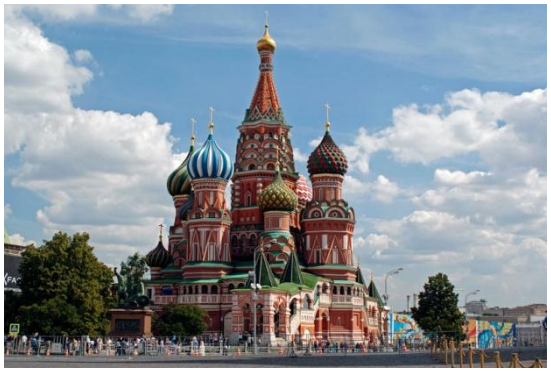
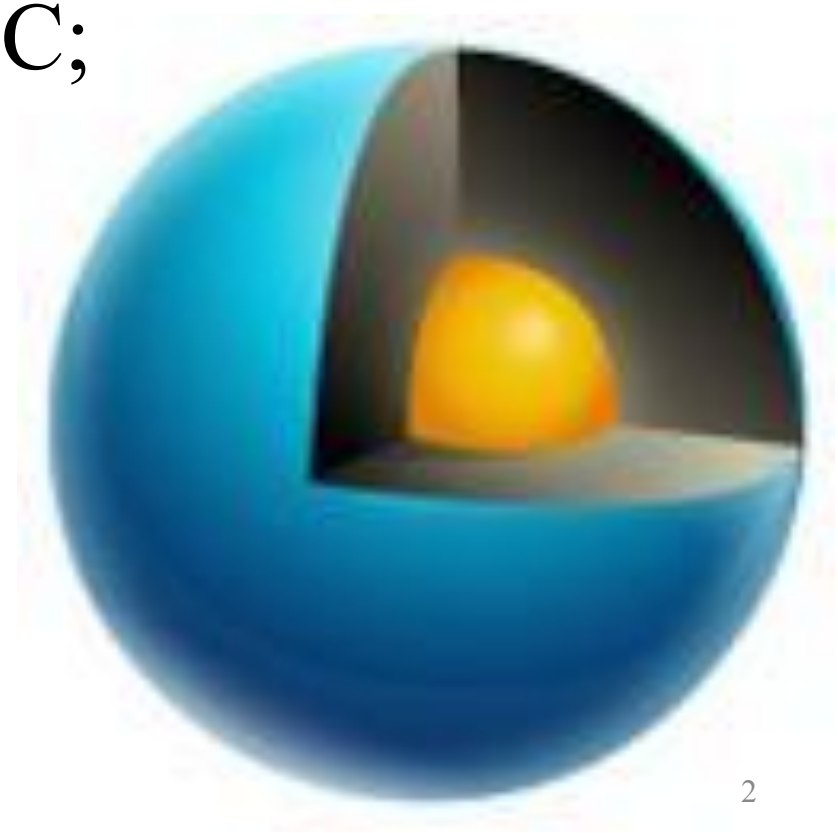


Архитектура операционных систем

Архитектура ОС

Как правило в ОС можно выделить две части:

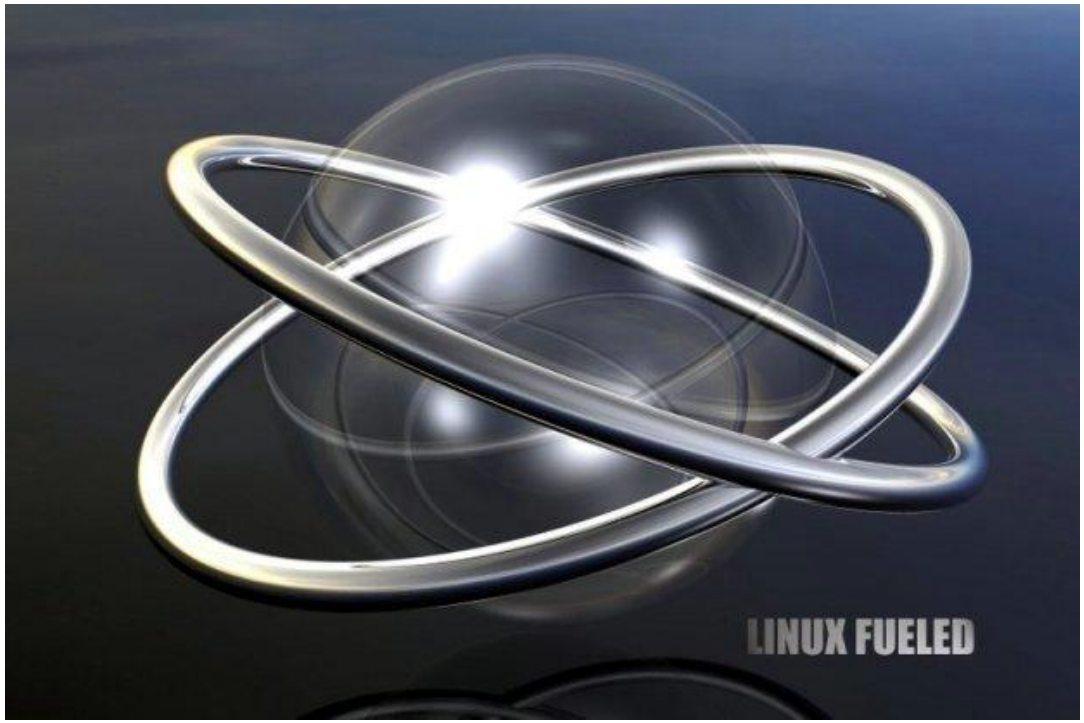
- ***ядро*** – часть ОС, выполняющая основные функции ОС;
- ***вспомогательные модули*** ОС, выполняющие не основные функции.



Архитектура ОС

Ядро выполняет базовые функции ОС:

- управление процессами
- управление памятью
- управление устройствами ввода-вывода
- и т.п.



Архитектура ОС

Функции ядра являются самые часто используемые. И производительность ОС определяется скоростью выполнения этих функций. Поэтому большая часть модулей ядра ОС постоянно находятся в оперативной памяти, т.е. являются ***резидентными***.

Архитектура ОС

Остальные модули тоже выполняют полезные функции, такие как: дефрагментация диска, перераспределение пространства дисков, драйвера устройств и т.п.

Архитектура ОС

Вспомогательные модули оформляются или в виде приложений или как библиотеки процедур.

Архитектура ОС

Среди вспомогательных модулей можно выделить:

- утилиты – программы, решающие отдельные задачи управления конкретным элементом, например: программа дефрагментация диска;
- системные обрабатывающие программы – компиляторы, компоновщики, отладчики;
- программы предоставления пользователям дополнительных услуг – специальный вариант пользовательского интерфейса, калькулятор;
- библиотеки процедур, которые ускоряют разработку программных приложений: библиотеки математических функции, функций ввода-вывода.

Архитектура ОС

Вспомогательные модули загружаются в память только во время выполнения и называются ***транзитными***.

Архитектура ОС

Для надежного функционирования, ОС должна иметь **привилегии** по отношению к пользовательским приложениям. Иначе, некорректно работающие приложения могут вмешаться в работу ОС и нарушить работу алгоритма или даже разрушить код ОС.

Также ОС должна обладать исключительными полномочиями, чтобы играть роль арбитра в споре приложений за ресурсы вычислительной системы.

Архитектура ОС

Такие привилегии для ОС нельзя обеспечить без поддержки аппаратной части компьютера. Эти аппаратные средства должны поддерживать два режима работы:

- ***пользовательский*** режим (user mode);
- ***привилегированный*** режим, который также называют режимом ***ядра*** (kernel mode) или режимом ***супервизора*** (supervisor mode).

Архитектура ОС

- ***пользовательский*** режим (user mode);
- ***привилегированный*** режим или режим ***супервизора*** (supervisor mode).

Архитектура ОС

Приложения, работающие в пользовательском режиме, ***запрещено*** выполнение некоторых критичных команд, например: *переключение процессора с задачи на задачу, управление устройствами ввода-вывода, доступ к распределению памяти* и т.п.

Архитектура ОС

При ***работе с памятью*** на приложения тоже накладываются ограничения: инструкция приложения может обращаться только в области памяти, выделенной этому приложению, и запрещается обращение к областям памяти, которые выделены под ОС или другим приложениям. Для этого используется механизм ***адресного пространства*** конкретного процесса.

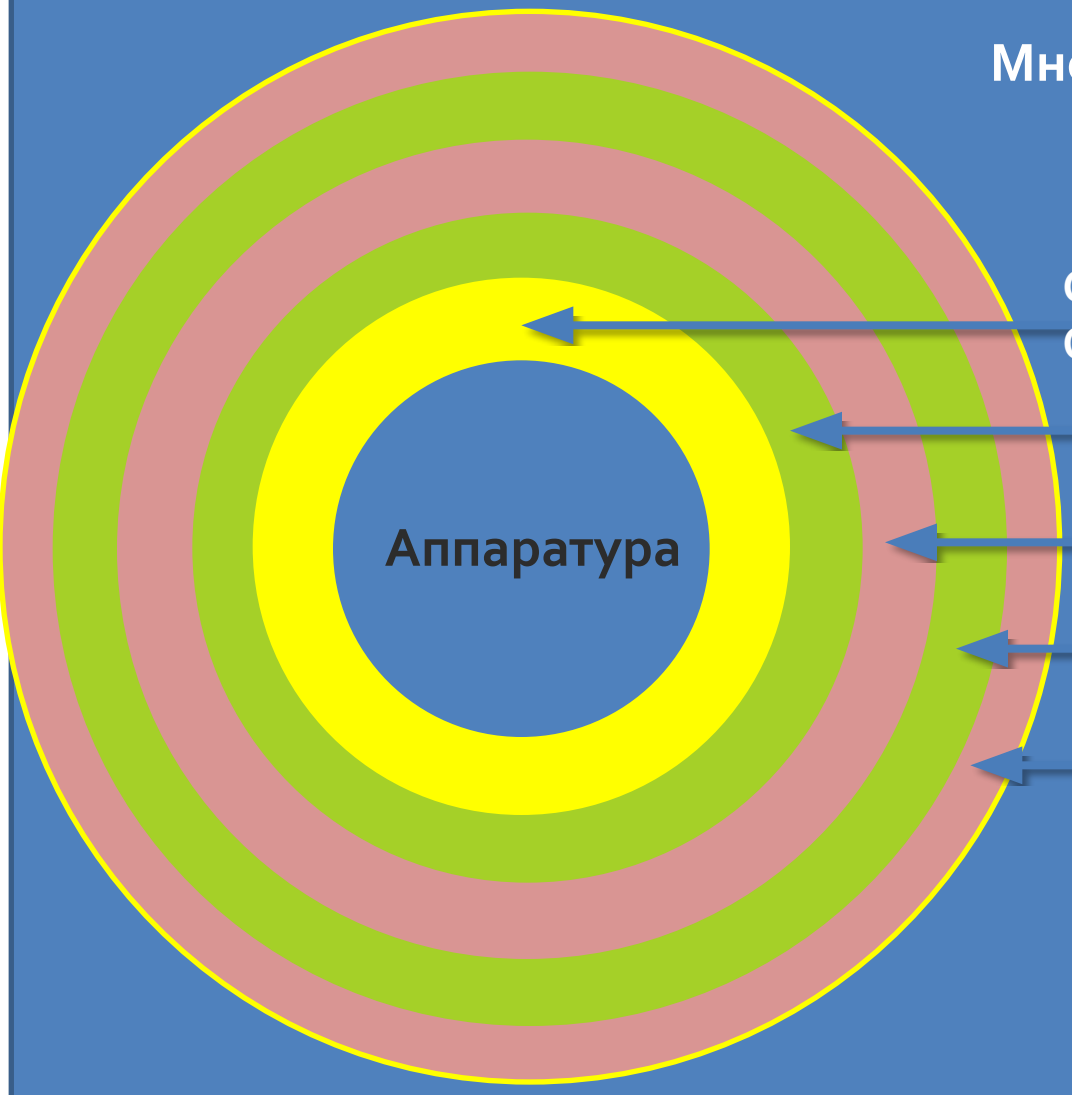
Архитектура ОС

Вычислительную систему, работающую под управлением ОС на основе ядра, можно рассматривать как многослойную систему, состоящую из слоев:

- аппаратура;
- ядро ОС;
- утилиты и обрабатывающие программы.

Архитектура ОС

Многослойная структура ядра ОС



Средства аппаратной поддержки ОС

Машинно-зависимые модули

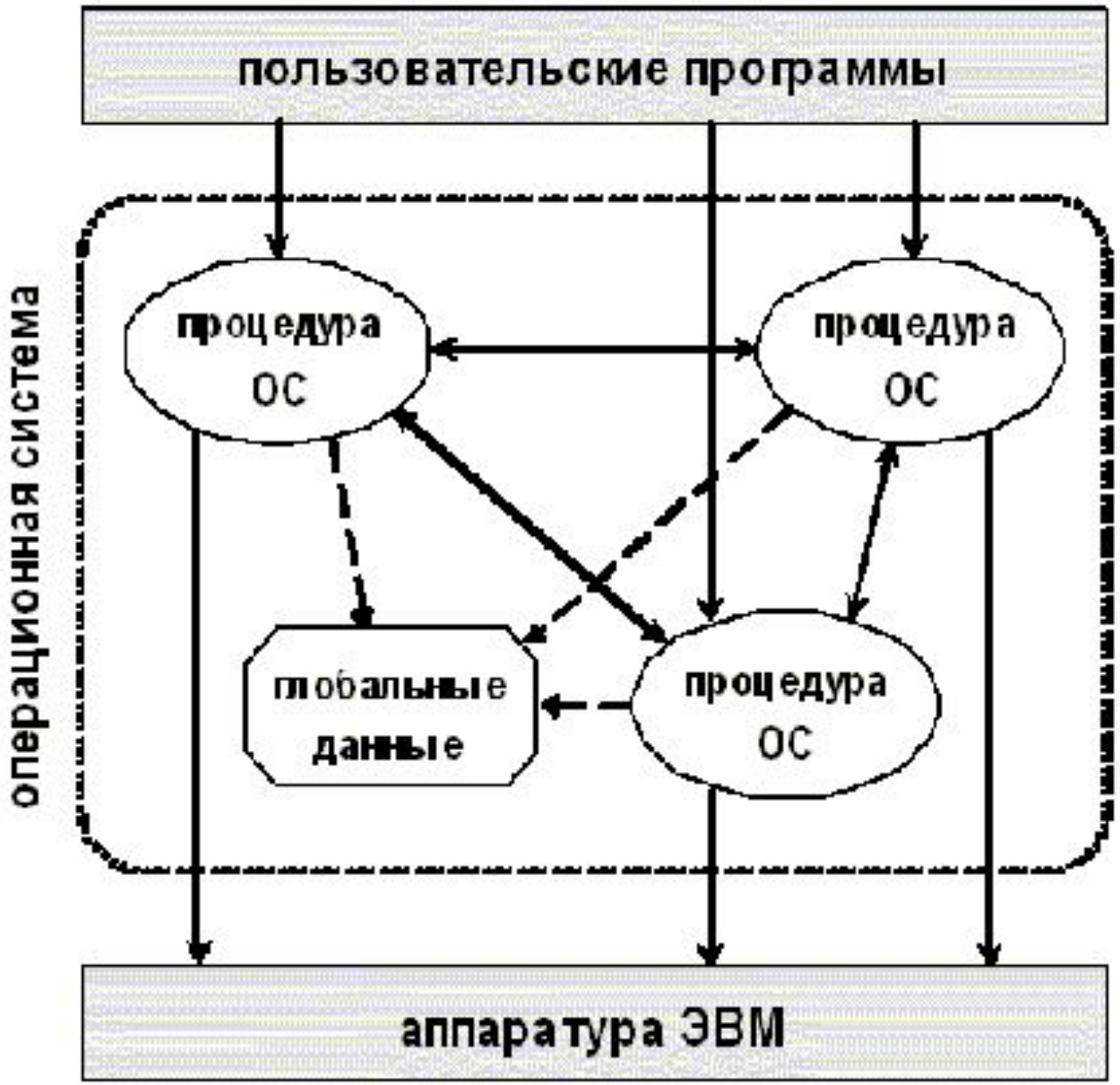
Базовые механизмы ядра

Менеджеры ресурсов

Интерфейс системных вызовов

Архитектура ОС

ОС как правило состоит из иерархии слоев. Каждый слой обслуживает вышележащий слой, выполняя для него некоторый набор функций (примитивы). Используя такие примитивы, вышележащий слой строит свои, более сложные функции, которые в свою очередь будут примитивами для следующего слоя.


пользовательские программы
операционная система
процедура ОС
процедура ОС
глобальные данные
процедура ОС
аппаратура ЭВМ

Функции включаемые в ядро

- Машиннозависимые программы (поддержка нескольких процессов)
- Некоторые функции управления процессами
- Обработка прерываний
- Поддержка пересылки сообщений
- Некоторые функции управления устройствами ввода-вывода, связанные с загрузкой команд в регистры устройств

Архитектура ОС

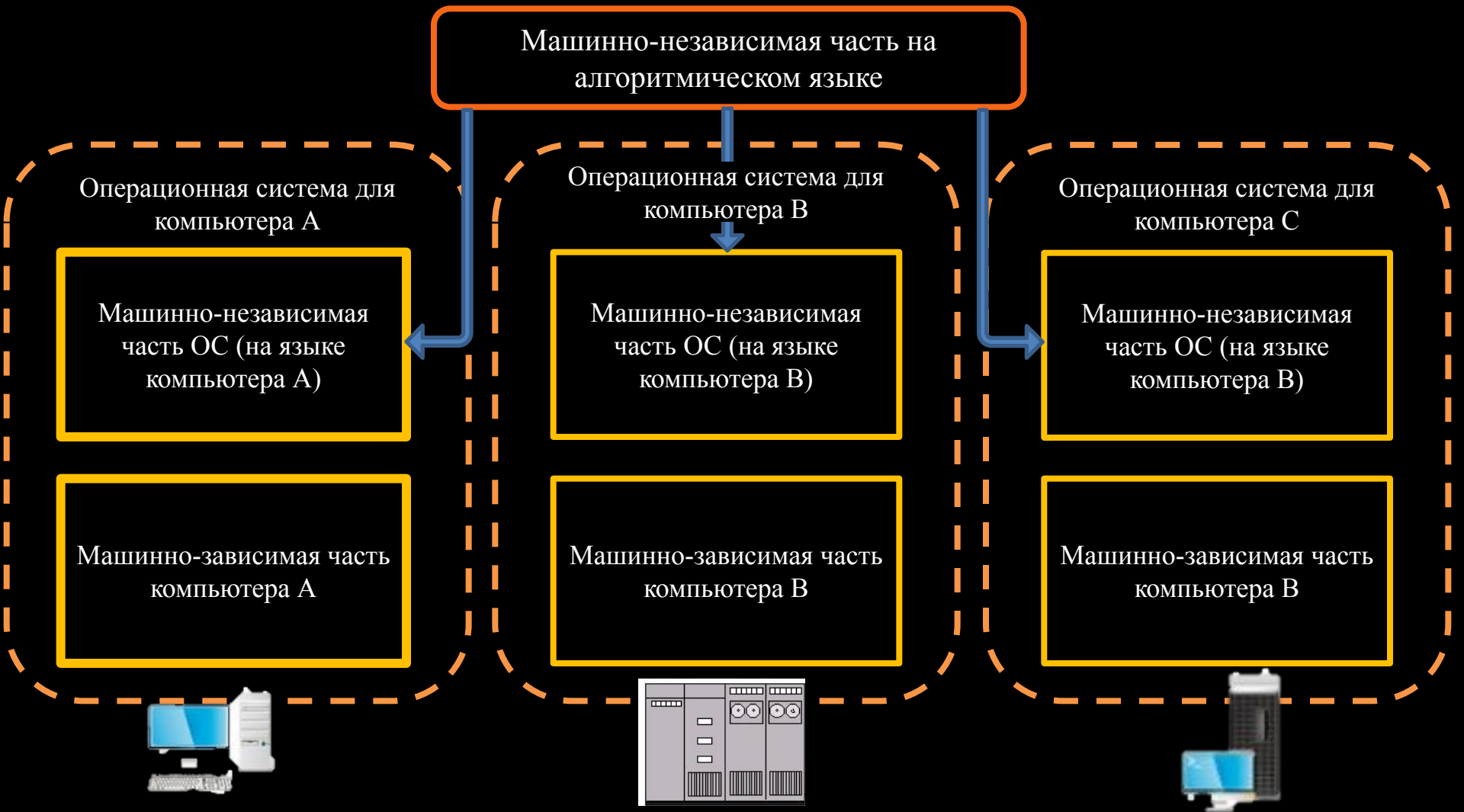
Ядро может состоять из следующих слоев.

- *Средства аппаратной поддержки ОС.* До сих пор об операционной системе говорилось как о комплексе программ, но, вообще говоря, часть функций ОС может выполняться и аппаратными средствами. Поэтому иногда можно встретить определение операционной системы как совокупности программных и аппаратных средств, что и отражено на рис. 3.8. К операционной системе относят, естественно, не все аппаратные устройства компьютера, а только средства аппаратной поддержки ОС, то есть те, которые прямо участвуют в организации вычислительных процессов: средства поддержки привилегированного режима, систему прерываний, средства переключения контекстов процессов, средства защиты областей памяти и т. п.
- *Машинно-зависимые компоненты ОС.* Этот слой образуют программные модули, в которых отражается специфика аппаратной платформы компьютера. В идеале этот слой полностью экранирует вышележащие слои ядра от особенностей аппаратуры. Это позволяет разрабатывать вышележащие слои на основе машинно-независимых модулей, существующих в единственном экземпляре для всех типов аппаратных платформ, поддерживаемых данной ОС. Примером экранирующего слоя может служить слой HAL операционной системы Windows NT.
- *Базовые механизмы ядра.* Этот слой выполняет наиболее примитивные операции ядра, такие как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно и т. п. Модули данного слоя не принимают решений о распределении ресурсов — они только отрабатывают принятые «наверху» решения, что и дает повод называть их исполнительными механизмами для модулей верхних слоев. Например, решение о том, что в данный момент нужно прервать выполнение текущего процесса А и начать выполнение процесса В, принимается менеджером процессов на вышележащем слое, а слою базовых механизмов передается только директива о том, что нужно выполнить переключение с контекста текущего процесса на контекст процесса В.
- *Менеджеры ресурсов.* Этот слой состоит из мощных функциональных модулей, реализующих стратегические задачи по управлению основными ресурса-

Архитектура ОС

исполнения принятых решений менеджер обращается к нижележащему слою базовых механизмов с запросами о загрузке (выгрузке) конкретных страниц. Внутри слоя менеджеров существуют тесные взаимные связи, отражающие тот факт, что для выполнения процессу нужен доступ одновременно к нескольким ресурсам — процессору, области памяти, возможно, к определенному файлу или устройству ввода-вывода. Например, при создании процесса менеджер процессов обращается к менеджеру памяти, который должен выделить процессу определенную область памяти для его кодов и данных.

- *Интерфейс системных вызовов.* Этот слой является самым верхним слоем ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы. Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения. Например, в операционной системе UNIX с помощью системного вызова `fd = open("/doc/a.txt", O_RDONLY)` приложение открывает файл a.txt, хранящийся в каталоге /doc, а с помощью системного вызова `read(fd, buffer, count)` читает из этого файла в область своего адресного пространства, имеющую имя `buffer`, некоторое количество байт. Для осуществления таких комплексных действий системные вызовы обычно обращаются за помощью к функциям слоя менеджеров ресурсов, причем для выполнения одного системного вызова может понадобиться несколько таких обращений.


Машинно-независимая часть на алгоритмическом языке
Операционная система для компьютера А
Машинно-независимая часть ОС (на языке компьютера А)
Машинно-зависимая часть компьютера А
Операционная система для компьютера В
Машинно-независимая часть ОС (на языке компьютера В)
Машинно-зависимая часть компьютера В
Операционная система для компьютера С
Машинно-независимая часть ОС (на языке компьютера В)
Машинно-зависимая часть компьютера В

Микроядерные операционные системы

Микроядро – минимальная стержневая часть операционной системы, служащая основой модульных и переносимых расширений.

Ядро ОС (микроядро), работая в привилегированном режиме, доставляет сообщение нужному серверу, сервер выполняет операцию, после чего ядро возвращает результаты клиенту с помощью другого сообщения

В микроядре содержится и исполняется минимальное количество кода, необходимое для реализации основных системных вызовов:

- передача сообщений;
- организация взаимодействия между внешними по отношению к микроядру процессами;
- поддержка управления прерываниями и др.

Микроядро обеспечивает пять различных типов сервисов:

- задания и потоки;
- межпроцессные коммуникации;
- управление вводом/выводом и прерываниями;
- управление виртуальной памятью;
- сервисы набора хоста и

В качестве приложения ядра работают следующие подсистемы и функции операционной системы:

- система управления файлами;
- поддержка внешних устройств;
- традиционные программные интерфейсы.

Структура ОС Windows

Рис. 11.4. Организация режима ядра Windows

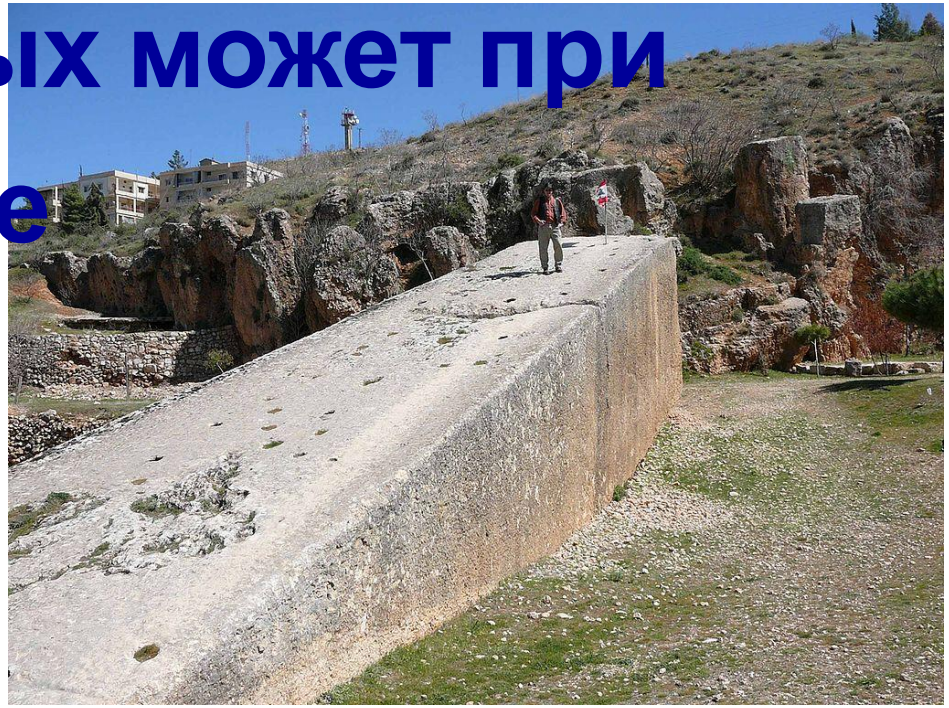
Монолитные операционные системы

Монолитные операционные системы являются прямой противоположностью микроядерным.

Монолитная ОС

Монолитная система представляет собой отсутствие структуры.

ОС написана как набор процедур, каждая из которых может при необходимости вызывать другие процедуры.



Для построения монолитной системы необходимо скомпилировать все отдельные процедуры, а затем связать их вместе в единый объектный файл с помощью компоновщика.

Структура монолитной ОС:

- 1. Главная программа, которая вызывает требуемые сервисные процедуры.
- 2. Набор сервисных процедур, реализующих системные вызовы.
- 3. Набор утилит, обслуживающих сервисные процедуры.

Система THE (Дейкстра)

- 0 – управление временем ЦП, переключение по прерыванию или истечению времени
- 1 – управление памятью, функции виртуальной памяти
- 2 – Связь между консолью оператора и процессами
- 3 – Управление устройствами ввода-вывода и буферизация потоков информации к ним
- 4 – пользовательские программы
- 5 – процесс системного оператора

Проблемы монолитных систем

- Сложность модификации и развития операционной системы.
- Переход к модели клиент-сервер и концепции микроядра

Преимущество микроядерной архитектуры перед монолитной заключается в том, что каждый компонент системы представляет собой самостоятельный процесс, запуск и останов которого не отражается на работоспособности остальных процессов.

Основные принципы построения

Основные принципы построения

- Принцип модульности;
- Принцип генерируемости;
- Принцип функциональной избыточности;
- Принцип виртуализации;
- Принцип независимости программ от внешних устройств
- Принцип совместимости;
- Принцип открытой и наращиваемой ОС;
- Принцип мобильности;
- Принцип обеспечения безопасности вычислений.

Принцип модульности

Модуль – функционально законченный элемент системы, отвечающий требованиям межмодульного интерфейса.

Принцип особого режима работы

Ядро операционной системы и низкоуровневые драйверы должны работать в специальном режиме работы.

Это повышает надежность выполнения вычислений.

Часть команд и команды обращения к специальным системным регистрам должны быть доступны только в привилегированном режиме.

Типовые средства аппаратной поддержки ОС

- Средства поддержки привилегированного режима
- Средства трансляции адресов
- Средства переключения процессов
- Система прерываний
- Системный таймер
- Средства защиты областей памяти

Принцип генерируемости ОС

Принцип генерируемости - возможность настраивать системную супервизорную часть (ядро и основные компоненты), исходя из конкретной конфигурации вычислительного комплекса и класса решаемых задач

Принцип функциональной избыточности

Принцип функциональной избыточности дает возможность проведения одной и той же работы различными способами

Принцип виртуализации

Принцип виртуализации позволяет представить структуру системы в виде определенного набора планировщиков процессов и распределителей ресурсов и использовать единую централизованную схему

Принцип независимости программ от внешних устройств

Принцип независимости заключается в том, что связь программы с конкретными устройствами производится не на уровне трансляции программы, а в период планирования ее

Принцип совместимости

Одним из аспектов **совместимости** является способность ОС выполнять программы, написанные:

- для других ОС;
- для более ранних версий данной операционной системы;
- для другой аппаратной платформы

Принцип открытой и наращиваемой ОС

Открытая ОС доступна для анализа как системным специалистам, обслуживающим вычислительную систему, так и пользователям.

Наращиваемая ОС позволяет не только использовать возможности генерации, но и вводить в состав ОС новые модули, совершенствовать

старые и т. д.

Принцип мобильности (переносимости)

Операционная система должна относительно легко переноситься:

- **с процессора одного типа на процессор другого типа;**
- **с аппаратной платформы (архитектуры вычислительной системы) одного типа на аппаратную платформу другого типа.**

Принцип обеспечения безопасности вычислений

Правила безопасности определяют следующие свойства:

- **защита ресурсов одного пользователя от других;**
- **установка квот по ресурсам для предотвращения захвата одним пользователем всех системных ресурсов.**

Расширение ядра

- Размещение чувствительных к режиму работы процессора серверов в пространстве ядра (в Windows NT кроме микроядра в привилегированном режиме работает executive – управление виртуальной памятью, объектами, вводом-выводом, и файловой системой (включая сетевые драйверы), взаимодействием процессов (планировщик), частично системой безопасности

Коммерческие версии микроядер

- NEXT (Mach)
- Windows NT (процессоры Intel, MIPS и Alpha) (программы для DOS, Windows, OS/2)
- Novell, IBM, Apple, Sun и др.

Монолитные системы Микоядерные

Модель клиент-сервер

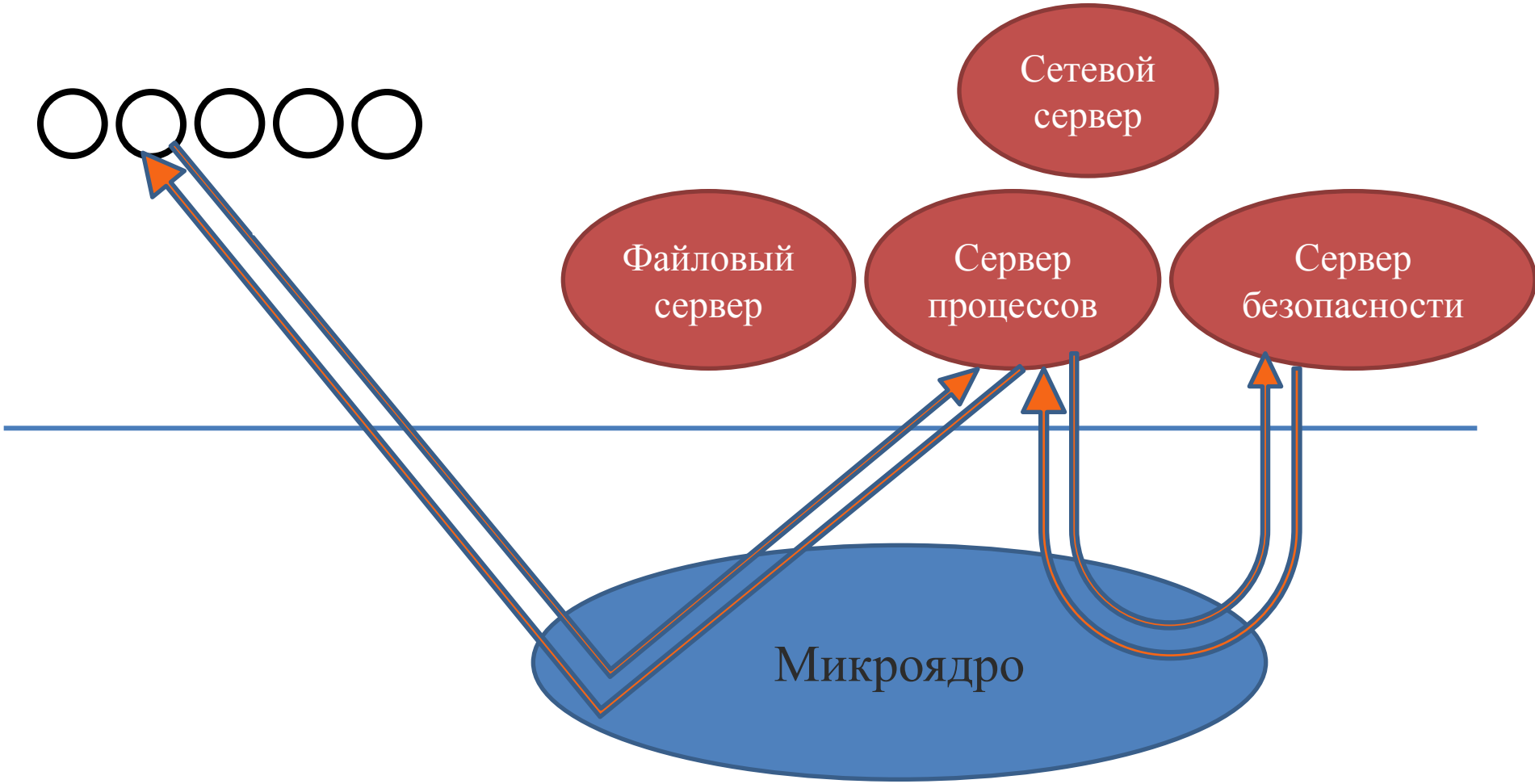
Модель клиент-сервер предполагает наличие программного компонента - потребителя какого-либо сервиса - клиента, и программного компонента - поставщика этого сервиса - сервера.

Взаимодействие между клиентом и сервером стандартизуется, так что сервер может обслуживать клиентов, реализованных различными способами и, может быть, разными производителями.

- Один и тот же программный компонент может быть клиентом по отношению к одному виду услуг, и сервером для другого вида услуг
- Это не столько технология, сколько удобное концептуальное средство ясного представления программных функций того или иного программного элемента или и той или иной ситуации

Структурирование ОС состоит в разбиении ее на несколько процессов - серверов, каждый из которых выполняет отдельный набор сервисных функций - например, управление памятью, создание или планирование процессов.

Каждый сервер выполняется в пользовательском режиме. Клиент, которым может быть либо другой компонент ОС, либо прикладная программа, запрашивает сервис, посылая сообщение на сервер.


Сетевой сервер
Файловый сервер
Сервер процессов
Сервер безопасности
Микроядро

- OC Workplace (IBM)
- OC Windows NT (Microsoft)

Для добавления новых функций и изменения существующих используется технология «сервер – клиент». Взаимодействие между сервером и клиентом стандартизуется, сервер может обслуживать клиентов, реализованных различными способами.

Главное требование – использование единообразного интерфейса.
Инициатором обмена является клиент, который посылает запрос серверу. Один и тот же программный компонент может быть и клиентом, и сервером.