


Совместимость ОС

1. Виды совместимости

2. Способы реализации
совместимости

Виды совместимости

Существует 2 принципиально
отличающихся вида
совместимости, которые не
следует путать



Совместимость
на двоичном
уровне

Совместимость
на уровне
исходных
текстов

Виды совместимости

Приложения обычно хранятся в компьютере в виде исполняемых файлов, содержащих двоичные образы кодов и данных.

Двоичная совместимость достигается в том случае, если можно взять исполняемую программу, работающую в среде одной ОС и запустить ее на выполнение в среде другой ОС.

Виды совместимости

Совм-ть на уровне исх. текстов требует наличия соответствующих компиляторов в составе ПО ПК, на котором предполагается использовать данное приложение, а также совм-ти на уровне библиотек и системных вызовов. При этом необходима перекомпиляция исход-ных текстов программ в

Виды совместимости

Таким образом, совместимость на уровне исходных текстов наиболее **важное значение** имеет для разработчиков приложений, в распоряжении которых находятся эти исходные тексты.

Виды совместимости

Для пользователей значение имеет только двоичная совместимость, так как только в этом случае они могут без специальных навыков и умений использовать программный продукт, поставляемый в виде двоичного исполняемого кода, в различных операционных средах и на разных

Виды совместимости

Каким типом совместимости - двоичной или совместимостью исходных текстов обладает ОС, зависит от многих факторов.

Самый значительный из них - архитектура процессора, на котором работает ОС.

Виды совместимости

Чтобы достичь двоичной совм-ти достаточно соблюсти условия:

- вызовы функций API, которые содержат приложения должны поддерживаться данной ОС;
- внутренняя структура исполняемого файла приложения должна соответствовать структуре исполняемых файлов данной ОС.

Виды совместимости

Несравнимо сложнее достигнуть двоичной совместимости операционным системам, предназначенным для выполнения на процессорах, имеющих различающиеся архитектуры.

Кроме соблюдения приведенных выше условий, необходимо также организовать эмуляцию двоичного

Виды совместимости

Для того, чтобы компьютер смог интерпретировать машинные инструкции, которые ему изначально непонятны, на нем должно быть установлено специальное программное обеспечение –

эмулятор

Виды совместимости

Назначение эмулятора в том, чтобы последовательно выбирать каждую двоичную инструкцию процессора, например, Intel, программным способом дешифровать ее, чтобы определить, какие действия она задает, затем выполнять эквивалентную подпрограмму, написанную в инструкциях

Виды совместимости

Тем не менее, существует другой, гораздо более эффективный выход из описанной ситуации – **использование** так называемых **прикладных программных сред**. Одной из составляющих, формирующих программную среду, является набор функций интерфейса прикладного программирования API, которым

Виды совместимости

Для сокращения времени выполнения чужих программ прикладные среды имитируют обращения к библиотечным функциям.

Виды совместимости

Эффективность данного подхода определяется тем, что большинство современных программ работают под управлением графических интерфейсов пользователя (GUI) типа Windows, UNIX при этом приложения, как правило, наибольшую часть времени тратят на выполнение, некоторых

Виды совместимости

Они непрерывно осуществляют вызовы библиотек GUI для манипулирования окнами и для других, связанных с GUI, действий. Именно эта особенность приложений позволяет прикладным средам компенсировать большие затраты времени, потраченные на покомандное эмулирование

Виды совместимости

Тщательно спроектированная программная среда имеет в своем составе библиотеки, имитирующие внутренние библиотеки GUI, но написанные на "родном" коде данной ОС. Таким образом, достигается существенное ускорение выполнения программ с API

Виды совместимости

Для того чтобы отличить такой подход от более медленного процесса эмулирования кода по одной команде за раз, его называют **трансляцией**

Виды совместимости

Для того, чтобы программа, написанная для одной ОС, могла быть выполнена в рамках другой ОС, недостаточно лишь обеспечить совместимость API. Вполне может случиться так, что концепции, положенные в основу разных ОС, войдут в противоречие друг с другом.

Виды совместимости

Например, в одной ОС приложению может быть разрешено непосредственно управлять устройствами ввода-вывода, а в другой действия являются прерогативой ОС. Совершенно естественно, что каждая ОС имеет свои собственные механизмы защиты ресурсов, свои алгоритмы обработки ошибок и исключительных ситуаций, особую структуру процесса и схему управления памятью, свою семантику доступа к файлам и графический

Виды совместимости

Все эти отличия определяются спецификой аппаратной платформы, на которой работает ОС, особенностями ее реализации или заложенные разработчиками системы как присущие данной системе свойства.

Для обеспечения совместимости необходимо организовать бесконфликтное сосуществование в рамках одной ОС нескольких

Способы реализации совмест-ти

Задачей прикладной среды является выполнение программы по возможности так, как если бы она выполнялась на "родной" ОС. Но потребности этих программ могут входить в конфликт с конструкцией данной ОС.

Специализированные драйверы устройств не всегда отвечают требованиям безопасности, возможны конфликты между схемами

Способы реализации совмест-ти

Также прикладная среда должна выполнять программы с приемлемой скоростью.

Этому требованию не могут удовлетворить широко используемые ранее эмулирующие системы.

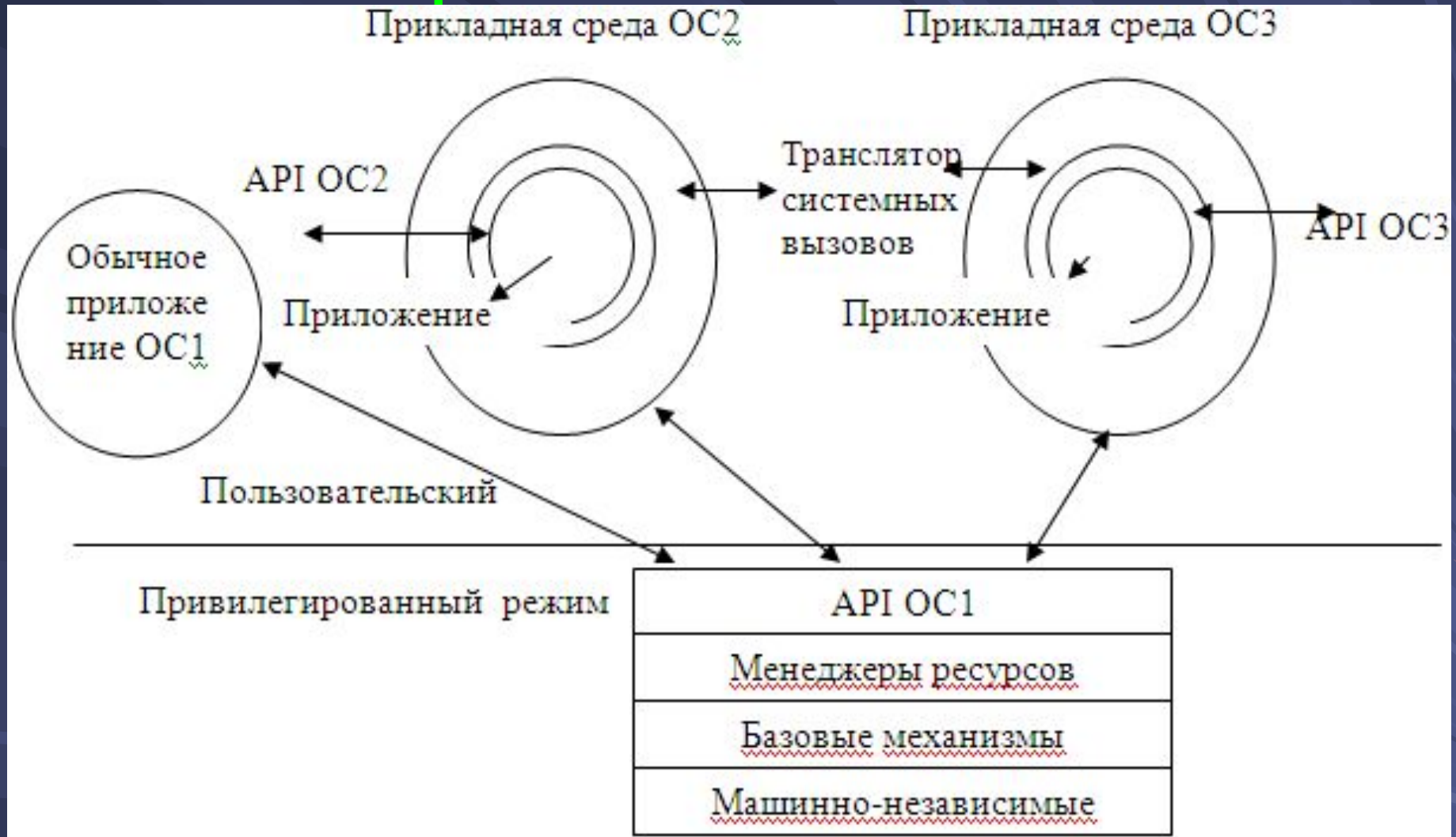
Для сокращения времени на выполнение чужих программ прикладные среды используют

Способы реализации совмест-ти

Несмотря на то, что реализация на практике полноценной прикладной среды, полностью совместимой со средой другой ОС является очень сложной задачей, существует несколько типовых подходов к ее решению.

Эти варианты отличаются особенностями архитектурных решений и функциональными возможностями, обеспечивающими различную

Способы реализации совмест-ти



Трансляция системных вызовов с помощью прикладных программных сред

Способы реализации совмест-ти

ОС1 поддерживает кроме своих "родных" приложений приложения ОС2 и ОС3.

Для этого есть прикладные программные среды, которые транслируют интерфейсы "чужих" API ОС2 и API ОС3 в интерфейс своей "родной" API ОС1.

Способы реализации совмест-ти

Существует способ построения множественных прикладных сред, использующий концепцию микро-ядерного подхода. При этом важно отделить базовые, общие для всех прикладных сред механизмы ОС от специфических для каждой из прикладных сред высокоуровневых функций, решающих стратегические задачи

Способы реализации совмест-ти

В соответствии с микроядерной архитектурой функции ОС реализуются микроядром и серверами пользовательского режима.

Важно: каждая прикладная среда оформляется в виде отдельного сервера пользовательского режима и не включает базовых механизмов

Способы реализации совмест-ти

Приложения, используя API, обращаются с системными вызовами к соответствующей прикладной среде через микроядро. Прикладная среда обрабатывает запрос, выполняет его (возможно, обращаясь для этого за помощью к базовым функциям микроядра) и отправляет приложению результат

Способы реализации совмест-ти

В ходе выполнения запроса прикладной среде приходится обращаться к базовым механизмам ОС, реализуемым микроядром и другими серверами ОС. Такому подходу к конструированию множественных прикладных средств присущи все достоинства и недостатки микроядерной архитектуры, в

Способы реализации совмест-ти

- очень просто можно добавлять и исключать прикладные среды, что является следствием хорошей расширяемости микроядерных ОС;
- надежность и стабильность: при отказе одной из прикладных сред все остальные сохраняют работоспособность;
- низкая производительность микроядер-ных ОС сильно сказывается на скорости работы прикладных сред а значит и на

Способы реализации совмест-ти

Множественные прикладные среды обеспечивают совместимость на двоичном уровне данной ОС с приложениями, написанными для других ОС. В результате пользователи получают большую свободу выбора ОС и более легкий доступ к качественному программному обеспечению.

Способы реализации совмест-ти

Вывод

Чтобы программа, написанная для одной ОС, могла быть выполнена в рамках другой ОС, недостаточно совместимости API. Необходимо "родное" окружение: структура процесса, средства управления памятью, средства обработки ошибок и исключительных ситуаций, механизмы защиты ресурсов и семантика файлового доступа.

Способы реализации совмест-ти

Вывод

Значит поддержка
нескольких прикладных
программных сред
является очень
непростой задачей,
тесно связанной со
структурой ОС

Системные вызовы (system calls) – интерфейс между ОС и пользовательской программой. Они создают, удаляют и используют различные объекты, главные из которых – процессы и файлы. Пользовательская программа запрашивает сервис у ОС, 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.

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 большинстве ОС системный вызов осуществляется командой программного прерывания (INT).

Прерывание (**hardware interrupt**) – это событие, генерируемое внешним (по отношению к процессору) устройством. посредством аппаратных прерываний аппаратура либо информирует процессор о том, что произошло 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 аппаратн. прерываний
– **прерывания таймера**,
генерируе-мые периодически через
фиксиро-ванный промежуток
времени.

Прерывания таймера использу-
ются ОС при планировании
процессов. Каждый тип аппаратных
прерываний имеет собственный
номер, однозначно определяющий

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.

Исключительная

ситуация

(**exception**) – событие, возникающее в результате попытки выполнения программой команды, которая по каким-либо причинам не может быть выполнена до конца.

Примерами таких команд могут быть попытки доступа к ресурсу при отсутствии достаточных привилегий или обращения к отсутствующей странице памяти.

Исключительные ситуации 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 выполняться дальше. Возникновение в процессе работы ОС исправимых исключительных ситуаций считается нормой.

Неисправимые исключительные ситуации чаще всего возникают в результате ошибок в программах (например, деление на ноль).

Обычно в таких случаях ОС реагирует завершением программы, вызвавшей исключительную ситуацию.

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

Главная задача **файловой системы (ФС)** – скрыть особенности ввода-вывода и дать программисту простую абстрактную модель файлов, независимых от устройств. Для чтения, создания, удаления, записи, открытия и закрытия файлов имеется обширная категория системных вызовов (создание, удаление, открытие, закрытие, чтение и т.д.).

Пользователям хорошо знакомы такие связанные с организацией файловой системы понятия, как:

- ❖ *каталог,*
- ❖ *текущий каталог,*
- ❖ *корневой каталог,*
- ❖ *путь.*

Для манипулирования этими объектами в ОС имеются системные вызовы.