Операционные системы
Архитектура ОС

лектор: Дроздов Сергей Николаевич
доцент кафедры МОП ЭВМ ЮФУ
_dr@mail.ru
sndrozdov@sfedu.ru

Таганрог 2020 г.

Ядро ОС

- Определения ядра (из разных источников):
  - центральная часть ОС, обеспечивающая приложениям координированный доступ к ресурсам компьютера:
  - часть операционной системы, постоянно находящаяся в оперативной памяти;
  - модули, выполняющие основные функции ОС, такие, как управление процессами, памятью, устройствами ввода-вывода и т. п.
  - часть ОС, работающая в привилегированном режиме.
- Основные проблемы разработки ОС – это проблемы организации ядра и его взаимодействия с другими частями системы.

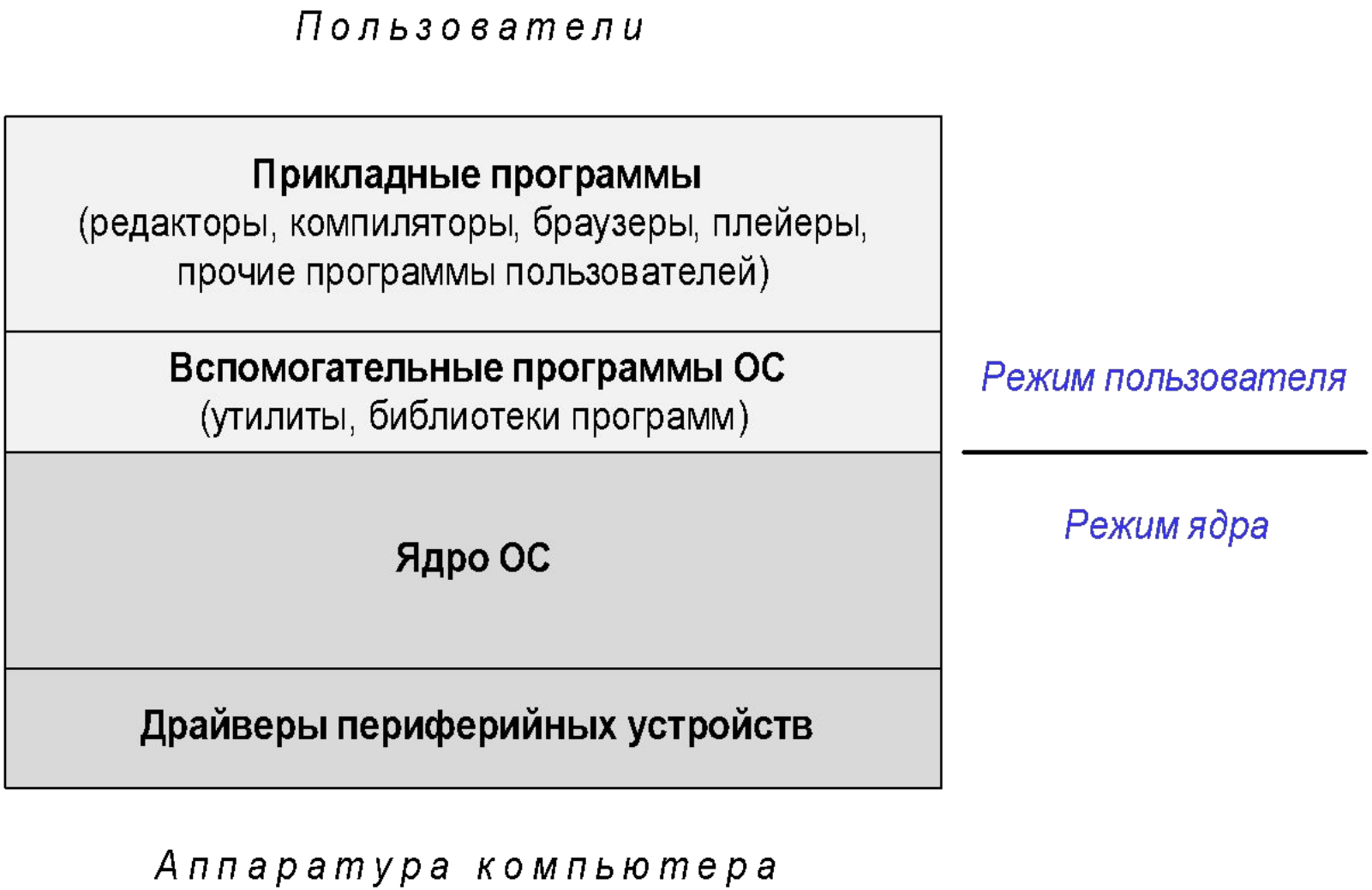
Режимы работы процессора

- Привилегированный режим (режим ядра, kernel mode):
  - обеспечивает неограниченный доступ ко всем ресурсам компьютера.
- Непривилегированный режим (пользовательский режим, user mode):
  - недоступны некоторые команды; может быть ограничен доступ к областям памяти, портам устройств и системным регистрам.
- Использование разделения режимов – основной способ защиты от вмешательства пользовательских процессов в работу системы и других процессов.
- Необходимость частого переключения режимов существенно снижает быстродействие.

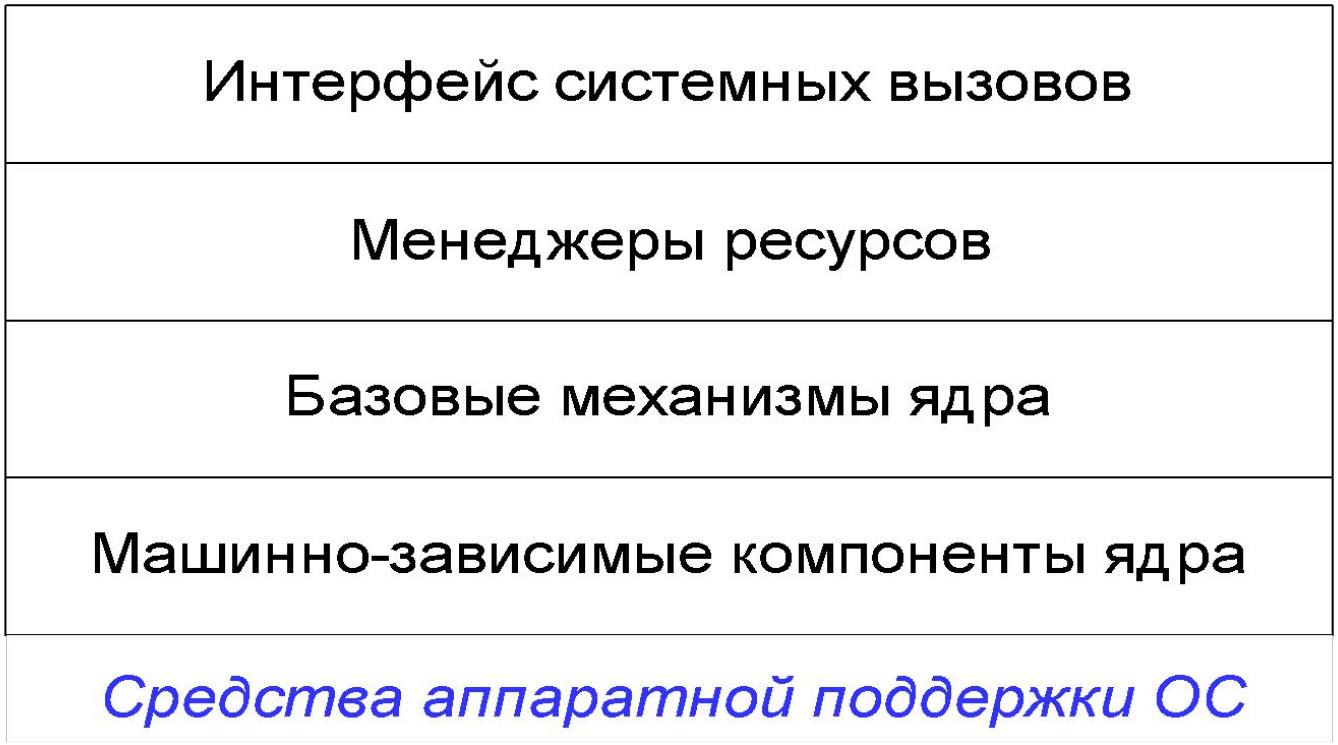
Место драйверов устройств в структуре ОС

- Драйверы обычно должны быть частью ядра, поскольку они непосредственно работают с аппаратурой.
- Драйверы – это отдельные файлы, которые должны включаться в ядро в процессе загрузки.
- В современных ОС имеется возможность подгружать и выгружать драйверы в ходе работы.
- Поскольку разнообразие драйверов очень велико и разрабатывают их очень разные люди, драйверы являются основным источником ошибок при работе ОС (до 80 – 90% ошибок). Драйвер, будучи частью ядра, в состоянии вызвать крах всей системы.

Общая структура ОС



Многослойная структура ядра



- Разделение ядра ОС на относительно независимые слои позволяет упростить разработку и отладку ядра, повысить надежность работы.

Средства аппаратной поддержки ОС

- Разделение режимов работы процессора (колец защиты). Включает в себя проверку допустимости команд.
- Средства трансляции адресов. Аппаратура обеспечивает замену виртуальных адресов физическими, в соответствии с программно управляемыми таблицами трансляции. Включает в себя проверку допустимости адресов.
- Средства переключения контекста. Позволяют быстро сохранить контекст текущего процесса (нити) и загрузить другой контекст.
- Система прерываний. Позволяет выбрать программу обработки прерывания, сохранить состояние прерванной программы, переключить режим процессора, маскировать прерывания, определять приоритеты прерываний.
- Системный таймер. Обычно – регистр-счетчик, значение которого аппаратно декрементируется на каждом такте. По достижении нуля посылает сигнал прерывания.

Машинно-зависимые компоненты ОС

- Конечно, вся ОС состоит из команд определенного процессора. Но для модулей, написанных на языке высокого уровня (C, C++ и т.п.) в большинстве случаев достаточно перетранслировать их.
- Существенная зависимость от архитектуры проявляется:
  - в составе регистров, включаемых в контекст процесса;
  - в способе обработки прерываний;
  - в способе работы с ПУ;
  - в механизме трансляции адресов;
  - и др.
- При переносе на другую аппаратную платформу машинно-зависимые модули приходится переписывать.
- В Windows NT был включен слой HAL (Hardware Abstractions Level), позволявший системе работать на 4 разных архитектурах (постепенно три из них отсохли, но HAL остался).

Базовые механизмы ядра

- Включают в себя:
  - переключение процессов;
  - обмен данными (свопинг) между основной памятью и дисками;
  - обслуживание прерываний;
  - пересылку сообщений;
  - и др.
- Обычно для различных задач, решаемых ОС, различают политику, реализуемую в ОС (что нужно сделать) и механизм (как это сделать). Базовые механизмы ядра – это именно механизмы, политика же определяется на уровне менеджеров ресурсов.
  - Например, переключение контекста процессов – это механизм. Выбор следующего текущего процесса – политика.
  - Чтение сектора диска – механизм. Как долго надо хранить в памяти прочитанный сектор – политика.
- Такой подход позволяет гибко изменять политику, не затрагивая механизмы.

Менеджеры ресурсов

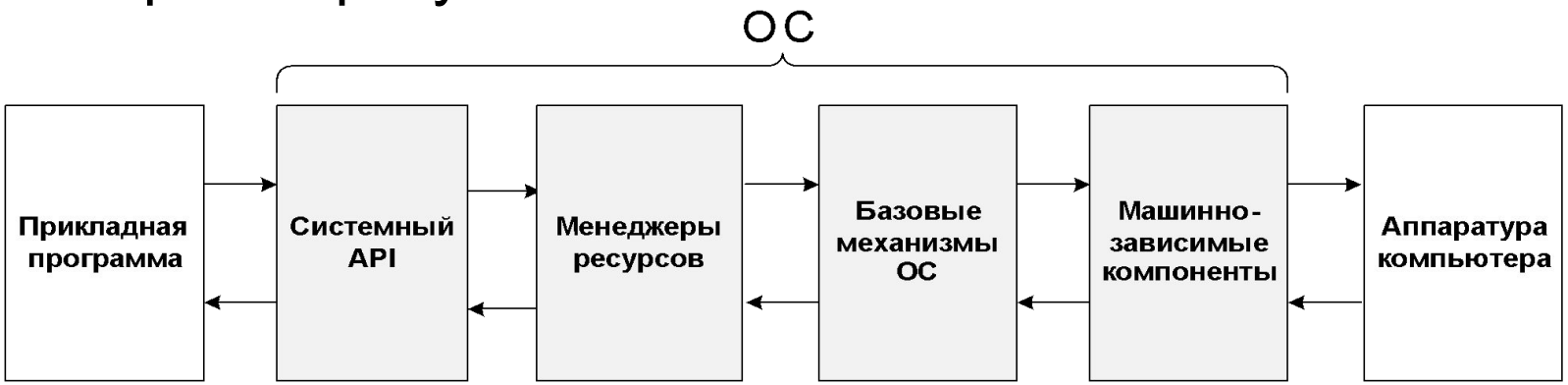
- Менеджеры ресурсов определяют политику (основные алгоритмы работы системы). Например:
  - диспетчер процессов;
  - диспетчер памяти;
  - диспетчер ввода/вывода;
  - менеджер файловой системы (ФС);
    - в случае нескольких ФС в рамках одной ОС – отдельные менеджеры каждой ФС плюс менеджер виртуальной ФС, выполняющий общие для всех операции;
  - менеджер сети;
  - и др.

Интерфейс системных вызовов

- Определяет возможности для прикладных программ использовать услуги ОС.
- Формализуется в виде API (Applied Programs Interface) – описания системных функций, их имен, количества и типа параметров.
- Реализуется чаще всего в виде программных прерываний для обращения к нижележащим слоям ядра.
- Некоторые простые API-функции могут выполняться без обращения к ядру.
- Над API часто надстроен слой библиотечных функций или классов конкретного языка (но это уже за пределами ОС).

Взаимодействие между слоями ядра

- Обычная последовательность действий – нисходящая цепочка вызовов нижележащих слоев с возвратом результатов.



- В принципе, ради повышения эффективности иерархия вызовов нередко нарушается.
- Часто необходимо взаимодействие между модулями одного слоя.
  - Например, при создании процесса диспетчер процессов обращается к менеджеру файловой системы для чтения программы и к менеджеру памяти для выделения оной.

Монолитное ядро

- Все основные подсистемы и все слои ядра работают в режиме ядра и проектируются как единое целое.
- Взаимодействие подсистем внутри ядра обеспечивается командами перехода и вызова подпрограмм.
- Плюс: высокое быстродействие.
- Минусы:
  - трудность отладки, сопровождения и развития системы;
  - возможность краха всей системы при ошибке в какой-либо ее части.

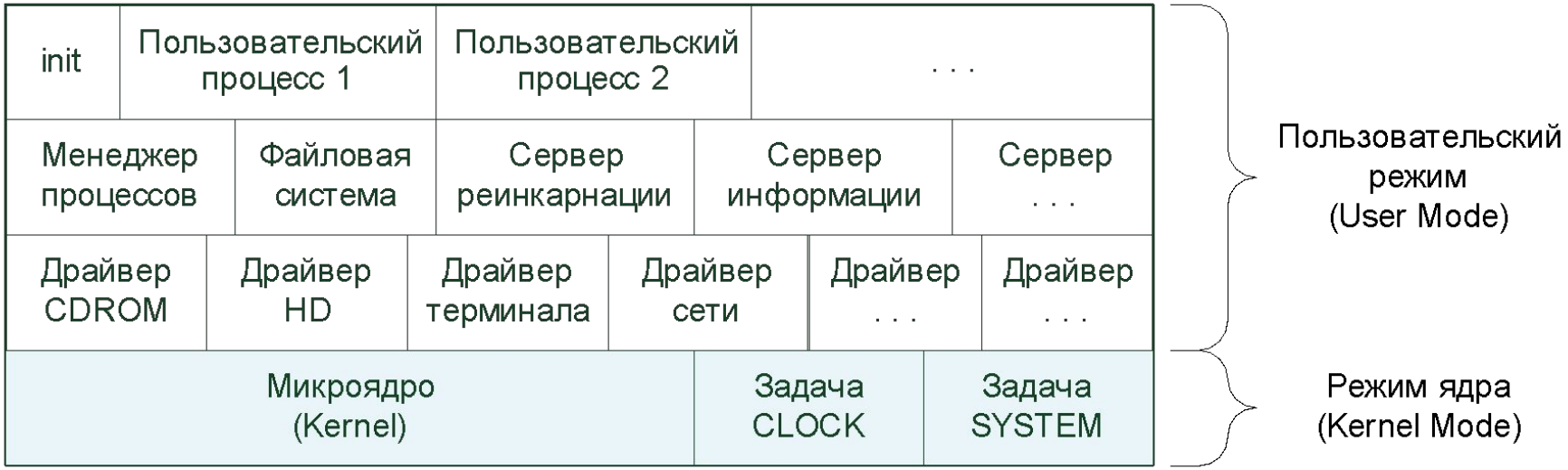
Модульно-монолитное ядро

- Основные подсистемы проектируются как отдельные модули с четко определенными возможностями взаимодействия. При загрузке модули объединяются в ядро.
- Сохраняется высокое быстродействие.
- Существенно облегчается отладка и сопровождение.
- Сохраняется возможность краха системы при сбое в отдельном модуле.

Микроядро

- В режиме ядра работает минимально необходимая часть кода системы.
- Большая часть модулей, традиционно включавшихся в ядро, работают в пользовательском режиме.
  - Сюда относятся, например, файловая система, планирование процессов, графика.
- Взаимодействие модулей ОС с ядром и друг с другом – обычно путем посылки сообщений при посредстве ядра.
- С драйверами – сложный вопрос.
- Плюсы:
  - повышается надежность;
  - работа ядра в пределах человеческого понимания.
- Минус: снижение быстродействия из-за частых вызовов микроядра другими частями системы.

Пример: структура микроядерной ОС MINIX 3



- Только самый нижний уровень работает в режиме ядра.
- Остальные части системы пользуются системными вызовами микроядра.
- Реализация системных вызовов основана на механизме обмена сообщениями.

Гибридное ядро

- Такой тип ядра является компромиссом между эффективностью монолитного ядра и надежностью микроядра.
- Нередко ОС, задуманные как микроядерные, постепенно эволюционируют в сторону монолитных.
- Один из вариантов: те части ОС (серверы и драйверы), которые в микроядерной архитектуре работали бы в пользовательском режиме и в отдельных адресных пространствах, в гибридной ОС работают в пространстве ядра, но при этом:
  - могут динамически подгружаться и выгружаться из памяти;
  - используют механизм сообщений для взаимодействия с микроядром.
- Самый известный пример – Windows NT и ее потомки.

Наноядро и пикоядро

- Наноядро – очень маленькое ядро, обеспечивающее только обработку аппаратных прерываний и передающее результат обработки в вышележащие слои ОС с помощью программных прерываний.
  - Например, на основании обработки прерываний от клавиатуры передать в ОС введенный символ.
- Основное назначение – снизить зависимость от аппаратуры, облегчить перенос на другую платформу.
- Иногда наноядро выполняет и другие функции – например, переключение нитей. Тогда это просто означает «очень маленькое микроядро».
- Что такое пикоядро, никто толком не знает. Очевидно, очень маленькое наноядро.

Экзоядро: основная идея

- Традиционные ОС скрывают от пользователя низкоуровневые особенности аппаратуры, предлагая взамен высокоуровневые абстракции: файлы, виртуальную память, IPC, нити и т.п.
- Недостаток такого подхода: система очень сложна, отсюда – снижение производительности и ненадежность.
- Можно выделить две стороны работы ОС:
  - предоставление процессам высокоуровневых абстракций;
  - контроль использования низкоуровневых ресурсов при реализации этих абстракций.
- Идея экзоядерных систем: оставить за ядром только вторую часть: выделение / освобождение аппаратных ресурсов и контроль прав доступа к ним, переложив на уровень пользовательских приложений использование этих ресурсов.

Экзоядро: пользовательский уровень

- Над маленьким экзоядром могут надстраиваться виртуальные машины, на которых, в принципе, могут работать разные ОС (Linux, Windows и др.).
- Чтобы это осуществить, каждая виртуальная машина должна содержать специальную библиотеку (libOS), управляющую использованием ресурсов.
- Фактически, каждая из таких библиотек дополняет экзоядро до ядра обычной ОС, но при этом работает в режиме пользователя, без переключения в режим ядра.
- Это повышает производительность системы.
- Малые размеры экзоядра позволяют обеспечить его надежность (правде, перекладывая проблемы надежности на уровень libOS).
- Работа с устройствами (чтение/запись портов и т.п.) ложится на libOS, а экзоядро только проверяет права доступа данного приложения к указанным портам.

Гипервизоры и виртуализация ОС

- Гипервизор – программа или аппаратная схема, обеспечивающая одновременное выполнение нескольких ОС на одном процессоре.
- Гипервизор управляет физическими ресурсами компьютера, предоставляя их «гостевым» ОС.
- Гипервизор должен обеспечивать:
  - соответствие поведения программ гостевых ОС их поведению в случае единственной ОС (за исключением, возможно, ограничения количества доступных ресурсов);
  - изоляцию ОС друг от друга;
  - достаточно высокую производительность.
- С точки зрения гостевой ОС, она выполняется на отдельном компьютере (виртуальной машине).
- Поэтому гипервизор называют также монитором виртуальных машин.

Типы гипервизоров

- Автономный гипервизор – устанавливается непосредственно на аппаратуру компьютера, не зависит от какой-либо ОС.
- Гипервизор на основе ОС – запускается как программа в среде базовой (хостовой) ОС. Разделяет ресурсы между гостевыми ОС и управляет их исполнением. Но работа с ПУ выполняется хостовой ОС.
- Гибридный гипервизор – состоит из автономного гипервизора, управляющего процессором и основной памятью, и небольшой хостовой ОС, предоставляющей гостевым ОС остальные ресурсы.
- Современные процессоры Intel и AMD имеют средства аппаратной поддержки виртуализации.