясь структурным элементом ОС, может быть логически разложено на ряд слоев:
  - Средства аппаратной поддержки ОС
  - Машинно-зависимые компоненты ОС (включает модули, отражающие специфику аппаратной платформы компьютера)
  - Базовые механизмы ядра (включает наиболее примитивные операции ядра – переключение контекстов процессов, диспетчеризация прерываний), модули выполняют решения принятые на более высоких уровнях
  - Менеджеры ресурсов (реализует задачи стратегического управления), включает менеджеры – диспетчеры процессов, ввода-вывода и т.п.
  - Интерфейсы системных вызовов (включает модули взаимодействия с приложениями и системными утилитами, функции API).

# Аппаратная зависимость ОС

- ▶ Операционная система в процессе работы взаимодействует с аппаратными средствами компьютера:
  - Средства поддержки привилегированного режима
  - Средства трансляции адресов
  - Средства переключения процессов
  - Защита областей памяти
  - Система прерываний
  - Системный таймер
- ▶ Это делает ОС привязанной к определенной аппаратной платформе

# Переносимость операционной системы

- ▶ Под переносимостью операционной системы понимается способность использования ОС на различных аппаратных платформах с минимальными изменениями в ее структуре. Для уменьшения числа машинно-зависимых модулей разработчики ОС ограничивают универсальность машинно-независимых модулей. Например, Windows разработана для нескольких типов процессоров и для многопроцессорных систем используются собственные модули.
- ▶ Для обеспечения переносимости следуют следующим правилам:
  - Большая часть кода написана на языке, трансляторы которого существуют для всех планируемых платформ;
  - Объем машино-зависимых частей кода должен быть минимизирован;
  - Аппаратно-зависимый код должен быть изолирован в нескольких модулях
- ▶ В идеале машино-зависимые модули ядра полностью экранируют остальную часть ОС от конкретных деталей аппаратной платформы (кэши, контроллеры прерываний и т.п.).

# Микроядерная архитектура

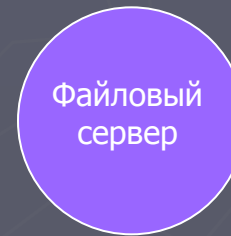
- ▶ Концепция микроядерной архитектуры заключается в выделении в качестве работающего в привилегированном режиме части ОС, ответственном за небольшой набор системных функций (управление процессами, обработка прерываний, управление виртуальной памятью, пересылка сообщений). Данная часть ОС называется **микроядром**.
- ▶ Все остальные высокоуровневые функции ядра разрабатываются в виде приложений, работающих в пользовательском режиме – **серверы ОС**.
- ▶ Взаимодействие между обычными приложениями и серверами ОС осуществляется через механизм обращений. **Клиентское приложение** отправляет запрос к серверу ОС через микроядро ОС. Такой механизм обеспечивает защиту работы приложений.

# Микроядерная архитектура

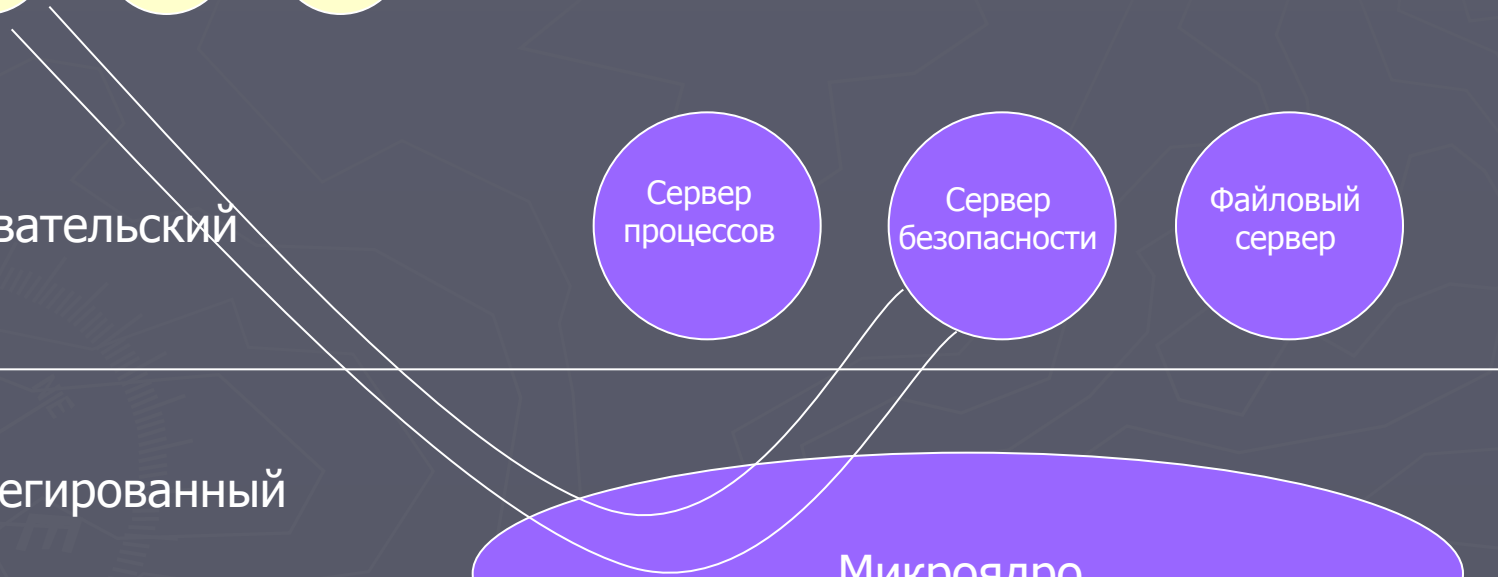
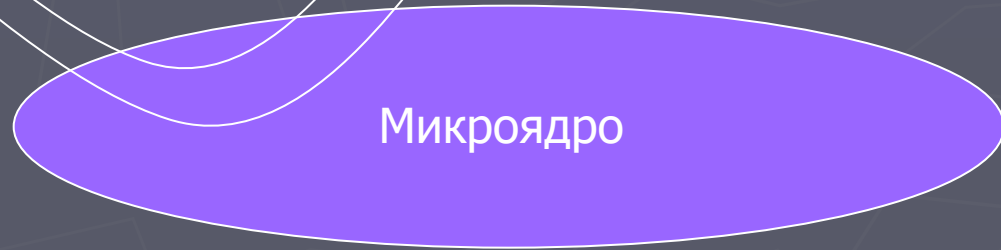
Приложения пользователей



Пользовательский режим



Привилегированный режим



# Достоинства микроядерной архитектуры

- ▶ Операционные системы, основанные на микроядерной архитектуре обладают рядом преимуществ, предъявляемых к современным ОС:
  - Переносимость (обусловлена малым числом модулей в аппаратно-зависимом микроядре)
  - Расширяемость (добавление новых функций связано с включением новых серверов ОС)
  - Надежность (обусловлена изолированностью процессов)
  - Поддержка распределенных вычислений (используется механизм взаимодействия приложений аналогичный взаимодействию в распределенных системах)
- ▶ Недостаток
  - Производительность (обладают меньшей производительностью)

# Совместимость операционных систем

- ▶ Совместимость – возможность операционной системы выполнять приложения, написанные для других ОС.
- ▶ Выделяют
  - **Двоичная совместимость** – на уровне кодов (программные модули могут быть просто перенесены и запущены)
  - **Совместимость исходных текстов** – приложения могут быть перекомпилированы в новый исполняемый модуль для ОС.
- ▶ Совместимость на уровне кодов может быть достигнута с помощью **эмуляции** двоичного кода.



# Прикладные программные среды

- ▶ Прикладная программная среда – совокупность средств ОС, предназначенная для организации выполнения приложений, использующих определенную систему машинных команд, определенный тип API.
- ▶ Каждая ОС создает хотя бы одну программную среду.
- ▶ Для обеспечения совместимости различных программных сред используются решения:
  - эмуляция двоичного кода,
  - трансляция API.