Структуры ОС: монолитные системы, многоуровневые системы, виртуальные машины, экзо-ядро и модель “клиент”

ПОДГОТОВИЛ СОЛДАТЕНКО А.А.
ГР. ТЭ621

План

- Структура ОС
- монолитные системы
- многоуровневые системы
- виртуальные машины
- экзо-ядро
- модель “клиент”

Структура ОС

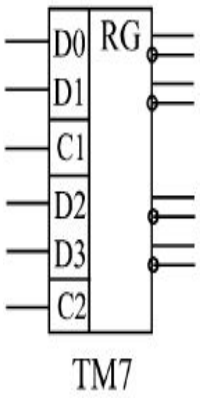
- С добавлением в операционные системы все новых функций, а также с ростом возможностей управляемого операционными системами аппаратного обеспечения и его разнообразия возрастает степень их сложности. Операционная система CTSS, введенная в эксплуатацию в Массачусетском технологическом институте в 1963 году, занимала в памяти около 32000 36-битовых слов. Операционная система OS/360, выпущенная фирмой IBM через год, содержала более миллиона машинных команд. Система Multics, совместная разработка которой была завершена специалистами Массачусетского технологического института и компанией Bell Laboratories к 1975 году, разрослась до 20 миллионов команд. Ради справедливости отметим, что впоследствии на меньших машинах стали появляться операционные системы и попроще, но и они неуклонно усложнялись с развитием аппаратного обеспечения и ростом требований со стороны пользователей. Так, современная система UNIX по своей сложности намного превосходит свой почти игрушечный первоначальный вариант, разработанный несколькими талантливыми программистами в начале 70-х годов. То же самое произошло с простой системой MS-DOS, со временем переросшей в сложные и мощные операционные системы OS/2 и Windows 2000. Так, операционная система Windows NT содержит около 16 миллионов строк кода, а в Windows 2000 этот показатель увеличен более чем в два раза.

- Увеличение размера полнофункциональных операционных систем и сложности выполняемых ими задач стало причиной возникновения трех широко распространенных проблем. Во-первых, операционные системы доходят до пользователей с хроническим опозданием. Это касается как выпуска новых операционных систем, так и обновления уже существующих. Во-вторых, в системах появляются скрытые ошибки, которые начинают проявлять себя в рабочих условиях и требуют исправления и доработки системы. В-третьих, рост производительности зачастую происходит не так быстро, как планируется. Как же следует организовать структуру операционных систем, чтобы упростить работу с ними и преодолеть перечисленные проблемы? Некоторые решения очевидны. Программное обеспечение должно состоять из модулей, что упростит организацию процесса его разработки и облегчит выявление и устранение ошибок. Модули по отношению друг к другу должны иметь тщательно разработанные и максимально простые интерфейсы, что также облегчит задачи программиста. Кроме того, меньше усилий потребует эволюция такой системы. Если взаимодействие модулей друг с другом происходит по простым и четким правилам, изменение любого модуля окажет минимальное влияние на остальные.

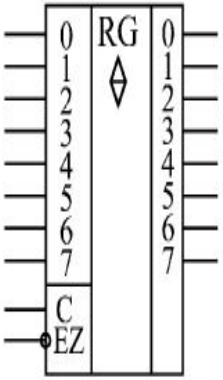
- Однако оказалось, что для больших операционных систем, код которых состоит из миллионов или десятков миллионов строк, принцип модульного программирования сам по себе не избавляет от всех проблем. По этой причине возросла популярность концепции уровней иерархии, а также информационной абстракции. В иерархической структуре современной операционной системы различные функции находятся на разных уровнях в зависимости от их сложности, временных характеристик и степени абстракции. Систему можно рассматривать как набор уровней, каждый из которых выполняет свой ограниченный круг заданий, входящий в комплекс задач операционной системы. Работа компонентов определенного уровня основывается на работе компонентов, находящихся на более низком уровне; функции более высокого уровня используют примитивы нижнего по отношению к нему уровня. В идеале уровни должны быть определены так, чтобы при изменении одного из них не изменялись остальные.

- Как правило, чем ниже уровень, тем меньше время работы его компонентов. Некоторые элементы операционной системы должны непосредственно взаимодействовать с аппаратным обеспечением компьютера, элементарные процессы в котором иногда длятся не более нескольких миллионных долей секунды. Составляющие операционной системы, поддерживающие взаимосвязь с пользователем, находятся на другом конце временного диапазона. Пользователи вводят команды весьма медленно — до одной команды за несколько секунд.
- В каждой отдельно взятой операционной системе перечисленные принципы применяются по-разному. Для получения общего представления об операционных системах на данном этапе изложения представим пример обобщенной модели иерархической операционной системы, описанной в [BROW84] и [DENN84]. Она, несомненно, полезна для понимания сути дела, хотя и не соответствует ни одной реальной операционной системе.

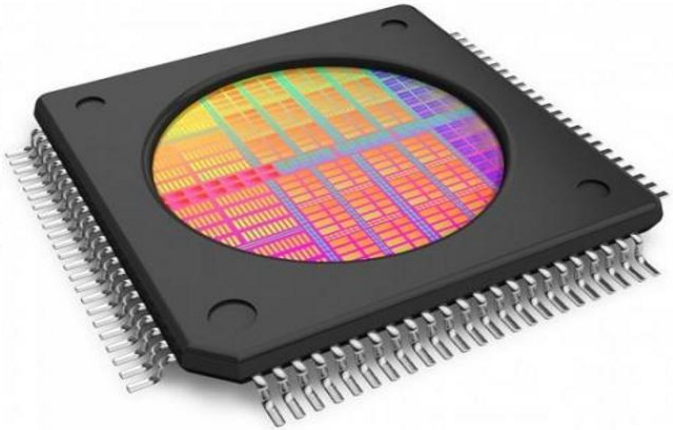
- **Уровень 1**. В него входят электронные схемы; объектами данного уровня являются регистры, ячейки памяти и логические элементы. Над этими объектами выполняются различные действия, такие, как очистка содержимого регистра или считывание ячейки памяти.

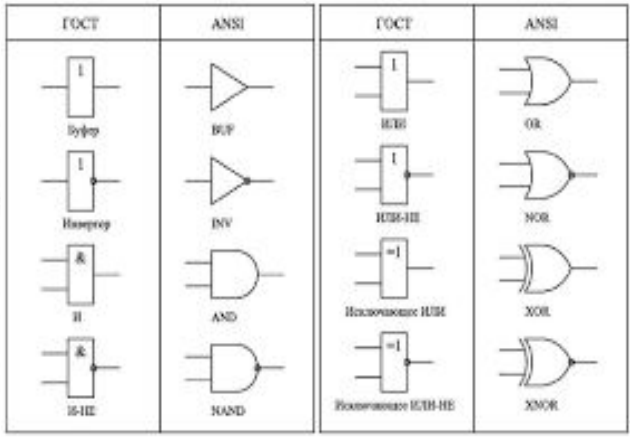


ТМ7



ИР22





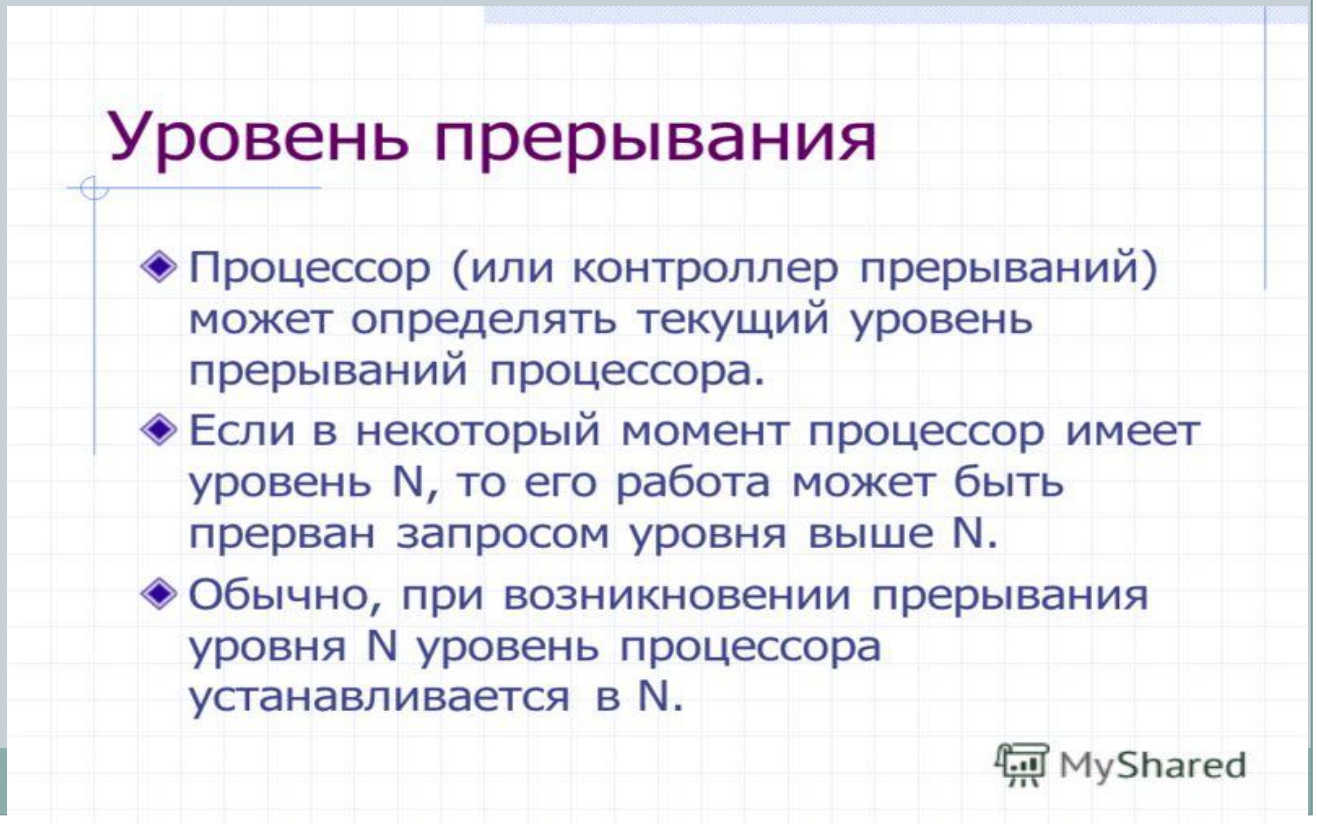
- **Уровень 2.** Набор команд процессора. В число операций, выполняемых на этом уровне, входят те, которые допускаются набором команд машинного языка, например сложение, вычитание, загрузка значения из регистра или сохранение в нем.

Команда	Доля (частота) использования при целочисленных вычислениях, %
1. Загрузка	22
2. Условные переходы	20
3. Сравнение	16
4. Запоминание	12
5. Сложение	8
6. Логическое И	6
7. Вычитание	5
8. Пересылки регистр-регистр	4
9. Вызов подпрограмм	1
10. Возврат из подпрограмм	1
Всего	96

- **Уровень 3.** Содержит концепцию процедуры (подпрограммы), а также операции вызова и возврата.

```
' Вызываемая процедура
Sub DisplayName(sName As String)
MsgBox "Hello, " & sName
End Sub
' Вызывающая процедура
Sub name()
DisplayName "Conrad" '1-ый способ вызова процедуры
'Call DisplayName("Conrad") 2-й способ вызова процедуры
End Sub
```

- **Уровень 4.** Уровень прерываний, которые заставляют процессор сохранить текущий контекст и выполнить подпрограмму обработки прерывания.



На самом деле первые четыре уровня не являются частями операционной системы, они составляют аппаратное обеспечение процессора. Однако на этих уровнях уже появляются некоторые элементы операционной системы, такие, как программы обработки прерываний. Вплотную к операционной системе мы подходим только на пятом уровне, на котором возникают концепции, связанные с многозадачностью.

- **Уровень 5.** На этом уровне вводится понятие процесса, под которым подразумевается работающая программа. В число фундаментальных требований к операционной системе, способной поддерживать одновременную работу не скольких процессов, входят способность приостанавливать процессы и возобновлять их выполнение. Для этого необходимо сохранять содержимое регистров аппаратного обеспечения, чтобы можно было переключаться с одного процесса на другой. Кроме того, если процессы должны взаимодействовать между собой, необходим механизм их синхронизации. Одной из важнейших концепций устройства операционных систем является семафор — простейший способ передачи сигналов.

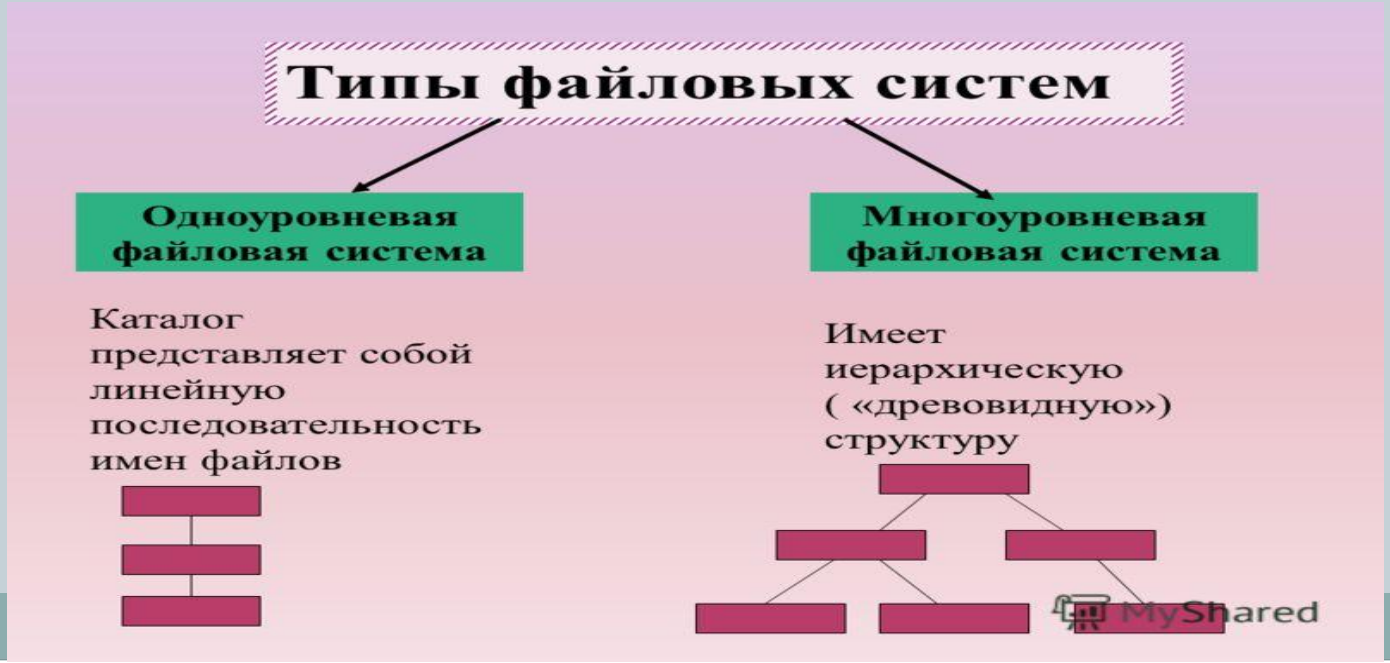
Уровень 6. Компоненты этого уровня взаимодействуют со вспомогательными запоминающими устройствами компьютера. На этом уровне происходит позиционирование считывающих головок и физическая передача блоков данных. Для планирования работы и уведомления процесса о завершении запрошенной операции уровень 6 использует компоненты уровня 5.

- **Уровень 7.** Создает логическое адресное пространство процессов. Уровень организует виртуальное адресное пространство в виде блоков, которые могут перемещаться между основной памятью и вспомогательным запоминающим устройством. Широко распространены следующие три схемы: использование страниц фиксированного размера, использование сегментов переменного размера и комбинация тех и других. Если нужный блок отсутствует в основной памяти, то данный уровень передает уровню 6 запрос о передаче этого блока.

- До сих пор речь шла только о взаимодействии операционной системы с процессором. Компоненты операционной системы, относящиеся к восьмому и более высоким уровням, вступают во взаимодействие с внешними объектами, такими, как периферийные устройства, а возможно — с сетью и компьютерами, подключенными к сети. Объектами этих уровней являются логические именованные объекты, которые могут совместно использоваться несколькими процессами, исполняющимися на одном или на нескольких компьютерах.

- **Уровень 8.** Отвечает за обмен информацией и сообщениями между процессами. На этом уровне происходит более богатый обмен информацией, чем на уровне 5, который обеспечивает работу первичного сигнального механизма для синхронизации процессов. Одним из наиболее мощных инструментов подобного типа является конвейер, представляющий собой логический канал передачи данных между процессами. Конвейер определяется как канал, передающий вывод одного процесса на вход другого; кроме того, он может быть использован и для связи с процессом внешних устройств или файлов.

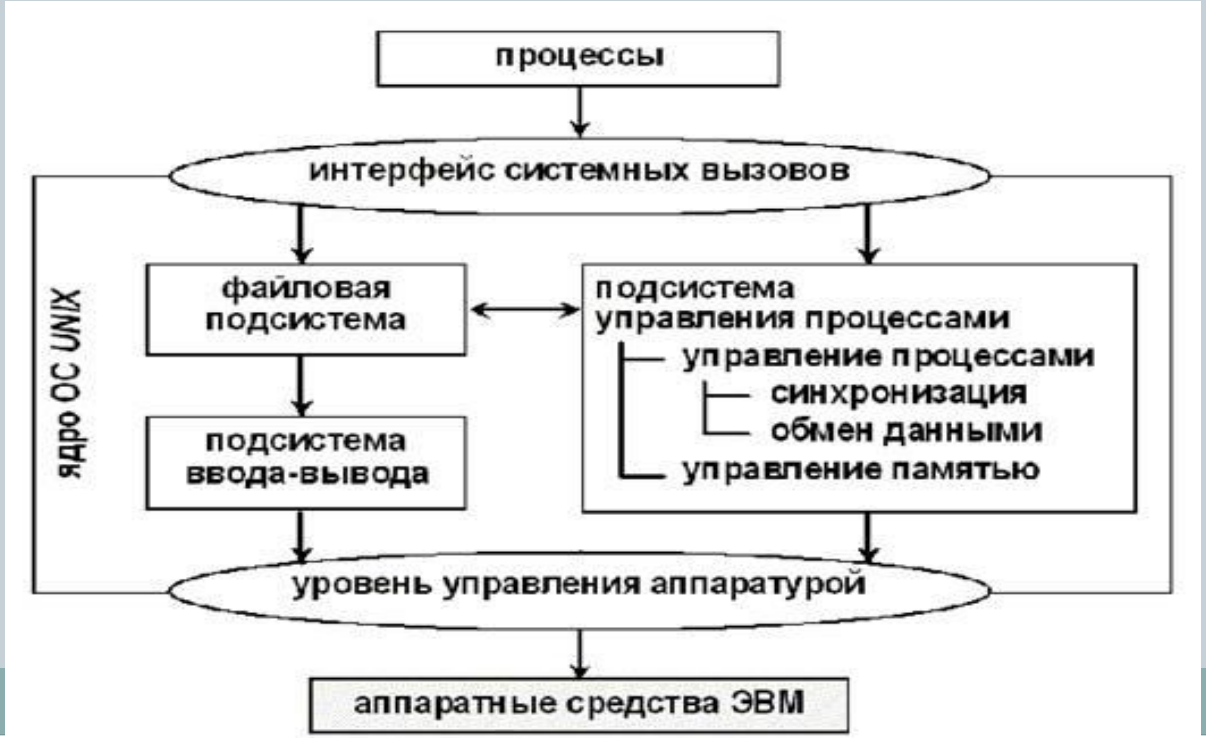
- **Уровень 9.** Обеспечивает долгосрочное хранение файлов. На этом уровне данные, хранящиеся на вспомогательном запоминающем устройстве, рассматриваются как абстрактные объекты переменной длины, в противоположность аппаратно-зависимому рассмотрению вторичной памяти как набора дорожек, секторов и блоков фиксированного размера, присущему уровню 6.



- **Уровень 10.** Предоставляет доступ к внешним устройствам с помощью стандартных интерфейсов

- **Уровень 11.** Поддерживает связь между внешними и внутренними идентификаторами системных ресурсов и объектов. Внешний идентификатор — это имя, которое может использоваться приложением или пользователем. Внутренний идентификатор — это адрес или другой индикатор, используемый нижними уровнями операционной системы для обнаружения объекта и управления им. Эта связь поддерживается с помощью каталога, который включает в себя не только взаимное отображение внешних и внутренних идентификаторов, но и такие характеристики, как, например, права доступа.

- **Уровень 12.** Предоставляет полнофункциональные средства поддержки процессов. Возможности этого уровня намного превосходят возможности уровня 5, на котором поддерживается только содержимое регистров процессора, имеющее отношение к процессу, и логика диспетчеризации процессов. На уровне 12 эта информация используется для упорядоченного управления процессами. Сюда же относится и виртуальное адресное пространство процессов, список объектов и процессов, с которыми оно может взаимодействовать, и правила, ограничивающие это взаимодействие; параметры, переданные процессам при их создании, и прочие характеристики процессов, которые могут быть использованы операционной системой для управления.

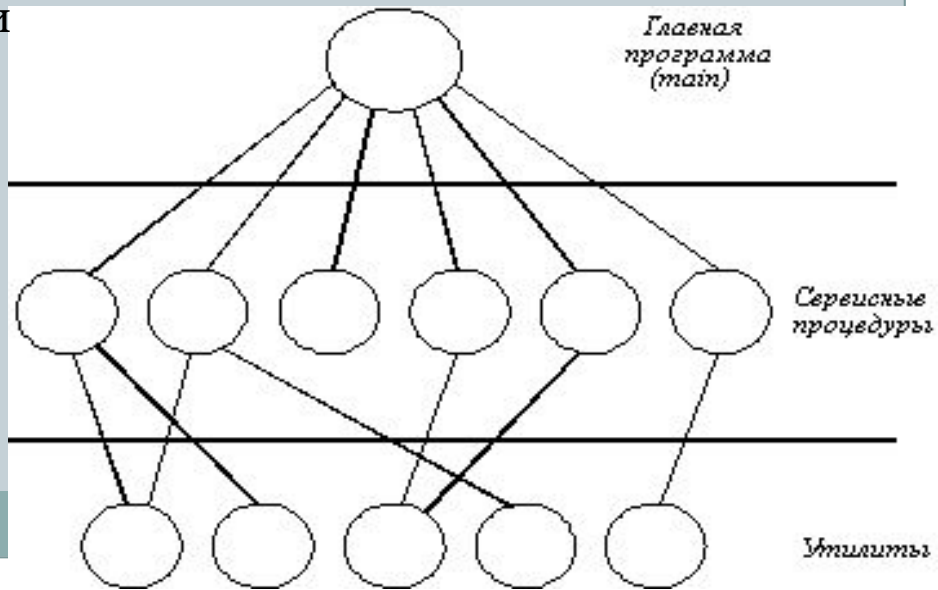


- **Уровень 13.** Обеспечивает взаимодействие операционной системы с пользователем. Этот уровень называется оболочкой (shell), так как он отделяет пользователя от деталей внутреннего устройства операционной системы и представляет ее пользователю как набор сервисов. Оболочка принимает команды пользователя или инструкции управления заданиями, интерпретирует их, создает необходимые процессы и управляет ими. На этом уровне, например, может быть реализован графический интерфейс, предоставляющий пользователю возможность выбора команды с помощью меню и отображающий результаты работы на экране.

Уровень	Название	Объекты	Пример операций
13	Оболочка	Пользовательская среда программирования	Инструкции командного языка оболочки
12	Пользовательские процессы	Пользовательские процессы	Завершение процесса, приостановка, возобновление работы
11	Каталоги	Каталоги	Создание, удаление, подключение, поиск
10	Устройства	Внешние устройства (принтер, монитор, клавиатура)	Открытие, закрытие, чтение, запись
9	Файловая система	Файлы	Создание, удаление, открытие, закрытие, чтение, запись
8	Коммуникации	Конвейеры	Создание, удаление, открытие, закрытие, чтение, запись
7	Виртуальная память	Сегменты, страницы	Чтение, запись, выборка
6	Локальная вторичная память	Блоки данных, каналы устройств	Чтение, запись, распределение, выборка
5	Примитивные процессы	Примитивные процессы, семафоры, список процессов	Приостановка, возобновление выполнения, ожидание и передача сигнала
4	Прерывания	Программы обработки прерываний	Вызов, маскирование, повтор
3	Процедуры	Процедуры, стеки вызова, дисплеи[3]	Вызов, возврат
2	Набор команд	Стек вычисления, интерпретатор микропрограмм, данные	Загрузка, сохранение, сложение, вычитание, ветвление
1	Электронные схемы	Регистры, шлюзы, шины и т.п.	Очистка, пересылка, активация

Монолитные системы

- Несомненно, такая организация операционной системы является самой распространенной. Здесь вся операционная система работает как единая программа в режиме ядра. Операционная система написана в виде набора процедур, связанных вместе в одну большую исполняемую программу. При использовании этой технологии каждая процедура может свободно вызвать любую другую процедуру, если та выполняет какое-нибудь полезное действие, в котором нуждается первая процедура. Наличие нескольких тысяч процедур, которые могут вызывать друг друга сколь угодно часто, нередко приводит к громоздкой и непонятной системе.



- Для построения исполняемого файла монолитной системы необходимо сначала скомпилировать все отдельные процедуры (или файлы, содержащие процедуры), а затем связать их все вместе, воспользовавшись системным компоновщиком. Здесь, по существу, полностью отсутствует сокрытие деталей реализации — каждая процедура видна любой другой процедуре (в отличие от структуры, содержащей модули или пакеты, в которых основная часть информации скрыта внутри модулей, и за пределами модуля его процедуры можно вызвать только через специально определяемые точки входа).

- Тем не менее даже такие монолитные системы могут иметь некоторую структуру. Службы (системные вызовы), предоставляемые операционной системой, запрашиваются путем помещения параметров в четко определенное место (например, в стек), а затем выполняется инструкция trap. Эта инструкция переключает машину из пользовательского режима в режим ядра и передает управление операционной системе). Затем операционная система извлекает параметры и определяет, какой системный вызов должен быть выполнен. После этого она перемещается по индексу в таблице, которая в строке k содержит указатель на процедуру, выполняющую системный вызов k.

- Такая организация предполагает следующую базовую структуру операционной системы:

  1. Основная программа, которая вызывает требуемую служебную процедуру.

  2. Набор служебных процедур, выполняющих системные вызовы.

  3. Набор вспомогательных процедур, содействующих работе служебных процедур.

  В этой модели для каждого системного вызова имеется одна ответственная за него служебная процедура, которая его и выполняет. Вспомогательные процедуры выполняют действия, необходимые нескольким служебным процедурам, в частности извлечение данных из пользовательских программ.

  В дополнение к основной операционной системе, загружаемой во время запуска компьютера, многие операционные системы поддерживают загружаемые расширения, в числе которых драйверы устройств ввода-вывода и файловые системы. Эти компоненты загружаются по мере надобности.

- Многоуровневые системы
- это системы с отдельно выделенными, иерархически расположенными, блоками (подсистемами), которые организованы по определенной структуре. Подсистемы, характеризующиеся рядом однородных характеристик по отношению к вышестоящим подсистемам, определяют уровень иерархической системы. Многоуровневые иерархические системы относятся к классу больших систем. Их широкое распространение в экономике, технике, военном деле обусловлено исторически сложившейся наиболее удобной формой обработки информации, распределения функций управления по отдельным уровням и подсистемам принятия решений на этих уровнях и в целом по системе. Многоуровневая экономическая система обычно сформирована в структуру, в которой выделяются взаимосвязанные иерархически расположенные подсистемы. Экономические подсистемы, характеризующиеся рядом однородных характеристик, определяют уровень иерархической системы, который замыкается и, как правило, подчинен подсистемам вышестоящего уровня.

Многоуровневые системы

Это системы с отдельно выделенными, иерархически расположенными, блоками (подсистемами), которые организованы по определенной структуре. Подсистемы, характеризующиеся рядом однородных характеристик по отношению к вышестоящим подсистемам, определяют уровень иерархической системы.

Многоуровневые системы (Layered systems)

5	Интерфейс пользователя
4	Управление вводом-выводом
3	Драйвер устройства связи оператора и консоли
2	Управление памятью
1	Планирование задач и процессов
0	Hardware

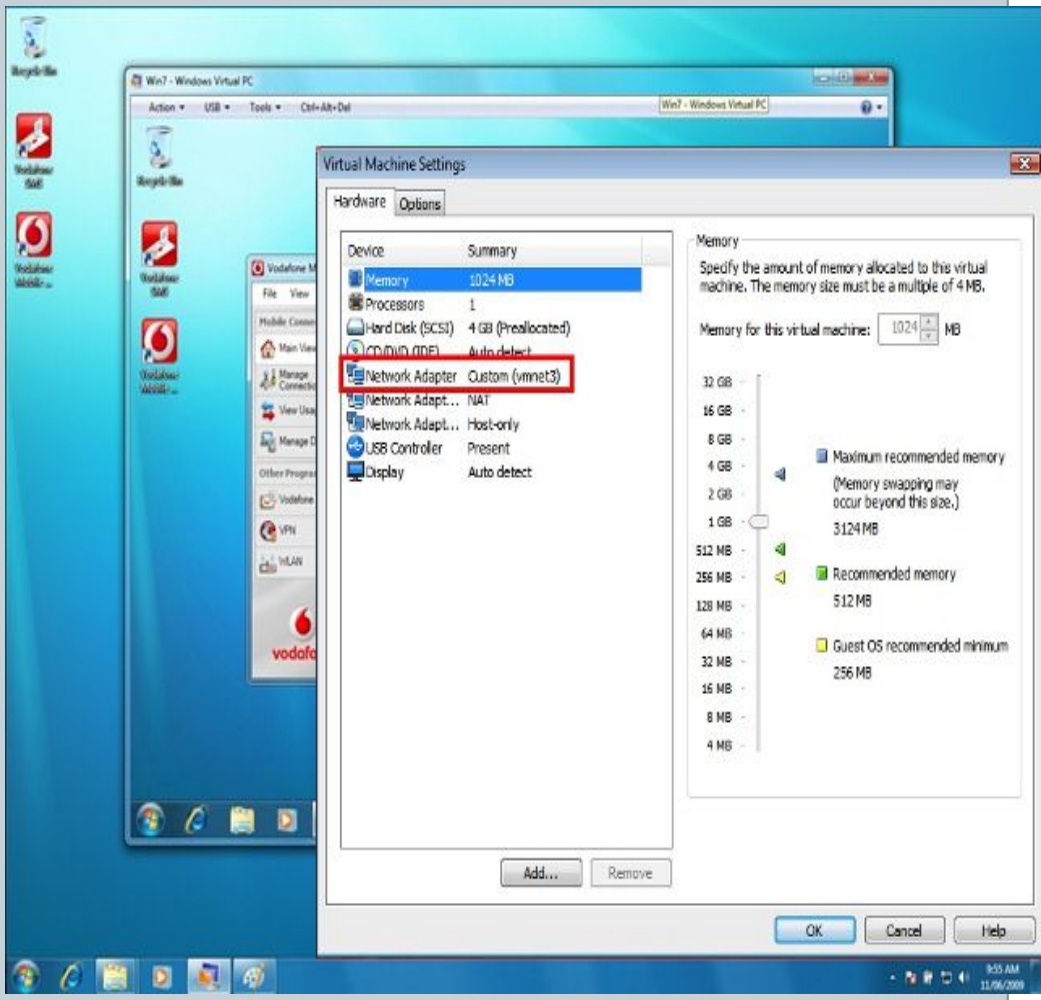
Слоеная система THE(Technishe Hogeschool Eindhoven) 1968 г

- Вся вычислительная система разбивается на ряд более мелких уровней с хорошо определенными связями между ними, так чтобы объекты уровня N могли вызывать только объекты уровня N-1

- Многоуровневые иерархические системы относятся к классу больших систем. Их широкое распространение в экономике, технике, военном деле обусловлено исторически сложившейся наиболее удобной формой обработки информации, распределения функций управления по отдельным уровням и подсистемам принятия решений на этих уровнях и в целом по системе. Многоуровневая экономическая система обычно сформирована в структуру, в которой выделяются взаимосвязанные иерархически расположенные подсистемы. Экономические подсистемы, характеризующиеся рядом однородных характеристик, определяют уровень иерархической системы, который замыкается и, как правило, подчинен подсистемам вышестоящего уровня.

Виртуальные машины

- Виртуальная машина — программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы (target — целевая, или гостевая платформа) и исполняющая программы для target-платформы на host-платформе (host — хост-платформа, платформа-хозяин) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы); также спецификация некоторой вычислительной среды (например: «виртуальная машина языка программирования Си»).



Виртуальные машины могут использоваться для:

- защиты информации и ограничения возможностей программ
- исследования производительности ПО или новой компьютерной архитектуры;
- эмуляции различных архитектур (например, эмулятор игровой приставки);
- оптимизации использования ресурсов мейнфреймов и прочих мощных компьютеров (см., например: IBM eServer);
- вредоносного кода для управления инфицированной системой: вирус PMBS, обнаруженный в 1993 году, а также руткит SubVirt, созданный в 2006 году в Microsoft Research, создавали виртуальную систему, которой ограничивался пользователь и все защитные программы (антивирусы и прочие).
- моделирования информационных систем с клиент-серверной архитектурой на одной ЭВМ (эмуляция компьютерной сети с помощью нескольких виртуальных машин).
- упрощения управления кластерами — виртуальные машины могут просто мигрировать с одной физической машины на другую во время работы.
- тестирования и отладки системного программного обеспечения;

Экзо-ядро

Экзо-ядро — ядро операционной системы компьютеров, предоставляющее лишь функции для взаимодействия между процессами и безопасного выделения и освобождения ресурсов.

Экзо — приставка, обозначающая нечто внешнее, находящееся снаружи.

В традиционных операционных системах ядро предоставляет не только минимальный набор сервисов, обеспечивающих выполнение программ, но и большое количество высокоуровневых абстракций для использования разнородных ресурсов компьютера: оперативной памяти, жестких дисков, сетевых подключений. В отличие от них, ОС на основе экзо-ядра предоставляет лишь набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой: выделение и высвобождение ресурсов, контроль прав доступа, и т. д. Экзо-ядро не занимается предоставлением абстракций для физических ресурсов — эти функции выносятся в библиотеку пользовательского уровня (так называемую libOS).

Модели клиент-сервер

- *Модели клиент-сервер* - это технология взаимодействия в информационной сети. *Сервер* обладает правом управления тем или иным ресурсом, а *клиент* – пользования им. Каждый конкретный сервер определяется видом того ресурса, которым он владеет. Например, назначением сервера баз данных является обслуживание запросов клиентов, связанных с обработкой данных; файловый сервер, или *файл-сервер,* распоряжается файловой системой.
- Этот принцип распространяется и на взаимодействие программ. Программа, выполняющая предоставление соответствующего набора услуг, рассматривается в качестве сервера, а программы, пользующиеся этими услугами, принято называть *клиентами.* Программы имеют распределенный характер, т.е. одна часть функций прикладной программы реализуется в программе-клиенте, а другая - в программе-сервере, а для их взаимодействия определяется некоторый *протокол.*