# **1. Операционные системы (лекция 1)**

## **1.1. Понятие операционной системы**

[**Операционная система**](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)**— комплекс**[**программ**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)**, обеспечивающий управление аппаратными средствами**[**компьютера**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)**, организующий работу с**[**файлами**](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB)**и выполнение**[**прикладных программ**](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)**, осуществляющий ввод и вывод данных.**

**Операционная система (ОС) - комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы (ВС) (Вычислительная система - взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации) и удобства работы с ней.**  
  
Назначение ОС - организация вычислительного процесса в вычислительной системе, рациональное распределение вычислительных ресурсов между отдельными решаемыми задачами; предоставление пользователям многочисленных сервисных средств, облегчающих процесс программирования и отладки задач. Операционная система исполняет роль своеобразного интерфейса (Интерфейс - совокупность аппаратуры и программных средств, необходимых для подключения периферийных устройств к ПЭВМ) между пользователем и ВС, т.е. ОС предоставляет пользователю виртуальную ВС. Это означает, что ОС в значительной степени формирует у пользователя представление о возможностях ВС, удобстве работы с ней, ее пропускной способности. Различные ОС на одних и тех же технических средствах могут предоставить пользователю различные возможности для организации вычислительного процесса или автоматизированной обработки данных.

В программном обеспечении ВС операционная система занимает основное положение, поскольку осуществляет планирование и контроль всего вычислительного процесса. Любая из компонент программного обеспечения обязательно работает под управлением ОС.

Сегодня наиболее известными операционными системами являются ОС семейства Microsoft Windows и UNIX-подобные системы.

## **1.2. Перечень некоторых известных операционных систем**

**UNIX**

Операционная система [UNIX](https://ru.wikipedia.org/wiki/UNIX) была разработана группой сотрудников Bell Labs под руководством [Денниса Ричи](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81_%D0%A0%D0%B8%D1%87%D0%B8), [Кена Томпсона](https://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D0%BD_%D0%A2%D0%BE%D0%BC%D0%BF%D1%81%D0%BE%D0%BD) и [Брайана Кернигана](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D0%B9%D0%B0%D0%BD_%D0%9A%D0%B5%D1%80%D0%BD%D0%B8%D0%B3%D0%B0%D0%BD) в 1969 году.

**BSD**

В конце 1970-х годов сотрудники Калифорнийского университета в Беркли внесли ряд усовершенствований в исходные коды UNIX, включая работу с протоколами [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP). Их разработка стала известна под именем [BSD](https://ru.wikipedia.org/wiki/BSD) — Berkeley Systems Distribution. Она распространялась под лицензией, позволяющей дорабатывать и совершенствовать продукт и передавать результат третьим лицам, с исходными кодами или без них, при условии указания авторства кода, написанного в Беркли.

**Linux**

В начале 1990-х годов студент Хельсинкского университета [Линус Торвальдс](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D1%83%D1%81_%D0%A2%D0%BE%D1%80%D0%B2%D0%B0%D0%BB%D0%B4%D1%81" \o "w:Линус Торвалдс) начал разработку ядра новой ОС для IBM-совместимых ПК, которое было названо [Linux](https://ru.wikipedia.org/wiki/Linux" \o "w:Linux). В настоящее время GNU/Linux (совокупность различных дистрибутивов построенных на базе ядра Linux) стоит на втором месте по популярности среди ОС, используемых на рабочих столах пользователей (первое место принадлежит Microsoft Windows).

**Amiga OS**

Операционная система для персональных компьютеров семейства [Amiga](https://ru.wikipedia.org/wiki/Amiga" \o "w:Amiga) (процессор Motorola 68k), имеет атипичное микроядро называемое Exec.

**DOS**

В 1980 Тимом Патерсоном (Tim Paterson) из Seattle Computer Products (SCP) была создана QDOS (Quick and Dirty Operating System). QDOS, по большей части, была 16-разрядным клоном [CP/M](https://ru.wikipedia.org/wiki/CP/M), но с новой файловой системой — [FAT](https://ru.wikipedia.org/wiki/FAT). QDOS была переименована в 86-DOS, поскольку разрабатывалась для работы на процессоре Intel 8086. Microsoft приобрела QDOS за $60 000 и продала её IBM уже как PC-DOS ([MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS)).

1 августа 1984 IBM объявляет о выпуске нового поколения персональных компьютеров — IBM PC/AT.

**Free DOS**

[FreeDOS](https://ru.wikipedia.org/wiki/FreeDOS) — свободно-распространяемая функциональная копия известной операционной системы [MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS).

**FreeBSD**

[FreeBSD](https://ru.wikipedia.org/wiki/FreeBSD) Свободная операционная система семейства [Unix](https://ru.wikipedia.org/wiki/UNIX" \o "w:UNIX), потомок [AT&T Unix](https://ru.wikipedia.org/wiki/System_V) по линии [BSD](https://ru.wikipedia.org/wiki/BSD), создана [Калифорниским Университетом Беркли](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D0%B8%D1%84%D0%BE%D1%80%D0%BD%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%B2_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8" \o "w:Калифорнийский университет в Беркли)

**Mac OS**

[Mac OS](https://ru.wikipedia.org/wiki/Mac_OS) - [Unix](https://ru.wikipedia.org/wiki/Unix" \o "w:Unix) подобная операционная система, разработанна компанией [Apple Computers Co.](https://ru.wikipedia.org/wiki/Apple" \o "w:Apple) в 1984 году, под названием "System 1". В 1997 году, была выпушена 8-я версия операционной системы, и операционная система была переиминована в Mac OS (Полное название: Macintosh Operating System).

**macOS 10**

[macOS](https://ru.wikipedia.org/wiki/macOS_X) - десятый выпуск линейки [Mac OS](https://ru.wikipedia.org/wiki/Mac_OS" \o "w:Mac OS), разрабатываеммая и выпускаемая американской компанией [Apple Computers Co.](https://ru.wikipedia.org/wiki/Apple" \o "w:Apple) (Сегодня: Apple Inc.). В отличее от Mac OS 9, macOS X была разработана на ядре [XNU](https://ru.wikipedia.org/wiki/XNU), и имела код из [FreeBSD](https://ru.wikipedia.org/wiki/FreeBSD" \o "w:FreeBSD). Начиная с версий OS X Yosemite, была введена функция Hand Off. С этой функцией можно было взаимодействовать с iOS и перехватывать данные приложения, и передать данные на iOS устройства c запушенным приложением.

**Microsoft Windows**

[Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) — это семейство операционных систем компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "w:Microsoft). Была создана для компьютеров [IBM](https://ru.wikipedia.org/wiki/IBM) с поддержкой [MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS)

**IBM OS/2**

[OS/2](https://ru.wikipedia.org/wiki/OS/2) — операционная система, разрабатывавшаяся компанией [IBM](https://ru.wikipedia.org/wiki/IBM) (первоначально совместно с [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "w:Microsoft), позже самостоятельно). В настоящее время работы над клиентскими версиями прекращены, в связи с широким распространением операционных систем семейства Windows NT. Серверные версии продолжают поддерживаться в связи с широким ореолом внедрения. Широко использовалась в США, в банковской и производственной сферах, а также в России, в банкоматах.

**ReactOS**

[ReactOS](https://ru.wikipedia.org/wiki/ReactOS) — операционная система, один из проектов сообщества Open Source. В ходе разработки предполагается добиться полной совместимости с приложениями и драйверами **Microsoft Windows(R) NT4.** Это открытая операционная система, основанная на принципах архитектуры Windows NT® (такие продукты компании Microsoft, как Windows XP, Windows 7, Windows Server 2012 построены на архитектуре Windows NT). Система была разработана с нуля, и таким образом не основана на Linux и не имеет ничего общего с архитектурой UNIX.

**Plan 9**

[Plan9](https://ru.wikibooks.org/wiki/Plan9) — Операционная система, разработанная в Bell Labs — колыбели UNIX и языка Си. Построена на идее использования файловых иерархий для представления любых ресурсов операционной системы и оборудования. Идеально подходит для построения распределенных систем.

**Inferno OS**

[Inferno](https://ru.wikibooks.org/wiki/Inferno) — продолжатель идей Plan9, отличительной особенностью которой является малые требования к ресурсам компьютера и возможность работы как поверх установленной ОС, так и самостоятельно. [VitaNuova](http://www.vitanuova.com/)

**Menuet OS**

[Menuet](https://ru.wikibooks.org/w/index.php?title=Menuet&action=edit&redlink=1) - самостоятельная операционная система написанная на языке программирования ассемблер. 64-битная версия коммерческая и требуется оплата.

**Колибри ОС**

[Колибри](https://ru.wikibooks.org/w/index.php?title=%D0%9A%D0%BE%D0%BB%D0%B8%D0%B1%D1%80%D0%B8&action=edit&redlink=1) - ответвление от Menuet OS, в отличие от Menuetа, полностью бесплатная.

**iOS**

[iOS](https://ru.wikibooks.org/w/index.php?title=IOS&action=edit&redlink=1) - (до 24 июня 2010 года — iPhone OS) — мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей и некоторых других устройств, основанная на macOS X, разрабатываемая и выпускаемая американской компанией Apple.

**ANDROID**

[ANDROID](https://ru.wikibooks.org/wiki/ANDROID) - операционная система, основана на ядре Linux, предназначена для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнесс-браслетов, игровых приставок, ноутбуков, нетбуков, смартбуков, Очков Google Glass[3], телевизоров[4] и других устройств (в 2015 году появилась поддержка автомобильных развлекательных систем[5] и бытовых роботов).

## **1.3. Взгляд на ОС с разных точек зрения**

***Операционная система как виртуальная машина***

Архитектура большинства компьютеров на уровне машинных команд очень неудобна для ее использования прикладными программами. Например, работа с диском предполагает знакомство с внутренним устройством его электронного компонента - контроллера для ввода команд вращения диска, поиска и форматирования дорожек, чтения и записи секторов и т.д. Ясно, что средний программист не в состоянии учитывать все особенности работы оборудования (в современной терминологии заниматься разработкой драйверов устройств), а должен иметь простую высокоуровневую абстракцию, скажем, представляя информационное пространство диска как набор файлов

Файл можно открывать для чтения или записи, использовать для получения или сброса информации, а потом закрывать. Это концептуально проще, чем заботиться о деталях перемещения головок дисков или организации работы мотора. Аналогичным образом, с помощью простых и ясных абстракций, скрываются от программиста все ненужные ему подробности организации прерываний, работы таймера, управления памятью и т.д. Более того, на современных вычислительных комплексах может быть создана иллюзия неограниченного размера операционной памяти и числа процессоров. Всем этим занимается операционная система.

С точки зрения пользователя функцией ОС является предоставление пользователю некоторой расширенной или виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину, тем самым скрывая от пользователя детали управления оборудованием (hardware).

Этот принцип иллюстрируется рис.1.



*Рис. 1. Операционная система, процессы, оборудование*

Как видно из рисунка, ОС играет роль "прослойки" между процессами пользователей и оборудованием системы. (Под оборудованием понимаются, как правило, внешние устройства, но можно трактовать этот термин и шире - включая в него все первичные ресурсы). Процессы пользователей не имеют непосредственного доступа к оборудованию и, говоря шире, к системным ресурсам. Если процессу необходимо выполнить операцию с системным ресурсом, в том числе, и с оборудованием, процесс выдает системный вызов. ОС интерпретирует системный вызов, проверяет его корректность, возможно, помещает в очередь запросов и выполняет его. Если выполнение вызова связано с операциями на оборудовании, ОС формирует и выдает на оборудование требуемые управляющие воздействия. Оборудование, выполнив операцию, заданную управляющими воздействиями, сигнализирует об этом прерыванием. Прерывание поступает в ядро ОС, которое анализирует его и формирует отклик для процесса, выдавшего системный вызов. Если выполнение системного вызова не требует операций на оборудовании, отклик может быть сформирован немедленно.

Управляющие воздействия и прерывания составляют интерфейс оборудования, системные вызовы и отклики на них - интерфейс процессов. В качестве синонима интерфейса процессов мы в соответствии со сложившейся в последнее время традицией часто будем употреблять аббревиатуру API (Application Programm Interface - интерфейс прикладной программы).

Отделение процессов пользователя от оборудования преследует две цели.

Во-первых, - безопасность. Если пользователь не имеет прямого доступа к оборудованию и вообще к системным ресурсам, то он не может вывести их из строя или монопольно использовать в ущерб другим пользователям.

Во-вторых, - обеспечение абстрагирования пользователя от деталей управления оборудованием. Вывод на диск, например, требует сложного программирования контролера дискового устройства, однако, все пользователи используют для этих целей простое обращение к драйверу устройства. Более того, в большинстве систем имеются библиотеки системных вызовов, обеспечивающие API для языков высокого уровня (прежде всего - для языка C). Можно также говорить о том, что ОС интегрирует ресурсы: из ресурсов низкого (физического) уровня она конструирует более сложные ресурсы, которые с одной стороны сложнее (по функциональным возможностям), а с другой стороны - проще (по управлению) низкоуровневых.

***Операционная система как менеджер ресурсов***

Операционная система предназначена для управления всеми частями весьма сложной архитектуры компьютера. Представим для примера, что случится, если несколько программ, работающих на одном компьютере, будут пытаться одновременно осуществлять вывод на принтер. Мы получили бы неупорядоченную смесь строчек и страниц, выведенных различными программами. Операционная система предотвращает хаос такого рода за счет буферизации информации, предназначенной для печати, на диске и организации очереди на печать. Для многопользовательских компьютеров, необходимость управления ресурсами и их защиты еще более очевидна.

Следовательно, операционная система как менеджер ресурсов, осуществляет упорядоченное и контролируемое распределение процессоров, памяти и других ресурсов между различными программами, их использующими.

Ресурс - "средство системы обработки данных, которое может быть выделено процессу обработки данных на определенный интервал времени". Простыми словами: ресурс - это все те аппаратные и программные средства и данные, которые необходимы для выполнения программы. Ресурсы можно подразделить на первичные и вторичные. К первой группе относятся те ресурсы, которые обеспечиваются аппаратными средствами, например: процессор, память - оперативная и внешняя, устройства и каналы ввода-вывода и т.п. Ко второй группе - ресурсы, порождаемые ОС, например, системные коды и структуры данных, файлы, семафоры, очереди и т.п. В последнее время в связи с развитием распределенных вычислений и распределенного хранения данных все большее значение приобретают такие ресурсы как данные и сообщения.

Функцией ОС является распределение процессоров, памяти, устройств и данных между процессами, конкурирующими за эти ресурсы. ОС должна управлять всеми ресурсами вычислительной машины таким образом, чтобы обеспечить максимальную эффективность ее функционирования. Критерием эффективности может быть, например, пропускная способность. Управление ресурсами включает решение двух общих, не зависящих от типа ресурса задач:

* планирование ресурса - то есть определение, кому, когда, а для делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;
* отслеживание состояния ресурса – то есть поддержание оперативной информации о том, занят или не занят ресурс, а для делимых ресурсов – какое количество ресурса уже распределено, а какое свободно.

Для решения этих общих задач управления ресурсами разные ОС используют различные алгоритмы, что, в конечном счете, и определяет их облик в целом, включая характеристики производительности, область применения и даже пользовательский интерфейс. Так, например, алгоритм управления процессором в значительной степени определяет, является ли ОС системой разделения времени, системой пакетной обработки или системой реального времени.

***Операционная система как защитник пользователей и программ***

Если вычислительная система допускает совместную работу нескольких пользователей, то возникает проблема организации их безопасной деятельности. Необходимо обеспечить сохранность информации на диске, чтобы никто не мог удалить или повредить чужие файлы. Нельзя разрешить программам одних пользователей произвольно вмешиваться в работу программ других пользователей. Нужно пресекать попытки несанкционированного использования вычислительной системы. Всю эту деятельность осуществляет операционная система как организатор безопасной работы пользователей и их программ.

***Операционная система как постоянно функционирующее ядро***

Наконец, можно дать и такое определение: операционная система это программа, постоянно работающая на компьютере и взаимодействующая со всеми прикладными программами. Казалось бы, это абсолютно правильное определение, но во многих современных операционных системах постоянно работает на компьютере лишь часть операционной системы, которую принято называть ее ядром.

## **1.4. Краткая история эволюции вычислительных систем**

• **Первый период (1945 - 1955**). Ламповые машины. Операционные систем отсутствовали.

Программирование осуществлялось исключительно на машинном языке. Об операционных системах не было и речи, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. За пультом мог находиться только один пользователь. Программа загружалась в память машины в лучшем случае с колоды перфокарт, а обычно с помощью панели переключателей. Вычислительная система выполняла одновременно только одну операцию (ввод-вывод, собственно вычисления, размышления программиста). Отладка программ велась с пульта управления с помощью изучения состояния памяти и регистров машины. В конце этого периода появляется первое системное программное обеспечение: в 1951-52 гг. возникают прообразы первых компиляторов с символических языков (Fortran и др.), а в 1954 г. Nat Rochester разрабатывает ассемблер для IBM-701. В целом первый период характеризуется крайне высокой стоимостью вычислительных систем, их малым количеством и низкой эффективностью использования.

• **Второй период (1955 - Начало 60-х**). Компьютеры на основе транзисторов. Пакетные операционные системы.

Изменяется сам процесс прогона программ. Теперь пользователь приносит программу с входными данными в виде колоды перфокарт и указывает требуемые для нее ресурсы. Такая колода получает название задания. Оператор загружает задание в память машины и запускает его на исполнение. Полученные выходные данные печатаются на принтере, и пользователь получает их обратно через некоторое (довольно большое) время.

Смена запрошенных ресурсов вызывает приостановку выполнения программ. В результате процессор часто простаивает. Для повышения эффективности использования компьютера задания с похожими требуемыми ресурсами начинают собирать вместе, создавая пакет заданий.

Появляются первые системы пакетной обработки, которые просто автоматизируют запуск одной программы из пакета за другой и, тем самым, увеличивают коэффициент загрузки процессора. При реализации систем пакетной обработки был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какую работу он хочет выполнить на вычислительной машине. Системы пакетной обработки явились прообразом современных операционных систем, они стали первыми системными программами, предназначенными для управления вычислительным процессом.

• Третий период (Начало 60-х - 1980). Компьютеры на основе интегральных микросхем. Первые многозадачные ОС.

Операции ввода-вывода в режиме spooling (сокращение от Simultaneous Peripheral Operation On Line) или подкачки-откачки данных. Введение техники подкачки-откачки в пакетные системы позволило совместить реальные операции ввода-вывода одного задания с выполнением другого задания, но потребовало появления аппарата прерываний для извещения процессора об окончании этих операций.

Пакетные системы начинают заниматься планированием заданий: в зависимости от наличия запрошенных ресурсов, срочности вычислений и т.д. на счет выбирается то или иное задание.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью мультипрограммирования: пока одна программа выполняет операцию ввода-вывода, процессор не простаивает, как это происходило при однопрограммном режиме, а выполняет другую программу. Когда операция ввода-вывода заканчивается, процессор возвращается к выполнению первой программы. При этом каждая программа загружается в свой участок оперативной памяти, называемый разделом, и не должна влиять на выполнение другой программы.

• Четвертый период (1980 – настоящее время). Персональные компьютеры. Классические, сетевые и распределенные системы.

Появление мультипрограммирования требует целой революции в строении вычислительной системы. Большую роль, здесь играет аппаратная поддержка, наиболее существенные особенности которой:

## **1.5. Основные задачи ОС**

***Реализация защитных механизмов***. Программы не должны иметь самостоятельного доступа к распределению ресурсов, что приводит к появлению привилегированных и непривилегированных команд. Привилегированные команды, например, команды ввода-вывода, могут исполняться только операционной системой. Говорят, что она работает в привилегированном режиме. Переход управления от прикладной программы к ОС сопровождается контролируемой сменой режима. Во-вторых, это защита памяти, позволяющая изолировать конкурирующие пользовательские программы друг от друга, а ОС от программ пользователей.

Все современные процессоры поддерживают, как минимум, два режима: ***привилегированный*** режим (он же режим ядра, kernel mode) и ***непривилегированный*** (режим задачи, режим пользователя, user mode). Программы, работающие в режиме ядра, имеют полный, неограниченный доступ ко всем ресурсам компьютера: его командам, адресам, портам ввода/вывода и т.п. В режиме задачи возможности программы ограничены, она, в частности, не может выполнить некоторые специальные команды. Аппаратное разграничение возможностей является абсолютно необходимым условием реализации надежной защиты данных в многопользовательской системе. Отсюда вытекает и определение ядра как части ОС, работающей в режиме ядра. Все остальные программы, как системные утилиты, так и программы пользователей, работают в режиме пользователя и должны обращаться к ядру для выполнения многих системных действий.

***Обработка прерываний.*** Внешние прерывания оповещают ОС о том, что произошло асинхронное событие, например, завершилась операция ввода-вывода. Внутренние прерывания (сейчас их принято называть исключительными ситуациями) возникают, когда выполнение программы привело к ситуации, требующей вмешательства ОС, например, деление на ноль или попытка нарушения защиты.

***Организация мультипрограммирования***

Интерфейс между прикладной программой и ОС был организован при помощи набора системных вызовов.

Организация очереди из заданий в памяти и выделение процессора одному из заданий потребовали планирования заданий.

Для переключения процессора с одного задания на другое возникла потребность в сохранении содержимого регистров и структур данных, необходимых для выполнения задания, иначе говоря, контекста, для обеспечения правильного продолжения вычислений.

Поскольку память является ограниченным ресурсом, оказались нужны стратегии управления памятью, то есть потребовалось упорядочить процессы размещения, замещения и выборки информации из памяти.

Так как программы могут пожелать произвести санкционированный обмен данными, стало необходимо их обеспечить средствами коммуникации.

Для корректного обмена данными необходимо предусмотреть координацию программами своих действий, т.е. средства синхронизации.

***Основные функции классических операционных систем:***

• Планирование заданий и использования процессора.

• Обеспечение программ средствами коммуникации и синхронизации.

• Управление памятью.

• Управление файловой системой.

• Управление вводом-выводом.

• Обеспечение безопасности

Каждая из приведенных функций обычно реализована в виде подсистемы, являющейся структурным компонентом ОС. В каждой конкретной операционной системе эти функции, конечно, реализовывались по-своему, в различном объеме. Они не были придуманы как составные части деятельности операционных систем изначально, а появились в процессе развития, по мере того, как вычислительные системы становились удобнее, эффективнее и безопаснее.

Дополнительные функции:

* параллельное или псевдопараллельное выполнение задач (многозадачность).
* взаимодействие между процессами: обмен данными, взаимная синхронизация.
* защита самой системы, а также пользовательских данных и программ от злонамеренных действий пользователей или приложений.
* разграничение прав доступа и многопользовательский режим работы (аутентификация, авторизация).

## **1.6. Основные понятия, связанные с ОС**

• Системные вызовы

• Прерывания

• Исключительные ситуации

• Файлы

• Процессы, нити

В любой операционной системе поддерживается некоторый механизм, который позволяет пользовательским программам обращаться за услугами ядра ОС. Эти средства общения с ядром часто назывались экстракодами или системными макрокомандами. В ОС UNIX такие средства называются **системными вызовами.**

***Системные вызовы*** (system calls) интерфейс между операционной системой и пользовательской программой. Они создают, удаляют и используют различные объекты, главные из которых процессы и файлы. Пользовательская программа запрашивает сервис у операционной системы, осуществляя системный вызов. Имеются библиотеки процедур, которые загружают машинные регистры определенными параметрами и осуществляют прерывание процессора, после чего управление передается обработчику данного вызова, входящему в ядро операционной системы. Цель таких библиотек сделать системный вызов похожим на обычный вызов подпрограммы.

Основное отличие состоит в том, что при системном вызове задача переходит в привилегированный режим или режим ядра (kernel mode). Поэтому системные вызовы иногда еще называют программными прерываниями в отличие от аппаратных прерываний, которые чаще называют просто *прерываниями*.

В этом режиме работает код ядра операционной системы, причем он исполняется в адресном пространстве и в контексте вызвавшей его задачи. Таким образом, ядро операционной системы имеет полный доступ к памяти пользовательской программы, и при системном вызове достаточно передать адреса одной или нескольких областей памяти с параметрами вызова и адреса одной или нескольких областей памяти для результатов вызова.

В большинстве операционных систем системный вызов осуществляется командой программного прерывания (INT). Таким образом, программное прерывание это синхронное событие.

***Прерывания***

Прерывание (hardware interrupt) событие, генерируемое внешним (по отношению к процессору) устройством. Посредством аппаратных прерываний аппаратура либо информирует центральный процессор о том, что возникло какое-либо событие, требующее немедленной реакции (например, пользователь нажал клавишу), либо сообщает о завершении асинхронной операции ввода-вывода (например, закончено чтение данных с диска в основную память).

Важный тип аппаратных прерываний - прерывания таймера, которые генерируются периодически через фиксированный промежуток времени. Прерывания таймера используются операционной системой при планировании процессов. Каждый тип аппаратных прерываний имеет собственный номер, однозначно определяющий источник прерывания.

*Аппаратное прерывание* это асинхронное событие, то есть оно возникает вне зависимости от того, какой код исполняется процессором в данный момент. Обработка аппаратного прерывания не должна учитывать, какой процесс является текущим.

***Исключительные ситуации***

*Исключительная ситуация* (exception) событие, возникающее в результате попытки выполнения программой недопустимой команды, доступа к ресурсу при отсутствии достаточных привилегий или обращения к отсутствующей странице памяти.

Исключительные ситуации так же, как и системные вызовы, являются синхронными событиями, возникающими в контексте текущей задачи. Исключительные ситуации можно разделить на исправимые и неисправимые. К исправимым относятся такие исключительные ситуации, как отсутствие нужной информации в оперативной памяти. После устранения причины исправимой исключительной ситуации программа может продолжить выполнение.

Возникновение в процессе работы операционной системы исправимых исключительных ситуаций является нормальным явлением. Неисправимые исключительные ситуации обычно возникают в результате ошибок в программах. Обычно операционная система реагирует на такие ситуации завершением программы, вызвавшей исключительную ситуацию.

***Файлы***

Файлы предназначены для хранения информации на внешних носителях, то есть, принято, что информация, лежащая, например, на диске, должна находиться внутри файла. Обычно под файлом понимают часть пространства на носителе информации, имеющую имя.

Главная задача файловой системы (file system) скрыть особенности ввода-вывода и дать программисту простую абстрактную модель файлов, независимых от устройств. Для чтения, создания, удаления, записи, открытия и закрытия файлов также имеется обширная категория системных вызовов (create, delete, open, close, read, write ).

Пользователям хорошо знакомы такие понятия, связанные с организацией файловой системы, как каталог, текущий каталог, корневой каталог, путь, для манипулирования которыми в операционной системе имеются системные вызовы.

***Процессы и нити***

**Проце́сс** — программа, которая выполняется в текущий момент. Стандарт [ISO 9000:2000](https://ru.wikipedia.org/wiki/ISO_9001:2000) определяет процесс как совокупность взаимосвязанных и взаимодействующих действий, преобразующих входящие данные в исходящие.

Компьютерная программа сама по себе — лишь пассивная последовательность инструкций. В то время как процесс — непосредственное выполнение этих инструкций.

Также, процессом называют выполняющуюся программу и все её элементы: [адресное пространство](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE), [глобальные переменные](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F), [регистры](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_(%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)), [стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA), открытые [файлы](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) и так далее.

**Пото́к выполне́ния** (тред; от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *thread* — нить) — наименьшая единица обработки, исполнение которой может быть [назначено](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [ядром операционной системы](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B). Реализация потоков выполнения и [процессов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) в разных операционных системах отличается друг от друга, но в большинстве случаев поток выполнения находится внутри процесса. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как [память](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D0%BC%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C), тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени).

## **1.7. Основные подходы к организации архитектуры ОС:**

**Состав операционной системы:**

В общем случае в состав **ОС** входят следующие модули:

* Программный модуль, управляющий файловой системой.
* Командный процессор, выполняющий команды пользователя.
* Драйверы устройств.
* Программные модули, обеспечивающие графический пользовательский интерфейс.
* Сервисные программы.
* Справочная система.

**Драйвер устройства** (device driver) – специальная программа, обеспечивающая управление работой устройств и согласование информационного обмена с другими устройствами.  
  
  
  
**Командный процессор** (command processor) – специальная программа, которая запрашивает у пользователя команды и выполняет их (интерпретатор программ).

Интерпретатор  команд отвечает за загрузку приложений и управление информационным потоком между приложениями.

Для упрощения работы пользователя в состав современных ОС входят программные модули, обеспечивающие графический пользовательский интерфейс.  
Процесс работы компьютера в определенном смысле сводится к обмену файлами между устройствами. В ОС имеется программный модуль, управляющий файловой системой.

**Сервисные программы** позволяют обслуживать диски (проверять, сжимать, дефрагментировать и др.), выполнять операции с файлами (копирование, переименование и др.), работать в компьютерных сетях.

Для удобства пользователя в состав ОС входит **справочная система**, позволяющая оперативно получить необходимую информацию о функционировании как ОС в целом, так и о работе ее отдельных модулей.

*Состав модулей ОС, а также их количество зависит от семейства и вида ОС. Так, например, в ОС MS DOS отсутствует модуль, обеспечивающий графический пользовательский интерфейс.*

Наиболее общим подходом к структуризации **операционной системы** является разделение всех ее модулей на две группы:

1. **Ядро** – это модули, выполняющие основные функции ОС.
2. **Вспомогательные модули**, выполняющие вспомогательные функции ОС. Одним из определяющих свойств ядра является работа в **привилегированном** **режиме**.

Модули ядра выполняют следующие базовые функции ОС: Управление процессами, Управление системой прерываний, Управление памятью, управление устройствами ввода-вывода, Функции, решающие внутрисистемные задачи организации вычислительного процесса: переключение контекстов, загрузка/вы­грузка страниц, обработка прерываний. Эти функции недоступны для приложе­ний. Функции, служащие для поддержки приложений, создавая для них так называемую прикладную программную среду.

Приложения могут обращаться к ядру с запросами – **системными вызовами** – для выполнения тех или иных действий: для открытия и чтения файла, вывода графической информации на дисплей, получения системного времени и т.д. Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования – API (**Application programming interface)**.

**Вспомогательные модули** ОС выполняют вспомогательные функции ОС (полезные, но менее обязательные чем функции ядра).

**Примеры вспомогательных модулей:**

* Программа архивирования данных.
* Программа дефрагментации диска.
* Текстовый редактор.

Вспомогательные модули ОС оформляются либо в виде приложений, либо в виде библиотек процедур. Вспомогательные модули ОС подразделяются на следующие группы:

**утилиты** – программы, решающие задачи управления и сопровождения компьютерной системы: обслуживание дисков и файлов.

**системные обрабатывающие программы** – текстовые или графические редакторы, компиляторы, компоновщики, отладчики.

программы предоставления пользователю дополнительных услуг пользовательского интерфейса (калькулятор, игры).

библиотеки процедур различного назначения, упрощающие разработку при­ложений (библиотека математических функций, функций ввода-вывода).

Как и обычные приложения, для выполнения своих задач утилиты, обрабатывающие программы и библиотеки ОС, обращаются к функциям ядра посредством системных вызовов.  
Функции, выполняемые модулями ядра, являются наиболее часто используемыми функциями операционной системы, поэтому скорость их выполнения определяет производительность всей системы в целом. Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, то есть являются резидентными.

Вспомогательные модули обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются транзитными. Такая организация ОС экономит оперативную память компьютера.

Примечание

*Разделение операционной системы на ядро и вспомогательные модули обеспечивает легкую расширяемость ОС. Чтобы добавить новую высокоуровневую функцию, достаточно разработать новое приложение, и при этом не требуется модифицировать основные функции, образующие ядро системы.*

Объектами ядра ОС являются:

* Процессы
* Файлы.
* События.
* Потоки
* Семафоры – объекты, позволяющие войти в заданный участок кода не более чем n потокам.
* Мьютексы – одноместные семафоры, служащие в программировании для синхронизации одновременно выполняющихся потоков.
* Файлы, проецируемые в память.

***Подходы к организации ядра ОС:***

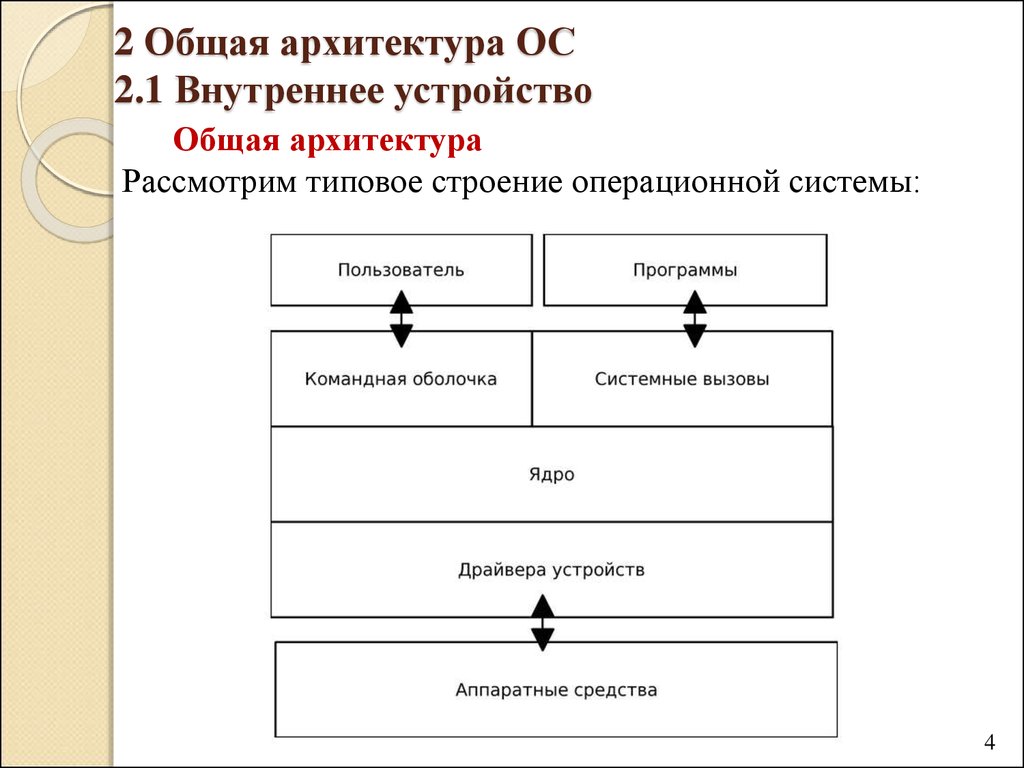
***Монолитное ядро***

По сути дела, операционная система это обычная программа, поэтому было бы логичным и организовать его так же, как устроено большинство программ, то есть составить из процедур и функций. В этом случае компоненты операционной системы являются не самостоятельными модулями, а составными частями одной большой программы. Такая структура операционной системы называется монолитным ядром (monolithic kernel) – см. рисунок 2.

**Рисунок 2 (а,б,в). Структура монолитного ядра операционной системы**



 А) Принцип монолитности



Б) Общая архитектура



В) Общая архитектура подробнее

Монолитное ядро представляет собой набор процедур, каждая из которых может вызвать каждую. Все процедуры работают в привилегированном режиме. Таким образом, монолитное ядро это такая схема операционной системы, при которой все ее компоненты являются составными частями одной программы, используют общие структуры данных и взаимодействуют друг с другом путем непосредственного вызова процедур. Для монолитной операционной системы ядро совпадает со всей системой.

Во многих операционных системах с монолитным ядром сборка ядра, то есть его компиляция, осуществляется отдельно для каждого компьютера, на который устанавливается операционная система. При этом можно выбрать список оборудования и программных протоколов, поддержка которых будет включена в ядро. Так как ядро является единой программой, перекомпиляция это единственный способ добавить в него новые компоненты или исключить неиспользуемые. Следует отметить, что присутствие в ядре лишних компонентов крайне нежелательно, так как ядро обычно полностью располагается в оперативной памяти. Кроме того, исключение ненужных компонент повышает надежность операционной системы в целом.

Монолитное ядро старейший способ организации операционных систем. Монолитное ядро включает:

Сервисные процедуры выполняются в привилегированном режиме, тогда как пользовательские программы в непривилегированном режиме. Для перехода с одного уровня привилегий на другой иногда может использоваться главная сервисная программа, определяющая, какой именно системный вызов был сделан, корректность входных данных для этого вызова, и передающая управление соответствующей сервисной процедуре с переходом в привилегированный режим работы.

Иногда выделяют также набор программных утилит, которые помогают выполнению сервисных процедур.

***Многослойные системы (Layered systems)***

Продолжая структуризацию, можно разбить всю вычислительную систему на ряд более мелких уровней с хорошо определенными связями между ними, так чтобы объекты уровня N могли вызывать только объекты из уровня N-1. Нижним уровнем в таких системах обычно является hardware, верхним уровнем интерфейс пользователя. Чем ниже уровень, тем более привилегированные команды и действия может выполнять модуль, находящийся на этом уровне. Впервые такой подход был применен при создании системы THE (Technishe Hogeschool Eindhoven) Дейкстрой и его студентами в 1968 г.

Многослойные системы хорошо реализуются. При использовании операций нижнего слоя не нужно знать, как они реализованы, нужно знать лишь, что они делают.

Многослойные системы хорошо тестируются. Отладка начинается с нижнего слоя и проводится послойно. При возникновении ошибки мы можем быть уверены, что она находится в тестируемом слое.

Многослойные системы хорошо модифицируются. При необходимости можно заменить лишь один слой, не трогая остальные. Но многослойные системы сложны для разработки: тяжело правильно определить порядок слоев, и что, к какому слою относится.

Ядро представляет собой сложный многофункциональный комплекс и многослойный подход обычно распространяется и на структуру ядра (рисунок 3).

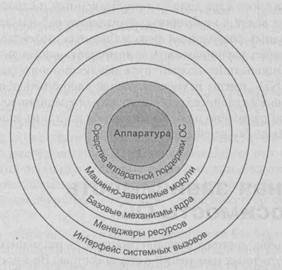


Рисунок 3 - Многослойная структура ядра ОС

Ядро может состоять из следующих слоев.

1. *Средства аппаратной поддержки ОС.* Часть функций ОС может выполняться аппаратными средствами. Иногда можно встретить определение операционной системы как совокупности программных и аппаратных средств. К операционной системе относят не все аппаратные устройства компьютера, а только средства аппаратной поддержки ОС, которые прямо участвуют в организации вычислительных процессов: средства поддержки привилегированного режима, систему прерываний, средства переключения контекстов процессов, средства защиты областей памяти.

2. *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, что позволяет разрабатывать вышележащие слои на основе машинно-независимых модулей, существующих в единственном экземпляре для всех типов аппаратных платформ, поддерживаемых данной ОС.

Linux, Mac OS, Windows NT – имеют четко определенный слой программных модулей (HAL, Hardware Abstraction Layer), экранирующих особенности аппаратуры.

3. *Базовые механизмы ядра.* Этот слой выполняет наиболее примитивные операции ядра (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, являются только исполнительными механизмами для модулей верхних слоев и выполняют их решения.

4. *Менеджеры ресурсов.* Этот слой состоит из функциональных модулей, реализующих стратегические задачи по управлению основными ресурсами ВС. Обычно на данном слое работают менеджеры (диспетчеры) процессов, ввода-вывода, файловой системы и оперативной памяти. Каждый из менеджеров ведет учет свободных и используемых ресурсов определенного типа и планирует их распределение в соответствии с запросами приложений. Например, менеджер виртуальной памяти управляет перемещением страниц из оперативной памяти на диск и обратно. Менеджер должен отслеживать интенсивность обращений к страницам, время пребывания их в памяти, и многие другие параметры, на основании которых он время от времени принимает решения о том, какие страницы необходимо выгрузить и какие - загрузить. Для исполнения принятых решений менеджер обращается к слою базовых механизмов с запросами. Внутри слоя менеджеров существуют тесные взаимные связи, так как 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.

5. *Интерфейс системных вызовов.* Это самый верхний слой ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы (read(fd, buffer, count)). Системные вызовы обычно обращаются за помощью к функциям слоя менеджеров ресурсов, причем для выполнения одного системного вызова может понадобиться несколько таких обращений.

Приведенное разбиение ядра ОС на слои является достаточно условным. В реальной системе количество слоев и распределение функций между ними может быть и иным. Слой машинно-зависимых модулей может сливаться со слоем базовых механизмов и частично со слоем менеджеров ресурсов. Базовые механизмы могут не выделяться в отдельный слой и эти функции выполняют менеджеры ресурсов.

И наоборот, ядро может состоять из большего количества слоев. Менеджеры ресурсов могут обладать многослойной структурой.

Способ взаимодействия слоев в реальной ОС также может отклоняться от этой схемы. Для ускорения работы ядра в некоторых случаях происходит непосредственное обращение с верхнего слоя к функциям нижних слоев, минуя промежуточные. Сами функции системных вызовов также могут нарушать иерархию слоев и обращаться прямо к базовым механизмам ядра.

Выбор количества слоев ядра - ответственная и сложная задача: увеличение числа слоев ведет к замедлению работы ядра за счет дополнительных накладных расходов на межслойное взаимодействие, а уменьшение числа слоев ухудшает расширяемость и логичность системы. Обычно ОС, прошедшие долгий путь развития (ранние версии Unix) имеют неупорядоченной ядро с небольшим числом четко выделенных слоев, у ОС развивающихся недавно ядро структурировано в большей степени.

***Микроядерная архитектура***

Современная тенденция в разработке операционных систем это перенесение значительной части системного кода на уровень пользователя и одновременной минимизации ядра. Речь идет о подходе к построению ядра, называемом микроядерной архитектурой (microkernel architecture) операционной системы, когда большинство ее составляющих являются самостоятельными программами.

В этом случае взаимодействие между ними обеспечивает специальный модуль ядра, называемый микроядром. Микроядро работает в привилегированном режиме и обеспечивает взаимодействие между программами, планирование использования процессора, первичную обработку прерываний, операции ввода-вывода и базовое управление памятью (рис. 4).

Остальные компоненты системы взаимодействуют друг с другом путем передачи сообщений через микроядро.

Основное достоинство микроядерной архитектуры высокая степень модульности ядра операционной системы. Это существенно упрощает добавление в него новых компонент. В микроядерной операционной системе можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д.

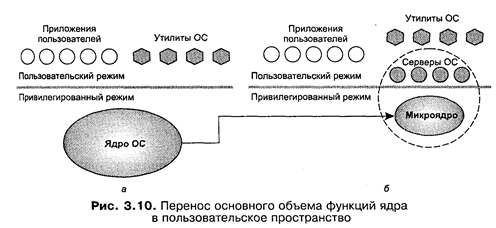
Существенно упрощается процесс отладки компонент ядра, так как новая версия драйвера может загружаться без перезапуска всей операционной системы. Компоненты ядра операционной системы ничем принципиально не отличаются от пользовательских программ, поэтому для их отладки можно применять обычные средства. В то же время, микроядерная архитектура операционной системы вносит дополнительные накладные расходы, связанные с передачей сообщений, что существенно влияет на производительность.

Для того чтобы микроядерная операционная система по скорости не уступала операционным системам на базе монолитного ядра, требуется очень аккуратно проектировать разбиение системы на компоненты, стараясь минимизировать взаимодействие между ними. Таким образом, основная сложность при создании микроядерных операционных систем необходимость очень аккуратного проектирования.



*Рис. 4 –Микроядро в привилегированном режиме*

*В состав микроядра обычно входят****машинно-зависимые модули****, а также модули, выполняющие базовые (но не все!) функции ядра по управлению процессами, обработке прерываний, управлению виртуальной памятью, пересылке сообщений и управлению устройствами ввода-вывода, связанные с загрузкой или чтением регистров устройств. Набор функций микроядра обычно соответствует функциям слоя базовых механизмов обычного ядра. Такие функции операционной системы трудно, если не невозможно, выполнить в пространстве пользователя.*

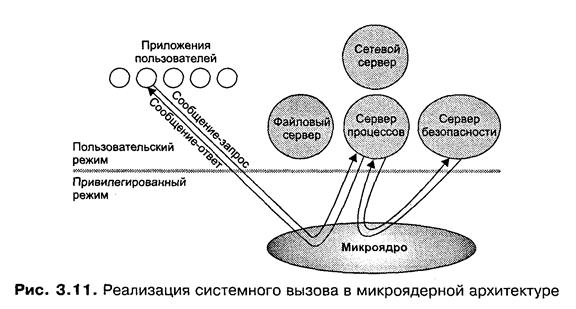
**

*Рис. 5 - Перенос функций ядра в пользовательское пространство*

*Все остальные более высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме. Однозначного решения о том, какие из системных функций нужно оставить в привилегированном режиме, а какие перенести в пользовательский, не существует. В общем случае многие менеджеры ресурсов, являющиеся неотъемлемыми частями обычного ядра — файловая система, подсистемы управления виртуальной памятью и процессами, менеджер безопасности и т. п., — становятся «периферийными» модулями, работающими в пользовательском режиме.*

*Работающие в пользовательском режиме менеджеры ресурсов имеют принципиальные отличия от традиционных утилит и обрабатывающих программ операционной системы. Они обслуживают запросы локальных приложений, и поэтому называются серверами ОС. Очевидно, что для реализации микроядерной архитектуры необходимым условием является наличие в операционной системе эффективного способа вызова процедур одного процесса из другого. Поддержка такого механизма и является одной из главных задач микроядра.*

*Схематично механизм обращения к функциям ОС, оформленным в виде серверов, выглядит следующим образом (рис. 6).****Клиент****, которым может быть либо прикладная программа, либо другой компонент ОС, запрашивает выполнение некоторой функции у соответствующего сервера, посылая ему сообщение. Непосредственная передача сообщений между приложениями невозможна, так как их адресные пространства изолированы друг от друга. Микроядро, выполняющееся в привилегированном режиме, имеет доступ к адресным пространствам каждого из этих приложений и поэтому может работать в качестве посредника. Микроядро сначала передает сообщение, содержащее имя и параметры вызываемой процедуры нужному серверу, затем сервер выполняет запрошенную операцию, после чего ядро возвращает результаты клиенту с помощью другого сообщения. Таким образом,****работа микроядерной операционной системы соответствует известной модели клиент-сервер, в которой роль транспортных средств выполняет микроядро.***

**

*Рис. 6 - Реализация системного вызова*

***Производительность****. При классической организации ОС (рис. 7, а) выполнение системного вызова сопровождается двумя переключениями режимов, а при микроядерной организации (рис. 7, б) — четырьмя. Таким образом, операционная система на основе микроядра при прочих равных условиях всегда будет менее производительной, чем ОС с классическим ядром. Именно по этой причине микроядерный подход не получил такого широкого распространения, которое ему предрекали.*

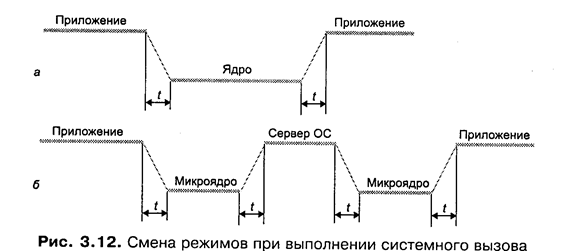
**

Рис. 7 - Смена режимов при выполнении системного вызова

***Смешанные системы***

Все рассмотренные подходы к построению операционных систем имеют свои преимущества и недостатки. В большинстве случаев современные операционные системы используют различные комбинации этих подходов.

Примером смешанного подхода может служить возможность запуска операционной системы с монолитным ядром под управлением микроядра. Так устроены 4.4BSD и MkLinux, основанные на микроядре Mach. Микроядро обеспечивает управление виртуальной памятью и работу низкоуровневых драйверов. Все остальные функции, в том числе взаимодействие с прикладными программами, осуществляется монолитным ядром. Данный подход возник в результате попыток использовать преимущества микроядерной архитектуры, сохраняя по возможности хорошо отлаженный код монолитного ядра.

Наиболее тесно элементы микроядерной архитектуры и элементы монолитного ядра переплетены в ядре Windows NT. Хотя Windows NT часто называют микроядерной операционной системой, это не совсем так. Микроядро NT слишком сложно и велико (более 1 Мб), чтобы носить приставку микро. Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах. В тоже время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

Таким образом, Windows NT можно с полным правом назвать гибридной операционной системой – рис. 8.

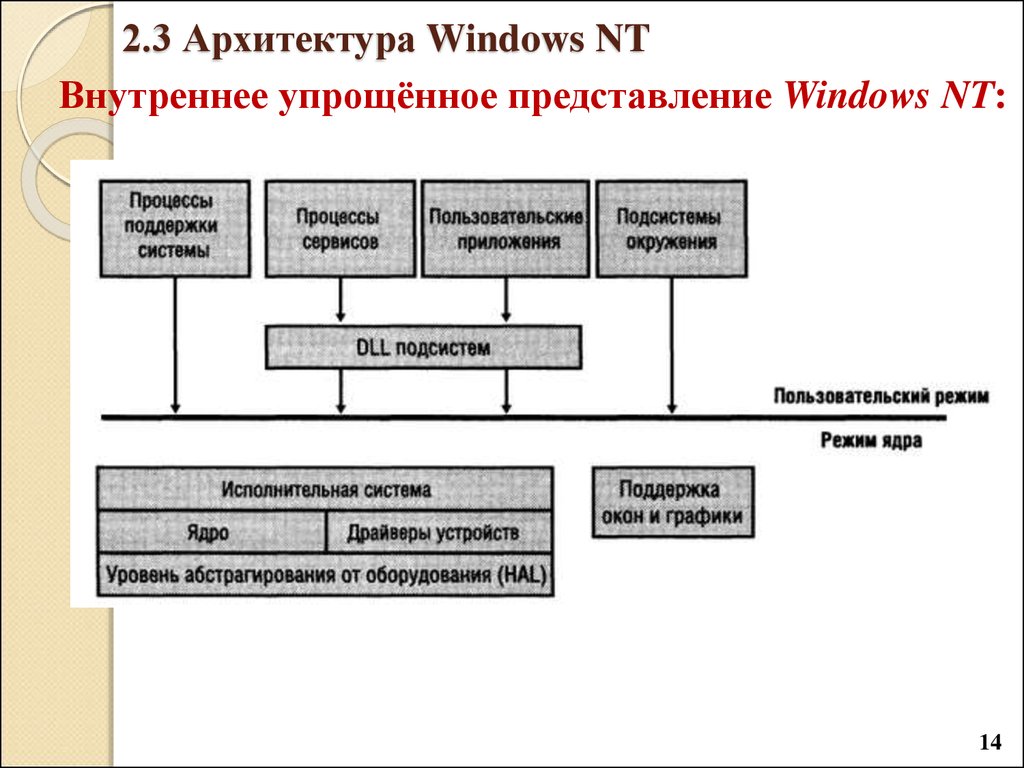


Рис.8 – Упрощенная структура Windows NT

***Прочие архитектурные подходы к организации ОС:***

***Построение ОС на базе объектно-ориентированного подхода*** дает возможность использовать все его достоинства, хорошо зарекомендовавшие себя на уровне приложений, внутри операционной системы, а именно: аккумуляцию удачных решений в форме стандартных объектов, возможность создания новых объектов на базе имеющихся с помощью механизма наследования, хорошую защиту данных за счет их инкапсуляции во внутренние структуры объекта, что делает данные недоступными для несанкционированного использования извне, структуризованность системы, состоящей из набора хорошо определенных объектов.

***Наличие нескольких прикладных сред*** дает возможность в рамках одной ОС одновременно выполнять приложения, разработанные для нескольких ОС. Многие современные операционные системы поддерживают одновременно прикладные среды MS-DOS, Windows, UNIX (POSIX), OS/2 или хотя бы некоторого подмножества из этого популярного набора. Концепция множественных прикладных сред наиболее просто реализуется в ОС на базе микроядра, над которым работают различные серверы, часть которых реализуют прикладную среду той или иной операционной системы.

***Распределенная организация операционной системы*** позволяет упростить работу пользователей и программистов в сетевых средах. В распределенной ОС реализованы механизмы, которые дают возможность пользователю представлять и воспринимать сеть в виде традиционного однопроцессорного компьютера. Характерными признаками распределенной организации ОС являются: наличие единой справочной службы разделяемых ресурсов, единой службы времени, использование механизма вызова удаленных процедур (RPC) для прозрачного распределения программных процедур по машинам, многонитевой обработки, позволяющей распараллеливать вычисления в рамках одной задачи и выполнять эту задачу сразу на нескольких компьютерах сети, а также наличие других распределенных служб.

**Гипервизор -** программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких немодифицированных ОС на одном и том же компьютере. Гипервизор обеспечивает изоляцию ОС друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.

Гипервизор также может (но не обязан) предоставлять работающим под его управлением на одном компьютере ОС средства связи и взаимодействия между собой (например, через обмен файлами или сетевые соединения) так, как если бы эти ОС выполнялись на разных физических компьютерах.

Гипервизор сам по себе в некотором роде является минимальной операционной системой (микроядром или наноядром). Он предоставляет запущенным под его управлением ОС сервис виртуальной машины, виртуализируя или эмулируя реальное (физическое) аппаратное обеспечение конкретной машины, и управляет этими виртуальными машинами, выделением и освобождением ресурсов для них.

*Виртуальная машина – программная или аппаратная среда, выполняющая некоторый код. Обычно виртуальная машина эмулирует работу реального компьютера. На компьютере может функционировать несколько виртуальных машин.*

Гипервизор позволяет независимое «включение», перезагрузку, «выключение» любой из виртуальных машин с той или иной ОС. При этом ОС, работающая в виртуальной машине под управлением гипервизора не знает, что она выполняется в виртуальной машине, а не на реальном аппаратном обеспечении.

## **1.8. Классификация ОС**

Существует несколько схем классификации операционных систем. Возможные критерии:

• Реализация многозадачности

• Поддержка многопользовательского режима.

• Многопроцессорная обработка

• Реализация режима реального времени

***По числу одновременно выполняемых задач*** операционные системы могут быть разделены на два класса:

• многозадачные (Unix, OS/2, Windows).

• однозадачные (например, MS-DOS) и

Многозадачная ОС, решая проблемы распределения ресурсов и конкуренции, полностью реализует мультипрограммный режим.

***Поддержка многопользовательского режима***

По числу одновременно работающих пользователей ОС можно разделить на:

• однопользовательские (MS-DOS, Windows 3.x);

• многопользовательские (Windows NT, Unix).

Наиболее существенно отличие заключается в наличии у многопользовательских систем механизмов защиты персональных данных каждого пользователя.

***Многопроцессорная обработка***

Многопроцессорные системы состоят из двух или более центральных процессоров, осуществляющих параллельное выполнение команд. Поддержка мультипроцессирования является важным свойством ОС и приводит к усложнению всех алгоритмов управления ресурсами. Многопроцессорная обработка реализована в таких ОС, как Linux, Solaris, Windows NT и в ряде других.

Многопроцессорные ОС разделяют на симметричные и асимметричные. В симметричных ОС на каждом процессоре функционирует одно и то же ядро и задача может быть выполнена на любом процессоре, то есть обработка полностью децентрализована.

В асимметричных ОС процессоры неравноправны. Обычно существует главный процессор (master) и подчиненные (slave), загрузку и характер работы которых определяет главный процессор.

***Системы реального времени***

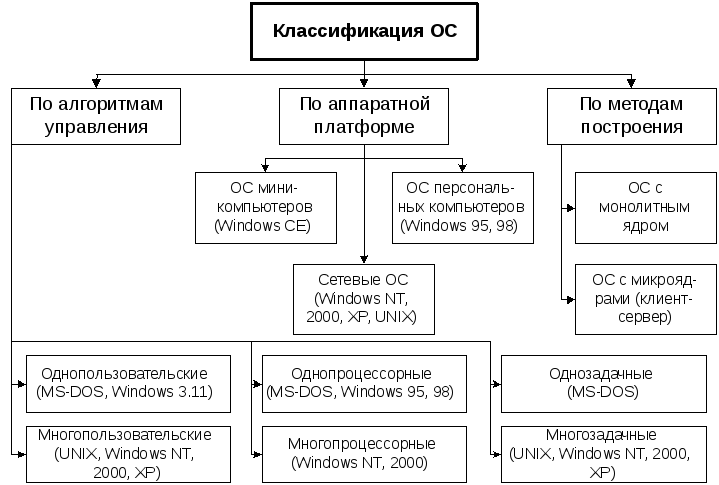
В разряд многозадачных ОС, наряду с пакетными системами и системами разделения времени, включаются также системы реального времени, не упоминавшиеся до сих пор.

Они используются для управления различными техническими объектами или технологическими процессами. Такие системы характеризуются предельно допустимым временем реакции на внешнее событие, в течение которого должна быть выполнена программа, управляющая объектом. Система должна обрабатывать поступающие данные быстрее, чем те могут поступать, причем от нескольких источников одновременно.

Столь жесткие ограничения сказываются на архитектуре систем реального времени, например, в них может отсутствовать виртуальная память, поддержка которой дает непредсказуемые задержки в выполнении программ.

Приведенная классификация ОС не является исчерпывающей.

Можно рассмотреть некоторые другие классификации, например, см. рис. 9.



## **1.8. Роли пользователей ОС**

Возможные роли пользователей ОС приведены в таблице 1.



## **ЛИТЕРАТУРА:**

|  |
| --- |
| **Основной учебник:**   1. **Современные операционные системы. Э. Таненбаум. 4-е изд. - СПб.: 2015. — 1120 с.** 2. Предыдущие издания книги 1 (основные сведения по ОС аналогичны) |
| **Прочая литература:** |
| 1. Батаев А., Налютин Н., Синицын С. Операционные системы и среды. Учебник. −М: Академия 2014 −272 с.:ил. |
| 1. **Малахов и др. Программирование приложений с использованием Windows API (методическое пособие). Филиал федерального Государственного бюджетного образовательного учреждения высшего профессионального образования «Национальный исследовательский университет «МЭИ» в г. Смоленске, 2011 г.** |
| 1. Мартемьянов Ю.Ф., Яковлев Ал.В., Яковлев Ан.В. Операционные системы. Концепция построения и обеспечения безопасности. Учебное пособие для вузов. −М: Грячая линия −Телеком, 2011. −332 с.:ил. |
| 1. В.Г. Олифер, Н.А. Олифер Сетевые операционные системы. - СПб.: Издательство "Питер", 2010. - 544 с. ил. |
| 1. Партыка Т.Л., Попов И.И. Операционные системы, среды и оболочки: учебное пособие. −М: ФОРУМ, 2013. −544с.:ил. |
| 1. Безруков, В.А. WIN32 API программирование [Электронный ресурс] : . — Электрон. дан. — Спб. : НИУ ИТМО (Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики), 2009. — 96 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=40770 — Загл. с экрана. |
| 1. Блинков, Ю.В. Операционные системы, среды и оболочки. Часть 1. Операционные системы: учебное пособие [Электронный ресурс] : . — Электрон. дан. — Пенза : ПензГТУ (Пензенский государственный технологический университет), 2011. — 218 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=62776 — Загл. с экрана. |
| 1. Борисов, Е.В. Разработка операционной системы и компилятора. Проект Оберон. [Электронный ресурс] : / Е.В. Борисов, Л.Н. Чернышов. — Электрон. дан. — М. : ДМК Пресс, 2012. — 559 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=39992 — Загл. с экрана. |
| 1. Волосатова, Т.М. Основные концепции операционной системы UNIX [Электронный ресурс] : учебное пособие / Т.М. Волосатова, С.В. Грошев, С.В. Родионов. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана (Московский государственный технический университет имени Н.Э. Баумана), 2010. — 96 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=52399 — Загл. с экрана. |
| 1. Мартемьянов, Ю.Ф. Операционные системы. Концепции построения и обеспечения безопасности [Электронный ресурс] : учебное пособие / Ю.Ф. Мартемьянов, Яковлев Ал.В., Яковлев Ан.В. — Электрон. дан. — М. : Горячая линия-Телеком, 2011. — 332 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=5176 — Загл. с экрана. |
| 1. Хэвиленд К. Системное программирование в UNIX [Электронный ресурс] : / Хэвиленд К., Грей Д., Салама Б. — Электрон. дан. — М. : ДМК Пресс, 2007. — 368 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=1223 — Загл. с экрана. |

Дополнительно:

Классификацию операционных систем можно осуществлять несколькими способами.

1. По способу организации вычислений:
   * системы пакетной обработки (batch processing operating systems) – целью является выполнение максимального количества вычислительных задач за единицу времени; при этом из нескольких задач формируется пакет, который обрабатывается системой;
   * системы разделения времени (time-sharing operating systems) – целью является возможность одновременного использования одного компьютера несколькими пользователями; реализуется посредством поочередного предоставления каждому пользователю интервала процессорного времени;
   * системы реального времени (real-time operating systems) – целью является выполнение каждой задачи за строго определённый для данной задачи интервал времени.
2. По типу ядра:
   * системы с монолитным ядром (monolithic operating systems);
   * системы с микроядром (microkernel operating systems);
   * системы с гибридным ядром (hybrid operating systems).
3. По количеству одновременно решаемых задач:
   * однозадачные (single-tasking operating systems);
   * многозадачные (multitasking operating systems).
4. По количеству одновременно работающих пользователей:
   * однопользовательские (single-user operating systems);
   * многопользовательские (multi-user operating systems).
5. По количеству поддерживаемых процессоров:
   * однопроцессорные (uniprocessor operating systems);
   * многопроцессорные (multiprocessor operating systems).
6. По поддержке сети:
   * локальные (local operating systems) – автономные системы, не предназначенные для работы в компьютерной сети;
   * сетевые (network operating systems) – системы, имеющие компоненты, позволяющие работать с компьютерными сетями.
7. По роли в сетевом взаимодействии:
   * серверные (server operating systems) – операционные системы, предоставляющие доступ к ресурсам сети и управляющие сетевой инфраструктурой;
   * клиентские (client operating systems) – операционные системы, которые могут получать доступ к ресурсам сети.
8. По типу лицензии:
   * открытые (open-source operating systems) – операционные системы с открытым исходным кодом, доступным для изучения и изменения;
   * проприетарные (proprietary operating systems) – операционные системы, которые имеют конкретного правообладателя; обычно поставляются с закрытым исходным кодом.
9. По области применения:
   * операционные системы мэйнфреймов – больших компьютеров (mainframe operating systems);
   * операционные системы серверов (server operating systems);
   * операционные системы персональных компьютеров (personal computer operating systems);
   * операционные системы мобильных устройств (mobile operating systems);
   * встроенные операционные системы (embedded operating systems);
   * операционные системы маршрутизаторов (router operating systems).

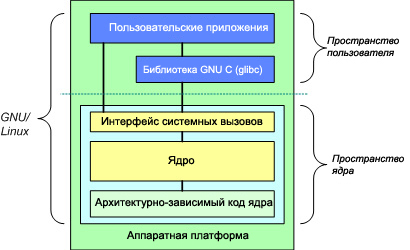
**Требования к операционным системам**

Основное требование, предъявляемое к современным операционным системам – выполнение функций, перечисленных выше в параграфе "Функции операционных систем". Кроме этого очевидного требования существуют другие, часто не менее важные [[3](http://intuit.valrkl.ru/course-1143/Invalid_Ref)]:

* расширяемость – возможность приобретения системой новых функций в процессе эволюции; часто реализуется за счет добавления новых модулей;
* переносимость – возможность переноса операционной системы на другую аппаратную платформу с минимальными изменениями;
* совместимость – способность совместной работы; может иметь место совместимость новой версии операционной системы с приложениями, написанными для старой версии, или совместимость разных операционных систем в том смысле, что приложения для одной из этих систем можно запускать на другой и наоборот;
* надежность – вероятность безотказной работы системы;
* производительность – способность обеспечивать приемлемые время решения задач и время реакции системы.

**Архитектуры ОС:**

Начнем с относительно простого ядра Linux.

[](https://cs8.pikabu.ru/post_img/2016/02/28/6/1456653366150591206.jpg)

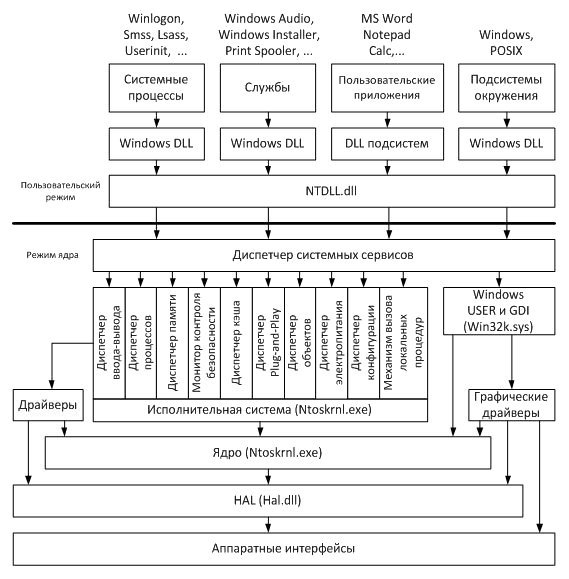
Обсуждать тут в принципе нечего - любая программа через системные вызовы обращается к ядру ОС через интерфейс системных вызовов. Ядро через архитектурно зависимый код транслирует команды аппаратному обеспечению. Все это очень просто и быстро, единственное что заслуживает внимание - то что программы "видят" себя в едином виртуальном окружении независимо от архитектуры системы. Структура и системные вызовы стандартизированы и едины для любой аппаратной платформы. Графичесая подсистема является одной из внешних программ, поэтому сменить графическую оболочку можно без перезагрузки ОС.

Теперь перейдем к Windows:

Как видим структура ядра намного более сложная. Преимущество это или недостаток? Каждый решает для себя сам. Программа под Windows обращается через документированный Windows API к "своей" библиотеке (например Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll), эти библиотеки по внутреннему протоколу (документация для разработчиков не из Microsoft не доступна) обращается по протоколу Native API к Ntdll.dll и далее передается через диспетчер системных сервисов ядру (все это внутри Ntoskrnl.exe).

После обработки в режиме ядра и использованию драйверов данные передаются Hal.dll - библиотеке которая непосредственно управляет оборудованием. Второй системой управляющей оборудованием является графическое ядро (проприетарные функции видеокарт в основном касаются именно этого).

Важно отметить одну особенность пользовательского режима ядра Windows - "Подсистемы окружения". Эта компонента позволяет Windows использовать коды стандартов POSIX, Win16 и т.п. Данный механизм по сути является набором виртуальных ядер сторонних ОС и позволяет быстро адаптировать под Windows любой сторонний код.

[](https://cs8.pikabu.ru/post_img/2016/02/28/6/1456653478161419729.jpg)

Не менее важным является включение графической подсистемы в ядро (в современных серверных ОС данная система отключена или отсутствует), что во-первых не дает сменить графическую оболочку "на ходу", а во-вторых уже не способствует быстродействию...

На анализе архитектур можно сделать вывод, что ядро ОС Linux принципиально обеспечивает наилучшее быстродействие и надежность за счет своей монолитности, а ядро Windows позволяет быстро адаптировать любой стороний код для совместимости с операционной системой, однако такая адаптация имеет свою цену - систему приходится перезагружать по любому мало-мальскому поводу.