Архитектура опреационных систем

Курс АОС.
Тема
«Архитектура ядра»

Москалев А.А.

Два режима исполнения кода

В **режиме ядра** (*kernel mode*)
разрешено выполнение всех инструкций.
Программа имеет доступ к любым аппаратным средствам (физическая память, регистры процессора, устройства ввода-вывода.

В **пользовательском режиме** (*user mode)*
прямой доступ к любым аппаратным средствам (регистрам, памяти, устройствам ввода-вывода) запрещен либо ограничен.

Классическая архитектура ОС

- **Наиболее общим подходом к структуризации ОС является подразделение модулей на две группы:**

- модули, выполняющие **основные** функции ОС - **ядро ОС**;
- модули, выполняющие **вспомогательные** функции ОС.

Модули ядра

Модули ядра выполняют базовые функции ОС:

управление

- процессами,
- памятью,
- файлами,
- устройствами ввода-вывода и др.

Модули ядра

- *Функции модулей ядра - это часто используемые функции ОС*

  ⇒

  Скорость выполнения этих функций определяет производительность всей системы в целом

  ⇒

  Большинство модулей ядра являются резидентными.

Вспомогательные модули ОС

Не относящиеся к ядру модули выполняют полезные, но не являющиеся обязательными функции. Будем называть их вспомогательными.

Обычно вспомогательные модули подразделяются на следующие группы:

- **утилиты** - программы, которые решают отдельные задачи управления и сопровождения компьютерной системы (сжатие, дефрагментация …*);*
- **библиотеки процедур и функций** различного назначения (библиотека математических функций, библиотека функций ввода-вывода и т.д.);
- **программы предоставления пользователю дополнительных услуг** - специальный вариант пользовательского интерфейса, калькулятор, некоторые *игры;*
- **системные обрабатывающие программы** - текстовые и графические редакторы, компиляторы, компоновщики, отладчики.

Вспомогательные модули ОС

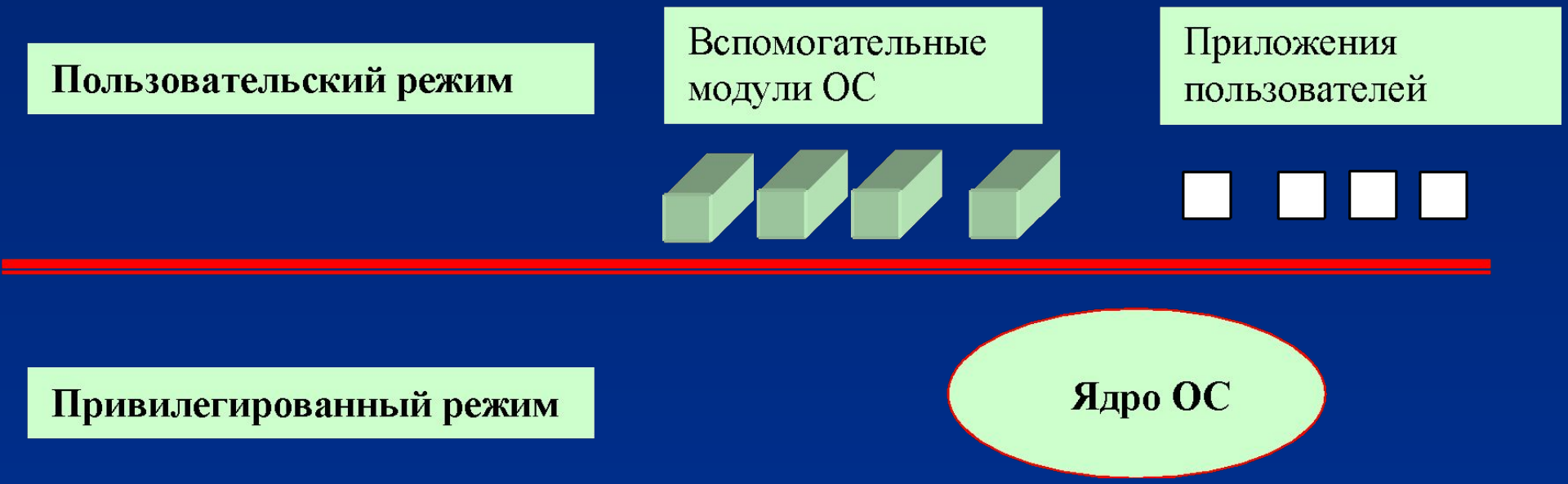
- Вспомогательные модули ОС обращаются к функциям ядра, как и обычные приложения, посредством системных вызовов.
- Вспомогательные модули, в отличие от модулей ядра, являются ***транзитными***.

Классическая архитектура ОС(1)

Архитектуру ОС, основанную на привилегированном ядре и приложениях, выполняемых в пользовательском режиме, называют классической.

Ее используют многие популярные ОС (UNIX, VAX VMS, IBM OS/390, OS/2, Windows NT (с модификациями)).

Классическая архитектура ОС(2)

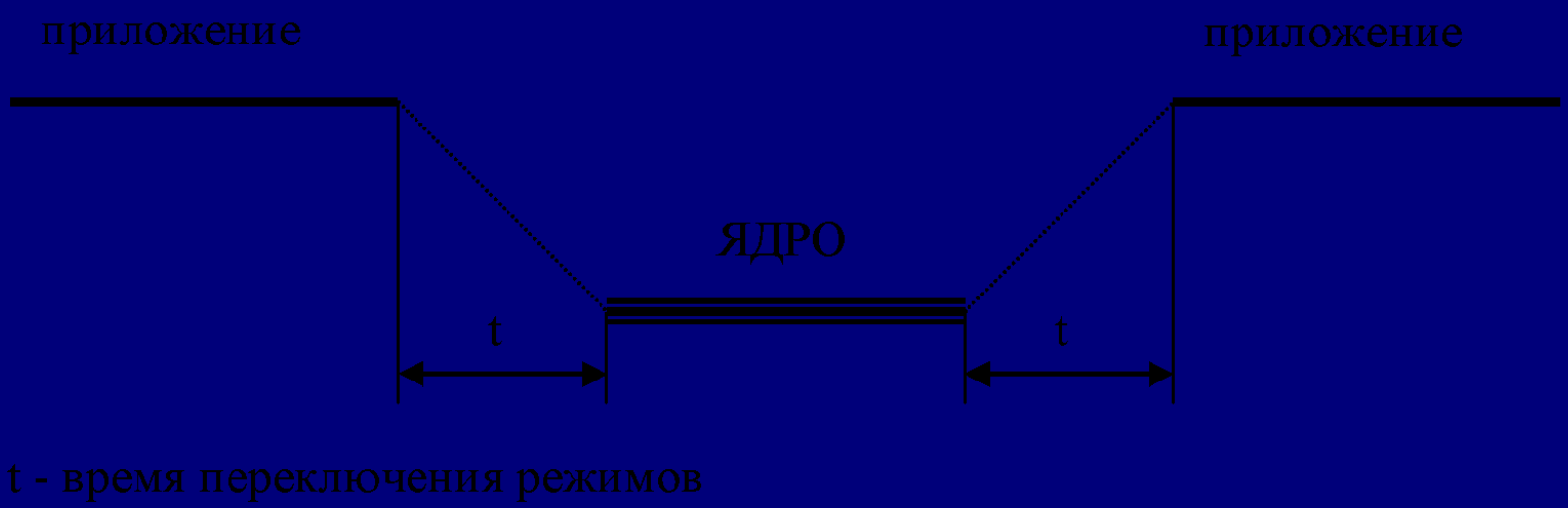


Классическая архитектура ОС(3)

- Термин "ядро" в различных ОС трактуется по-разному. Но чаще всего, именно это свойство -- работа в привилегированном режиме - служит основным определением понятия "ядра".
- Каждое приложение пользовательского режима работает в своем адресном пространстве и защищено тем самым от вмешательства других приложений.
- Код ядра имеет доступ к областям памяти всех приложений, но сам полностью от них защищен.
- Приложения обращаются к ядру с запросами на выполнение системных функций.

Классическая архитектура ОС(4)

Работа системы с привилегированным ядром замедляется за счет затрат времени на переключение режима при выполнения системных вызовов.



Оптимизация производительности (пример)

В операционной системе Novell NetWare работа и ядра и приложений осуществляется в привилегированном режиме

выполнение ядра и других загружаемых модулей системы в привилегированном режиме исключает необходимость переключения режима работы процессора при исполнении программного кода.

РЕЗЮМЕ
(простейшая структурная организация ОС)

Все компоненты ОС разделяются на модули, выполняющие основные функции ОС (ядро), и модули, выполняющие вспомогательные функции ОС.

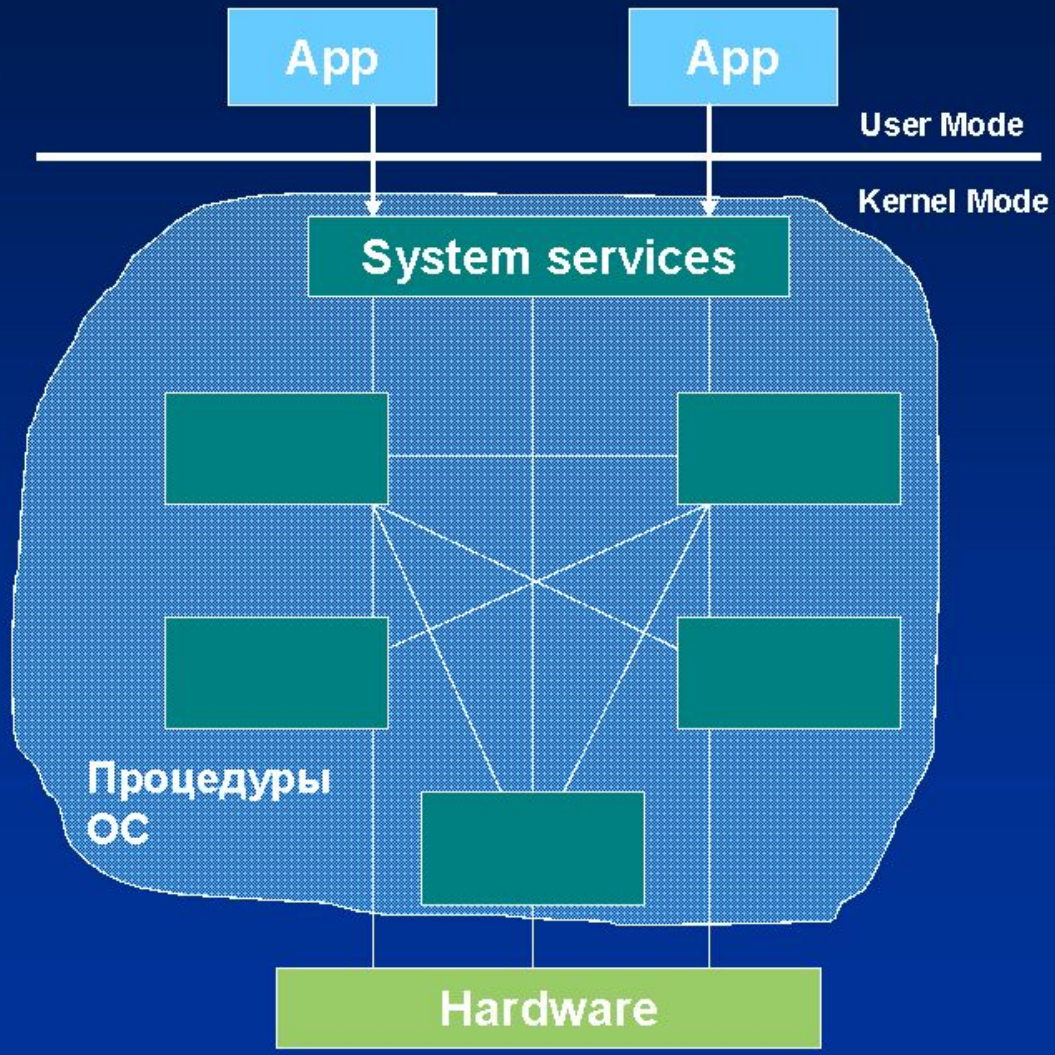
Вспомогательные модули оформляются либо в виде приложений, либо в виде библиотек процедур и функций. Вспомогательные модули являются транзитными. Модули ядра – чаще всего резидентными.

Устойчивость ОС повышается путем выполнения функций ядра в привилегированном режиме, а вспомогательных модулей ОС и пользовательских приложений - в пользовательском.

Архитектура операционных систем

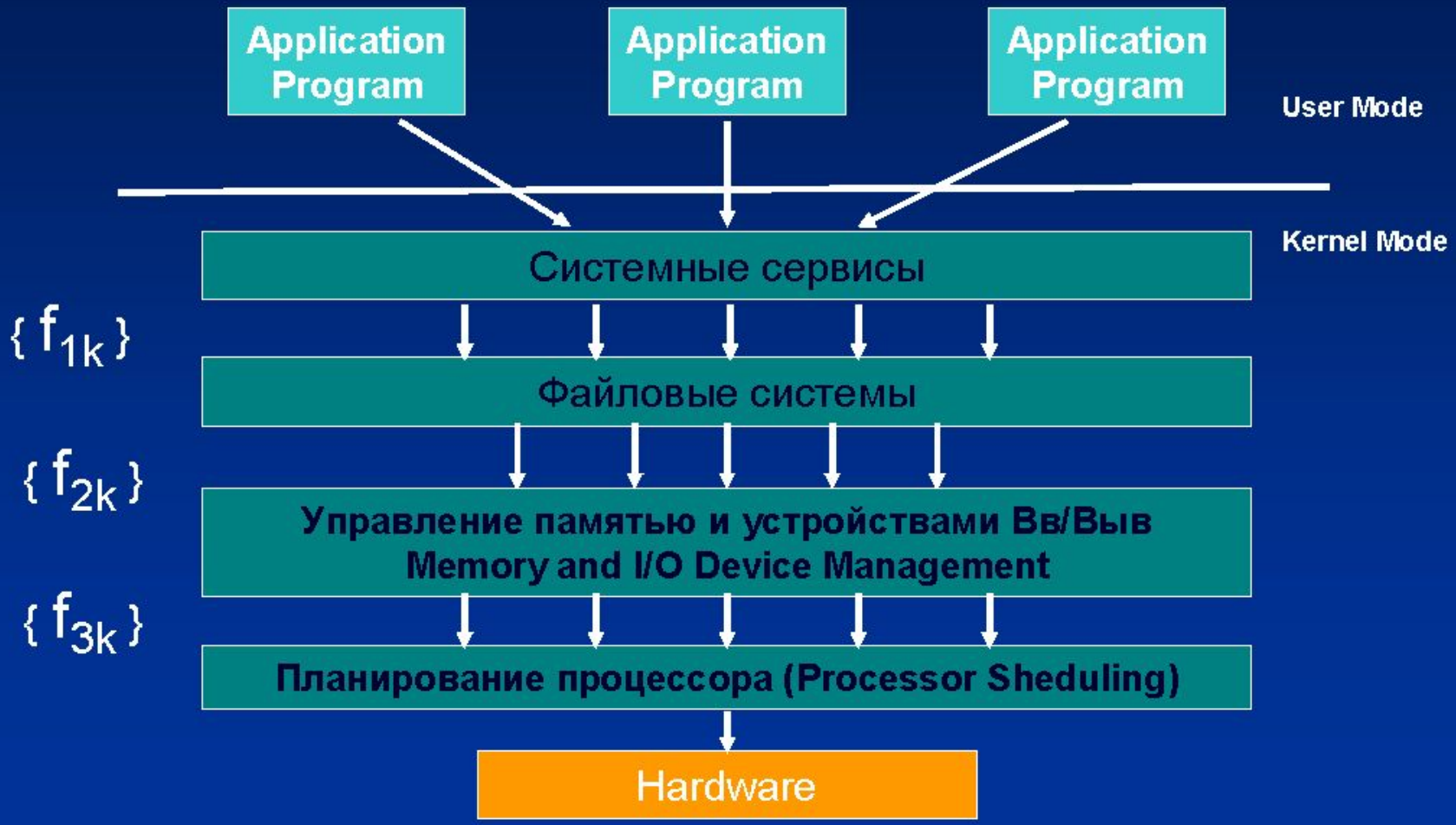
Монолитная архитектура ядра

- Неструктурирована
- При обращении к системе происходит переключение режима пользователя на режим ядра



Слоистая архитектура(Layered OS)

- Каждый слой получает доступ только к интерфейсу нижележащего слоя



Многослойное ядро

- Ядро может быть представлено в виде совокупности следующих уровней (слоев):
- **машинно-зависимые компоненты ОС** - часть функций ОС, выполняемая аппаратными средствами; программные модули, поддерживающие аппаратную платформу;
- **базовые механизмы ядра** - наиболее примитивные операции ядра:программное переключение контекстов процессов, диспетчеризация прерываний, подкачка страниц и т. п.;
- **менеджеры ресурсов** - модули, управляющие основными ресурсами компьютера; обычно это менеджеры процессов, ОП, ввода-вывода, файловой системы)
- **интерфейс системных вызовов** - функции API, обслуживающие системные вызовы.

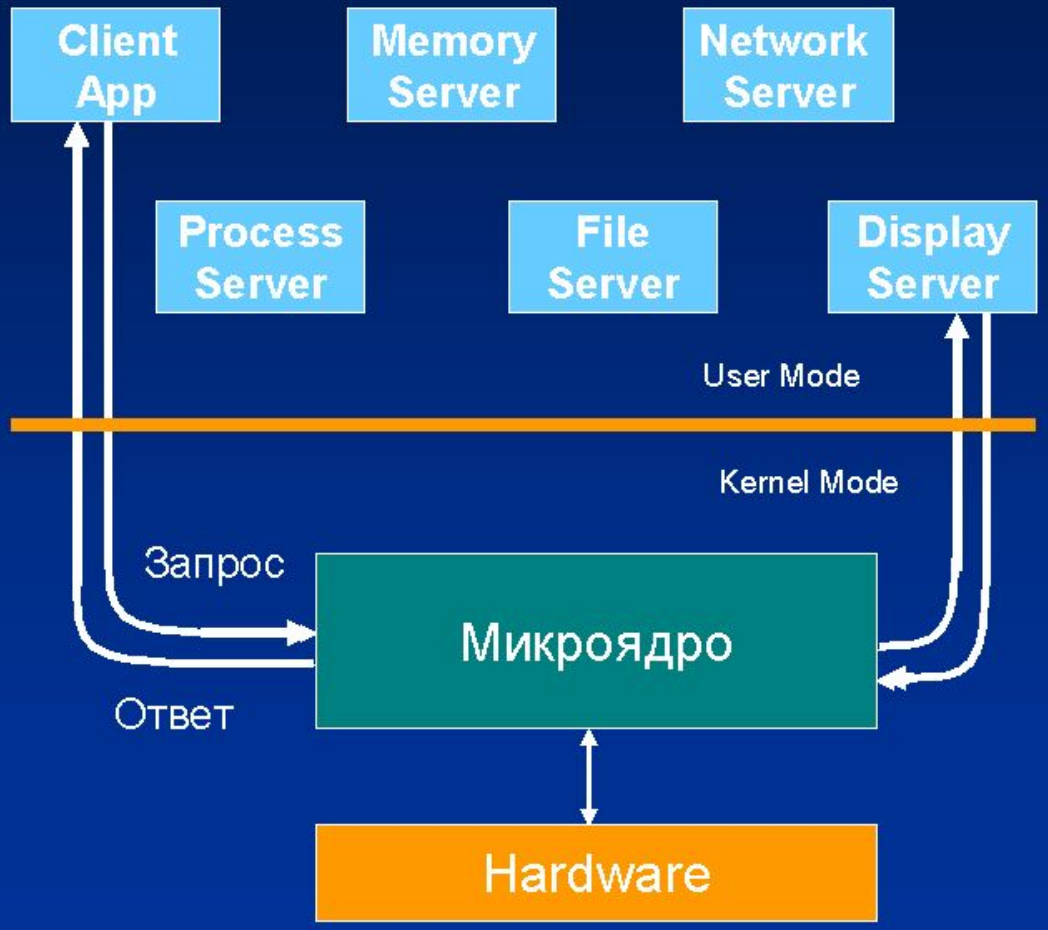
Микроядерная (клиент-серверная) архитектура

Пользовательский режим:

- Клиентские приложения
- Серверы – менеджеры ресурсов

Микроядро:

- Планир - е процессора
- Управление памятью
- Межпроцессные взаимодействия (IPC)



Менеджеры ресурсов: подсистема управления файлами, подсистемы управления виртуальной памятью и процессами, менеджер безопасности и др.

Назначение каждого менеджера - обслуживать запросы других приложений (в том числе других менеджеров) .

Программы, ориентированные на выполнение запросов других программ называются **серверами**.

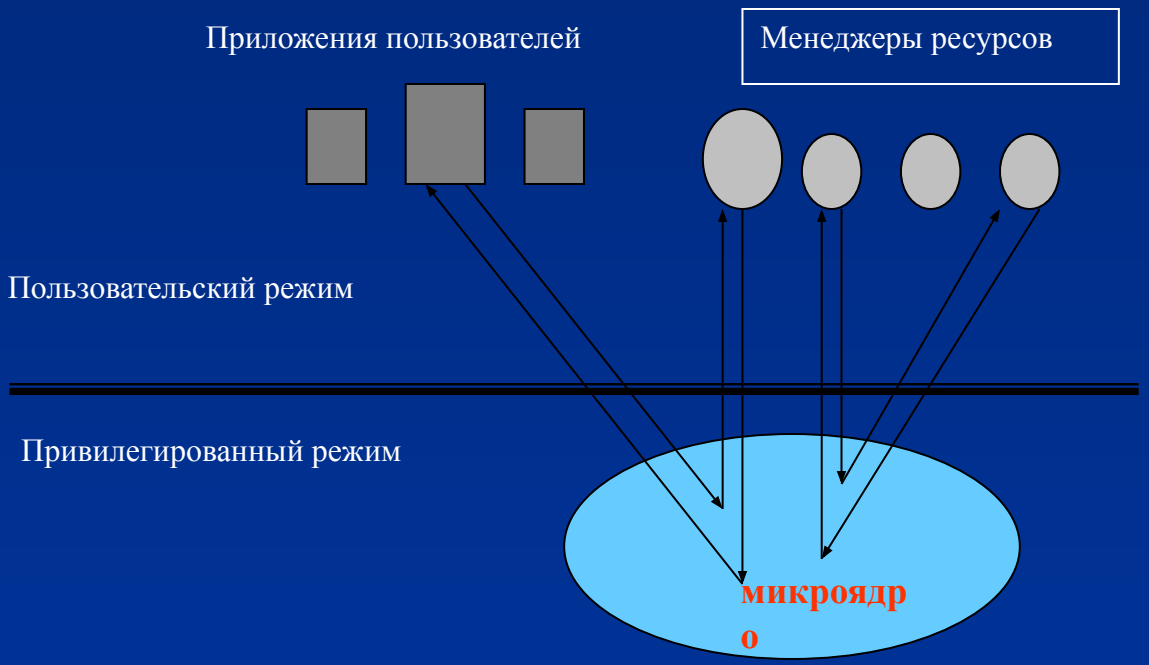
Таким образом менеджеры ресурсов реализуются в виде серверов:файлового сервера; сервера процессов; сервера памяти и др.

Механизм обращения к серверам ОС

Клиент (прикладная программа или другой сервер ОС) запрашивает выполнение некоторой функции у соответствующего сервера, посылая ему сообщение.

! Непосредственная передача сообщений между приложениями/серверами невозможна, так как их адресные пространства изолированы друг от друга.

Только микроядро имеет доступ к адресным пространствам каждого из приложений, поэтому может работать в качестве посредника.



Преимущества микроядерной архитектуры

Микроядерные ОС удовлетворяют большинству требований, предъявляемым к современным ОС:

- **обладают переносимостью** (весь машинно-зависимый код изолирован в микроядре ⇒ необходимо мало изменений при переносе системы на новый процессор, к тому же все изменения сгруппированы вместе)
- **высокая степень расширяемости** (для того, чтобы добавить новую подсистему требуется разработать новое приложение, для чего не требуется затрагивать микроядро; с другой стороны, пользователь легко может удалить ненужные подсистемы, удалять из ядра было бы сложнее)

Преимущества микроядерной архитектуры

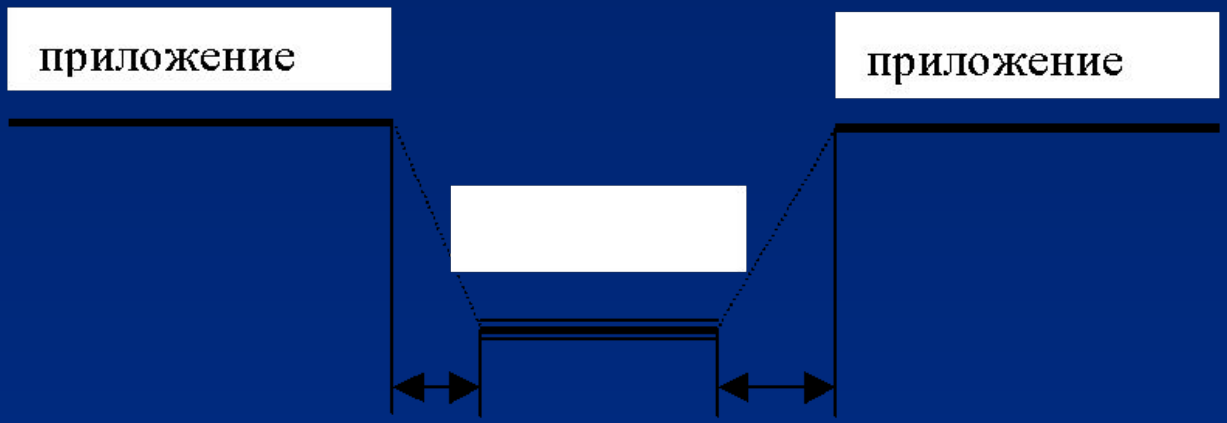
- ✓ **достаточная надежность**
  (каждый сервер ОС выполняется в своем адресном пространстве и таким образом защищен от других серверов, кроме того, если происходит крах одного сервера, он может быть перезапущен без останова или повреждения других серверов).

- ✓ **поддержка распределенных вычислений**
  (серверы ОС могут работать на разных компьютерах - асимметричная ОС; возможен перенос однопроцессорных ОС на распределенные компьютеры)

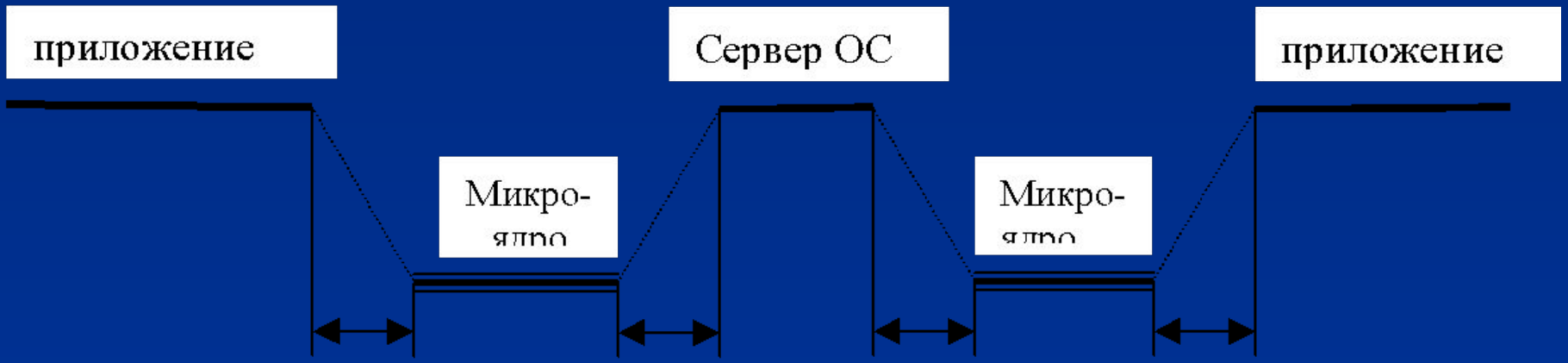
Недостатки микроядерной архитектуры

Основной (и по сути единственный) недостаток микроядерной архитектуры - **снижение производительности**.

При классической организации ОС выполнение системного вызова сопровождается двумя переключениями режимов:



А при микроядерной организации - четырьмя:



Однако, при современных скоростях процессоров снижение производительности в 20%

РЕЗЮМЕ
(микроядерная архитектура ОС)

⇒ Микроядерная архитектура является альтернативой классическому способу построения ОС.

⇒ В привилегированном режиме остается работать только маленькая часть ОС - микроядро. Все остальные функции ядра оформляются в виде приложений, работающих в пользовательском режиме.

⇒ Микроядерные ОС обладают переносимостью, расширяемостью, надежностью, поддерживают распределенные приложения. За эти достоинства приходится платить снижением производительности.

Представитель микроядерных ОС

ОС реального времени QNX -

наиболее яркий представитель микроядерных ОС

Микроядро QNX поддерживает только
планирование и диспетчеризацию процессов,
взаимодействие процессов,
обработку прерываний и
сетевые службы нижнего уровня.

(Несколько десятков системных вызовов. Объем ядра - 8-46 Кб.)

Структура ОС Windows 2000

ОС состоит из двух основных частей:
- сама ОС, работающая в режиме ядра,
- подсистемы окружения, менеджеры ресурсов, вспомогательные службы, работающие в режиме пользователя.

Базовые механизмы, файловая система и другие основные компоненты системы постоянно находятся в режиме ядра. Практически вся ОС помещена в пространство ядра.

Смешанная архитектура -- архитектуру **Windows 2000** можно **отнести и к монолитной**, и к микроядерной.

Пользовательский режим

Привилегированный режим

Для сокрытия аппаратных различий предназначен специальный уровень аппаратных абстракций (HAL, Hardware Abstraction Layer). Работа уровня HAL заключается в том, чтобы предоставить всей остальной системе абстрактные аппаратные устройства. Эти устройства представляются в виде машинно-независимых служб (вызовы процедур и макросы), которые могут использоваться остальной ОС и драйверами. Т.е. весь машинно-зависимый код вынесен на уровень HAL.

Истоки Windows NT

- **Разработка началась в конце 1988 начале 1989 после того как Дэйв Гатлер (Dave Cutler) и группа сотрудников DEC (Digital Equipment Corp.) перешла в Microsoft**
  - **Dave Cutler**— легенда в мире операционных систем
    - Руководитель проекта Digital's VMS (Virtual Memory System)
  - **Внутренне Windows NT имеет много общего с Digital's VMS (планирование, управление памятью, ввод/вывод, модель драйвера и др.)**
  - **VMS+1=WNT** (чистое совпадение?)
- **Исходной целью была замена OS/2**
  - **Позднее целью стала замена Windows 3.0**
- **Происхождение имени "Windows NT"**
  - **NT сокращение от New Technology**
    - По большому счету архитектура и пользовательский интерфейс не были действительно "новыми" (по сравнению с большинством 32-разрядных ОС)
  - **Risc - процессор i860, на который изначально NT была ориентирована, имел кодовое название N-Ten**
- **Интересная книга о ранних годах Windows NT:**
  - Show-stopper!: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft
  - By G. Pascal Zachary, ISBN: 0029356717

Замечание: "Windows" обозначает Windows 2000, Windows XP, and Windows Server 2003

История релизов

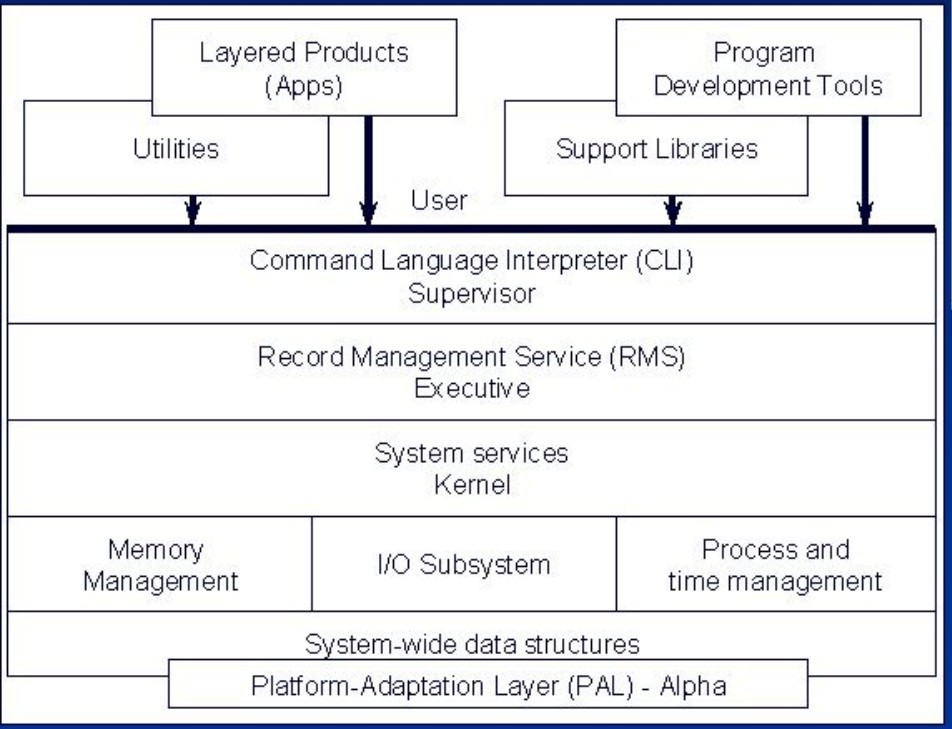
- **Хотя имена продуктов различаются, существует внутренняя идентификация версии по номеру сборки (“build number”)**
  - Номер сборки увеличивается всякий раз, когда осуществляется сборка NT из исходного кода (5-6 раз за неделю)
  - История на временной оси: http://windows2000.about.com/library/weekly/aa010218a.htm

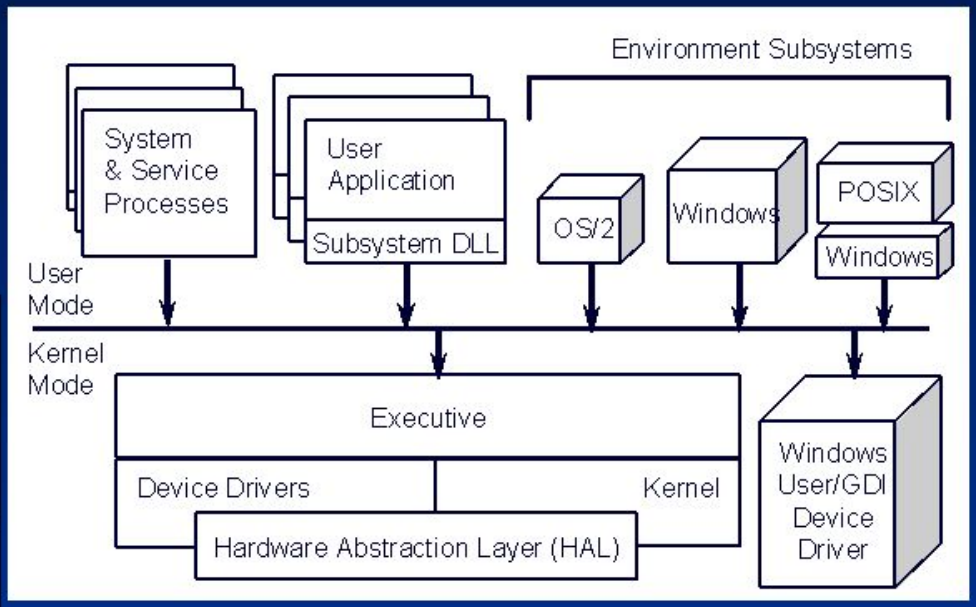
Build#	Version	Date
297	PDC developer release	Jul 1992
511	NT 3.1	Jul 1993
807	NT 3.5	Sep 1994
1057	NT 3.51	May 1995
1381	NT 4.0	Jul 1996
2195	Windows 2000 (NT 5.0)	Dec 1999
2600	Windows XP (NT 5.1)	Aug 2001
3790	Windows Server 2003 (NT 5.2)	Mar 2003
4051	Longhorn PDC Developer Preview	Oct 2003

VMS и Windows

Взгляд на архитектуру с высоты птичьего полета

Многослойная архитектура VAX/VMS





Укрупненная архитектура Windows

Windows And Linux Evolution

- Windows and Linux kernels are based on foundations developed in the mid-1970s

(see http://www.levenez.com for diagrams showing history of Windows & Unix)

Эволюция операционных систем