**СПО**

**Виртуализация** - *предоставление набора вычислительных ресурсов или их логического объединения*

Примером использования визуализации является возможность запуска нескольких ОС на одном компе  
Для виртуализации ОС применяется серия подходов, которые по типу реализации разделяются на программные и аппаратные

Программная виртуализация:  
• *динамическая трансляция. При такой трансляции проблемные команды гостевой ОС перехватываются гипервизором.  
После того как эти команды заменяются на безопасные, происходит возврат управления гостевой сис-мы  
• паравиртуализация. Это техника виртуализации, при которой гостевые ОС подготавливаются для исполнения в виртуализированной  
среде. Метод правиртуалзации позволяет добиться более высокой производительности применим лишь тогда, если гостевые ОС имеют  
открытые исходные коды, которые можно модифицировать.  
• встроенная виртуализация. Преимущества - удобства интерфейса для окон приложений из разных сис-ем. При тонкой настройке  
на аппаратную платформу производительность мало отличается от оригинальной ОС. Простая процедура обновления гостевой ОС.*

**Двухсторонняя виртуализация  
Аппаратная виртуализация.**

**Преимущества** - *упрощение разработки аппаратных платформ виртуализации за счет предоставления аппаратных  
интерфейсов управления и поддержки виртуальных гостевых систем.*

Возможность увеличения быстродействия платформ виртуализации.

Улучшается защищенность, появляется возможность переключения между несколькими запущенными независимыми платформами виртуализации  
на аппаратном уровне. Гостевая система становится не привязана к архитектуре хостовой платформы и к реализации платформы  
визуализации

Виртуализация на уровне ОС позволяет запускать изолированные и безопасные виртуальные машины на одном физ-ом узле, но не  
позволяет запускать ОС с ядрами, отличными от типа ядра базовой ОС

Области применения виртуализации

**Виртуальная машина** - *это окружение, которое представляется для гостевой ОС, как аппаратная. Однако это программное окружение, которое  
эмулируется программным обеспечение хостовой сис-мы. Эта эмуляция должна быть достаточно надежной, чтобы драйверы гостевой сис-мы* *могли стабильно работать*.

При использовании паравиртуализации виртуальная машина не эмулирует аппаратное обеспечение, а предлагает  
использовать специальное API. Виртуализация ресурсов может быть представлена как разделение одного физ-ого узла на несколько частей,  
каждая из которых видна для владельца в качестве отдельного сервера. Не является технологией виртуальных машин. Осуществляется на уровне ядра ОС.

Для виртуализации приложений, программное обеспечение визуализатора определяет при установке виртуализируемого  
приложения какие требуются ОС и эмулирует их. Таким образом создается необходимая специализированная среда для конкретно этого виртуализированного приложения.

Стандартные методы обслуживания ОС. Инженер должен создать точку восстановления:  
*• обновление ОС  
• установка или обновление оборудования  
• установка приложения  
• установка драйвера*

Чтобы открыть служебную программу и восстановление сис-мы необходимо выполнить:  
Пуск - все программы - стандартные - служебные - восстановление системы :D

**Обзор оборудования и мобильных устройств**

RISC технология означает упрощенный набор команд, ее впервые предложили в компании IBM. В ее основу положена идея максимального повышения быстродействия посредством упрощения инструкций и ограничения их длины. Благодаря этому подходу стало возможным не только повысить тактовую частоту, но и сократить очередь из команд на выполнения, а также снизить тепловыделение и потребление энергии  
CISC процессоры. Это более традиционный вид микропроцессоров, которые отличается от предыдущих полным набором команд. Такие процессоры не имеют фиксированной длины команд, а самих команд больше. Классическая CISC архитектура используется все реже из за пониженной тактовой частоты и высокой стоимости сборки, однако по прежнему востребована в серверах и рабочих станциях.

ARMIх86. Процессоры такой фирмы используется в большинстве мобильных устройств, архитектура х86 господствует в настольных компах.  
Big.Little. Это технология оптимизации потребления энергии за счет объединения более высокопроизводительных ядер с менее производительными (Cortex A15 и Cortex A7)  
HELIOx20 это новейший 10-ядерный чип медиатег на базе ARM CORTEX. Данный чип способен проработать на 30% дольше времени чем аналоги с сопоставимыми хар-ми, выполнялись тесты даже для конкретных сценариев  
Возможности трехкластерных SoС-процессоров MediaTek.

Особенности:  
*1) новейший видеочип ARM Mali обеспечивает высочайшую производительность графики в 2мерном и 3мером пространстве  
2) дополнительные встроенный процессор CORTEXm4 работает в фоновом режиме с крайне низким энергопотреблением, обеспечивая работу фоновых приложений  
3) контроллер 2 камер со встроенным 3д движком не только быстро работает, но и эффективно генерирует сложные объемные изображения, а встроенная технология шумоподавления доводит картинку до практически идеальной. Дисплей может работать с частотой обновления 120Гц вместо стандартных 60Гц что дает четкое изображение.*  **Потоки**

поток использует то же пространство стека, что и процесс. Каждый потом может работать с одной и той же областью памяти, в отличие от процессов, которые не могут получить доступ к памяти другого процесса. У каждого потока может быть собственный регистр и стек, но другие потоки могут их использовать. Когда 1 поток изменяет ресурс процесса, это изменение видно другим потокам. В параметрах указывается имя процедуры, которая выполняется в новом потоке.  
thread\_join - процедура блокирует вызывающий поток  
thread\_yield - позволяет процессу вступить.  
Ресурсы  
-это средство вычислительной системы, которое может быть выделено процессору на определенный интервал времени.  
По реальности существования:  
-физический (типа реально существует, в отличие от твоей девушки)  
-виртуальный (мнимый ресурс, который не существует в том виде, в котором показывает себя пользователю)  
Свойства ресурсов и их представления  
Виды ресурсов:  
• -аппаратные:  
• -ОЗУ, ПЗУ  
• -процессорное время  
• -пропускная способность  
• -программные  
• -файловые  
• -сетевые  
Цикл работы процессора:  
• выборка из памяти 1 команды  
• декодирование этой команды  
• выполнение этой команды → выборка команды..  
Для каждого типа процессора есть определенный набор команд, который он может выполять  
Регистр процессора — блок ячеек памяти, образующих сверхбыструю оперативную память внутри процессора  
Регистр общего назначения — применяется для хранения переменных и промежуточных результатов  
Регистр специального назначения:  
• -счетчик команд (указатель на следующую команду)  
• -регистр флагов (содержит состояние процессора)  
• -указатель стека — указывает на вершину стека в памяти  
У процессора могут быть отдельные блоки, для выборки и декодирования команд. Во время выполнения команды n, он может декодировать n+1 и писать n+2. Это называется конвеером, или ubisoft.  
Блок выбора — блок декодирования — исполнительный блок.  
Суперскалярный процессора имеет несколько исполнительных блоков, например 1 для целочисленной арифметики, 2 для арифметики с плавающей точкой и 3 для логических операций. Одновременно выбираются 2 команды, которые декодируются и помещаются в буферы хранения.  
Хар-ки процессора  
• тактовая частота (Ггц)  
• разрядность шины данных (кол-во инфы)  
• поддерживаемые инструкции  
• объем кэш памяти  
Иерархия:  
Регисты (1нс)  
Кэш (2нс)  
Озу (10нс)  
Пзу (10мс)  
Магнитная лента (100сек) :D  
Хранения программ и данных в процессе выполнения программ.  
Достоинства:  
быстродействие  
Недостатки:  
Энергозависимость  
ПЗУ — энергонезависимые устройства, предназначенные для хранения программной информации  
Внешняя память — внешние ЗУ (флешки и т. д.)  
Понятие тупика — процесс находится в состоянии тупика, детлока или клинча, если он ожидает некоторое событие, которое никогда не произойдет.  
Примеры тупиков:  
-ожидание события. Системе придется обнаруживать что данный процесс завис и онулировать его, чтобы выйти из тупика  
-бесконечное откладывание. Может происходить из-за дескриминационной политики планировщиков ресурсной системы.



Пример детлока  
Запрос/ Ресурс 1 Запрос\  
процесс 2 процесс 1  
Запрос\ Ресурс 2 Запрос/  
  
У процесса есть право монопольного управления ресурса.  
Процессы удерживают за собой ресурсы, уже выделенные им, ожидая выделения дополнительных ресурсов.  
Ресурсы нельзя отобрать у процессов, удерживающих их, пока эти ресурсы не будут использованы для завершения работы.  
Предотвращение тупиков  
-каждый процесс должен запрашивать все требуемые ресурсы сразу, при этом не выполняться, пока они ему не будут предоставлены.  
  
Управление памятью  
память — важный ресурс, который требует точного выполнения. Со временем была разработана концепция иерархии памяти. Превратить иерархию в абстракцию и затем управлять ей и есть задача ОС. Этим занимается компонент ОС — диспетчер памяти  
Функции ОС по управлению памятью:  
-учет свободной памяти  
-статическая и динамическая выделения памяти процессом и ее освобождение при завершении процесса.  
-Вытеснение процессов из ОП на диск  
-защита памяти, выделенные процессу от вмешательства других процессов  
Виртуализация памяти:  
• свопинг (процесс входа-выхода памяти с диска)  
• виртуальная память (между памятью и диском перемещаются образы процессов (сегменты, блоки, страницы))  
Недостатки свопинга — избыточность перемещаемых данных → неэффективность использования памяти  
Достоинства свопинга — меньшие затраты времени на преобразование адресов  
Виртуальная память не имеет указанных недостатков, но ее проблемой является преобразование логических адресов в физические  
Символьные имена  
транслятор  
Виртуальные адреса  
перемещающий загрузчик  
Физические адреса  
Символьные имена присваивает пользователь на алгоритмических языках программирования  
Физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности будут располагаться данные и команды  
Совокупность виртуальных адресов и процессов называется его виртуальным адресным пространством. Диапазон адресов виртуального пространства определяется разрядностью процесса  
Методы распределения памяти:  
• -фиксированными разделами  
• -динамическими разделами  
• -перемещаемыми разделами  
8мб — ОС  
Пр1  
8мб -  
Пр2  
8мб -  
-  
8мб -  
Пр3  
Одинаковые разделы  
-ОС-  
3мб  
Пр1  
4мб  
Пр2  
8мб  
Пр3  
Разные разделы  
  
  
Уровни планирования:  
Задания, ожидающие входа в систему  
Задания, ожидающие запуска  
Планирования верхнего уровня  
Приостановленные процессы  
-Активация Планирование промежуточного уровня  
Активные процессы  
Блокирование Планирование нижнего уровня  
Выполняющиеся процессы  
Завершенные процессы  
Дисциплина планирования должна быть:  
1. справедливой  
2. обеспечивать максимальную пропускную способность системы  
3. приемлимое время ответа для максимального кол-ва пользователей, работающих в интерактивном режиме  
4. сбалансирование использование ресурсов  
5. должна исключать бесконечное откладывание процессов  
6. учитывать приоритет  
Факторы, учитываемые при планировании:  
1) обязательно ли малое время ответа  
2) приоритет каждого процесса  
3) частоту переключений с низко приоритетных процессов  
4) длительность периода времени, в течении которого ожидат каждый процесс  
Планирование без переключений предусматривают что после предоставления ЦП какому либо процессу отобрать ЦП у этого процесса нельзя  
Приоритеты бывают статическими, динамическими и покупаемыми  
FIFO (first In first out)  
X Y Z → ЦП  
Циклическое планирование  
Каждый процесс получает временной квант, в течение которого он может использовать ЦП. Если завершение процесса не происходит, то этот процесс переводится в конец списка готовых к выполнению процессов, а ЦП предоставляется следующему процессу из списка  
Если квант времени выбирается слишком большим то система RR превращается в FIFO. Если квант выбирается слишком малым то контекстные переключения играют доминирующую роль, что ухудшает характеристики системы  
Многоуровневые очереди с обратными связями  
Квант времени при переходе в очередь более низкого уровня увеличивается  
Нижняя очередь организована по принципу RR, где процесс циркулирует до своего завершения.  
  
Система программирования является средством



разработки программ и включают:  
1) компилятор  
2) среду разработки программ  
3) отладчик  
4) оптимизация кода программ  
5) набор библиотек  
6) редактор связи  
7) сервисные утилиты  
8) справочные системы  
9) документатор исходного кода программ  
10) системы поддержки и управления проектом  
Инструментальная среда пользователя - специальные средства, встроенные в пакеты прикладных программ: библиотека функций, процедур и методов обработки.  
Языковые макросы - конструкторы экранных форм и объектов  
Конструкторы меню и другие…  
Интегрированная среда разработки - развитие локальных средств разработи.  
Дальнейшее развитие локальных средств разработки - повышение производительности труда программистов, автоматизация создания программ, разработка приложений для архитектуры клиент-сервер, запросов и отчетов.  
Этапы компилянции:  
1) препроцессинг (работа с директивами препроцессора (добавляет хеддоры в код))  
2) компиляция (преобразует код в Assember)  
Объектый файл, созданный Assember, хранящий машинный код, который еще не был связан с другими кусками машинного кода, конечного выполняемого программой (объектый код). Возможно сохранение данного объектного кода в статические библиотеки, чтобы не компилировать данный код снова.  
Компановщик (редактор связи) - связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы не можем запустить в дальнейшем. Таблица символов - структура данных, которая создается компилятором и хранится в объектных файлах. В ней хранятся имена переменных, функций и классов, где каждому идентификатору соотносится его тип и область видимости (using namespace std)(чтобы не было конфликта имен). Таблица символов хранит адреса ссылок на данные процедур других файлов.  
  
  
  
  
  
На этапе анализа выполняется распознавание текста, создание и заполнение таблиц идентификаторов. Результатом работы служит внутреннее представление программы, понятное компилятору. На этапе синтеза, на основании внутреннего представление программы и информации, содержащейся в таблице идентификаторов, покаждается текст программы. Так же присутствует компонент, отвечающий за анализ и исправление ошибок.  
Лексический анализ - проводится лексический контроль