Архитектура операционных систем

- Основные принципы построения операционных систем
  - Принцип модульности
  - Принцип функциональной избирательности
  - Принцип генерируемости ОС
  - Принцип функциональной избыточности
  - Принцип виртуализации
  - Принцип независимости программ от внешних устройств
  - Принцип совместимости
  - Принцип открытой и наращиваемой ОС
  - Принцип мобильности (переносимости)
  - Принцип обеспечения безопасности вычислений

Принцип модульности

- Под модулем в общем случае понимают функционально законченный элемент системы, выполненный в соответствии с принятыми межмодульными интерфейсами. По своему определению модуль предполагает возможность без труда заменить его на другой при наличии заданных интерфейсов.
- Особо важное значение при построении ОС имеют привилегированные, повторно входимые и реентерабельные модули, так как они позволяют более эффективно использовать ресурсы вычислительной системы.
- Принцип модульности отражает технологические и эксплуатационные свойства системы. Наибольший эффект от его использования достижим в случае, когда принцип распространен одновременно на операционную систему, прикладные программы и аппаратуру.

Принцип функциональной избирательности

- В ОС выделяется некоторая часть важных модулей, которые должны постоянно находиться в оперативной памяти для более эффективной организации вычислительного процесса. Эту часть в ОС называют ядром, так как это действительно основа системы. При формировании состава ядра требуется учитывать два противоречивых требования.
  - В состав ядра должны войти наиболее часто используемые системные модули.
  - Количество модулей должно быть таковым, чтобы объем памяти, занимаемый ядром, был бы не слишком большим.
- В состав ядра, как правило, входят модули по управлению системой прерываний, средства по переводу программ из состояния счета в состояние ожидания, готовности и обратно, средства по распределению таких основных ресурсов, как оперативная память и процессор.

Принцип функциональной избирательности

- Помимо программных модулей, входящих в состав ядра и постоянно располагающихся в оперативной памяти, может быть много других системных программных модулей, которые получают название транзитных.
- Транзитные программные модули загружаются в оперативную память только при необходимости и в случае отсутствия свободного пространства могут быть замещены другими транзитными модулями.
- В качестве синонима к термину «транзитный» можно использовать термин «диск-резидентный».

Принцип генерируемости ОС

- Основное положение этого принципа определяет такой способ исходного представления центральной системной управляющей программы ОС (ее ядра и основных компонентов, которые должны постоянно находиться в оперативной памяти), который позволял бы настраивать эту системную супервизорную часть, исходя из конкретной конфигурации конкретного вычислительного комплекса и круга решаемых задач.
- Эта процедура проводится редко, перед протяженным периодом эксплуатации ОС. Процесс генерации осуществляется с помощью специальной программы-генератора и соответствующего входного языка для этой программы, позволяющего описывать программные возможности системы и конфигурацию машины.
- В результате генерации получается полная версия ОС. Сгенерированная версия ОС представляет собой совокупность системных наборов модулей и данных.

Принцип генерируемости ОС

- **Принцип генерируемости** существенно упрощает настройку ОС на требуемую конфигурацию вычислительной системы. В наши дни при использовании персональных компьютеров с принципом генерируемости ОС можно столкнуться разве что только при работе с Linux.
- В этой UNIX-системе имеется возможность не только использовать какое-либо готовое ядро ОС, но и самому сгенерировать (скомпилировать) такое ядро, которое будет оптимальным для данного конкретного персонального компьютера и решаемых на нем задач.
- Кроме генерации ядра в Linux имеется возможность указать и набор подгружаемых драйверов и служб, то есть часть функций может реализовываться модулями, непосредственно входящими в ядро системы, а часть — модулями, имеющими статус подгружаемых, транзитных.

Принцип функциональной избыточности

- Этот принцип учитывает возможность проведения одной и той же работы различными средствами. В состав ОС может входить несколько типов мониторов (модулей супервизора, управляющих тем или другим видом ресурса), различные средства организации коммуникаций между вычислительными процессами.
- Наличие нескольких типов мониторов, нескольких систем управления файлами позволяет пользователям быстро и наиболее адекватно адаптировать ОС к определенной конфигурации вычислительной системы, обеспечить максимально эффективную загрузку технических средств при решении конкретного класса задач, получить максимальную производительность при решении заданного класса задач.

Принцип виртуализации

- Этот принцип позволяет представить структуру системы в виде определенного набора планировщиков процессов и распределителей ресурсов (мониторов) и использовать единую централизованную схему распределения ресурсов.
- Наиболее естественным и законченным проявлением концепции виртуальности является понятие виртуальной машины. По сути, любая операционная система, являясь средством распределения ресурсов и организуя по определенным правилам управление процессами, скрывает от пользователя и его приложений реальные аппаратные и иные ресурсы, заменяя их некоторой абстракцией.

Принцип виртуализации

- Чаще виртуальная машина, предоставляемая пользователю, воспроизводит архитектуру реальной машины, но архитектурные элементы в таком представлении выступают с новыми или улучшенными характеристиками:
  - единообразная по логике работы память (виртуальная) практически неограниченного объема. Среднее время доступа соизмеримо со значением этого параметра оперативной памяти. Организация работы с информацией в такой памяти производится в терминах обработки данных — в терминах работы с сегментами данных на уровне выбранного пользователем языка программирования;

Принцип виртуализации

- произвольное количество процессоров (виртуальных), способных работать параллельно и взаимодействовать во время работы. Способы управления процессорами, в том числе синхронизация и информационные взаимодействия, реализованы и доступны пользователям;
- произвольное количество внешних устройств (виртуальных), способных работать с памятью виртуальной машины параллельно или последовательно, асинхронно или синхронно по отношению к работе того или иного виртуального процессора, которые инициируют работу этих устройств. Информация, передаваемая или хранимая на виртуальных устройствах, не ограничена допустимыми размерами. Доступ к такой информации осуществляется на основе либо последовательного, либо прямого способа доступа.

Принцип виртуализации

- Чем больше виртуальная машина, реализуемая средствами ОС на базе конкретной аппаратуры, приближена к «идеальной» по характеристикам машине и, следовательно, чем больше ее архитектурно-логические характеристики отличны от реально существующих, тем больше степень виртуальности.
- Одним из аспектов виртуализации является организация возможности выполнения в данной ОС приложений, которые разрабатывались для других ОС. Реализация этого принципа позволяет такой ОС иметь очень сильное преимущество перед аналогичными ОС, не имеющими такой возможности.

Принцип независимости программ от внешних устройств

- Этот принцип заключается в том, что связь программ с конкретными устройствами производится не на уровне трансляции программы, а в период планирования её исполнения. В результате перекомпиляция при работе программы с новым устройством, на котором располагаются данные, не требуется.
- Принцип позволяет одинаково осуществлять операции управления внешними устройствами независимо от их конкретных физических характеристик. Например, программе, содержащей операции обработки последовательного набора данных, безразлично, на каком носителе эти данные будут располагаться. Смена носителя и данных, размещаемых на них (при неизменности структурных характеристик данных), не принесет каких-либо изменений в программу, если в системе реализован принцип независимости.

Принцип совместимости

- Это способность ОС выполнять программы, написанные для других ОС или для более ранних версий данной операционной системы, а также для другой аппаратной платформы.
- Необходимо разделять вопросы двоичной совместимости и совместимости на уровне исходных текстов приложений.
- Двоичная совместимость достигается в том случае, когда можно взять исполняемую программу и запустить ее на выполнение на другой ОС. Для этого необходимы:
  - совместимость на уровне команд процессора,
  - совместимость на уровне системных вызовов и даже на уровне библиотечных вызовов, если они являются динамически связываемыми.

Принцип совместимости

- Совместимость на уровне исходных текстов требует наличия соответствующего транслятора в составе системного программного обеспечения, а также совместимости на уровне библиотек и системных вызовов. При этом необходима перекомпиляция имеющихся исходных текстов в новый выполняемый модуль.
- Гораздо сложнее достичь двоичной совместимости между процессорами, основанными на разных архитектурах. Для того чтобы один компьютер выполнял программы другого (например, программу для ПК типа IBM PC желательно выполнить на ПК типа Macintosh фирмы Apple), этот компьютер должен работать с машинными командами, которые ему изначально непонятны.
- Выходом в таких случаях является использование так называемых прикладных сред или эмуляторов.
- Одним из средств обеспечения совместимости программных и пользовательских интерфейсов является соответствие стандартам POSIX. Использование стандарта POSIX позволяет создавать программы в стиле UNIX, которые впоследствии могут легко переноситься из одной системы в другую.

Принцип открытой и наращиваемой ОС

- Открытая ОС доступна для анализа как пользователям, так и системным специалистам, обслуживающим вычислительную систему.
- Наращиваемая (модифицируемая, развиваемая) ОС позволяет не только использовать возможности генерации, но и вводить в ее состав новые модули, совершенствовать существующие и т. д. Необходимо, чтобы можно было внести дополнения и изменения, и не нарушить целостность системы.
- Прекрасные возможности для расширения предоставляет подход к структурированию ОС по типу клиент—сервер с использованием микроядерной технологии. В соответствии с этим подходом ОС строится как совокупность привилегированной управляющей программы и набора непривилегированных услуг — «серверов». Основная часть ОС остается неизменной и в то же время могут быть добавлены новые серверы или улучшены старые. Этот принцип иногда трактуют как расширяемость системы.
- К открытым ОС, прежде всего, следует отнести UNIX-системы и, естественно, ОС Linux.

Принцип мобильности (переносимости)

- Операционная система относительно легко должна переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которая включает наряду с типом процессора и архитектуру вычислительной системы) одного типа на аппаратную платформу другого типа. Принцип переносимости очень близок принципу совместимости, но это не одно и то же.
  1. Большая часть ОС должна быть написана на языке, который имеется на всех системах, на которые планируется в дальнейшем ее переносить. Это, прежде всего, означает, что ОС должна быть написана на языке высокого уровня, предпочтительно стандартизованном, например на языке С. Программа, написанная на ассемблере, не является в общем случае переносимой.

Принцип мобильности (переносимости)

2. Важно минимизировать или, если возможно, исключить те части кода, которые непосредственно взаимодействуют с аппаратными средствами. Зависимость от аппаратуры может иметь много форм. Некоторые очевидные формы зависимости включают прямое манипулирование регистрами и другими аппаратными средствами.
3. Если аппаратно-зависимый код не может быть полностью исключен, то он должен быть изолирован в нескольких хорошо локализуемых модулях. Аппаратно-зависимый код не должен быть распределен по всей системе. Например, можно спрятать аппаратно-зависимую структуру в программно задаваемые данные абстрактного типа.

- Введение стандартов POSIX преследовало цель обеспечить переносимость создаваемого программного обеспечения.

Принцип обеспечения безопасности вычислений

- Обеспечение безопасности при выполнении вычислений является желательным свойством для любой многопользовательской системы. Правила безопасности определяют такие свойства, как защита ресурсов одного пользователя от других и установление квот по ресурсам для предотвращения захвата одним пользователем всех системных ресурсов (таких, как память).
- Обеспечение защиты информации от несанкционированного доступа является обязательной функцией сетевых операционных систем. Во многих современных ОС гарантируется степень безопасности данных, соответствующая уровню С2 в системе стандартов США.

Принцип обеспечения безопасности вычислений

- Безопасной считается система, которая «посредством специальных механизмов защиты контролирует доступ к информации таким образом, что только имеющие соответствующие полномочия лица или процессы, выполняющиеся от их имени, могут получить доступ на чтение, запись, создание или удаление информации».
- Иерархия уровней безопасности, приведенная в стандартах, помечает низший уровень безопасности как D, а высший — как A.
- В класс D попадают системы, оценка которых выявила их несоответствие требованиям всех других классов.

Принцип обеспечения безопасности вычислений

- Основными свойствами, характерными для систем класса С, являются наличие подсистемы учета событий, связанных с безопасностью, и избирательный контроль доступа. Класс (уровень) С делится на 2 подуровня: уровень С1, обеспечивающий защиту данных от ошибок пользователей, но не от действий злоумышленников; и уровень С2. На уровне С2 должны присутствовать:
  - средства секретного входа, обеспечивающие идентификацию пользователей путем ввода уникального имени и пароля перед тем, как им будет разрешен доступ к системе;
  - избирательный контроль доступа, позволяющий владельцу ресурса определить, кто имеет доступ к ресурсу и что он может с ним делать. Владелец делает это путем предоставляемых прав доступа пользователю или группе пользователей;

Принцип обеспечения безопасности вычислений

- средства учета и наблюдения (auditing), обеспечивающие возможность обнаружить и зафиксировать важные события, связанные с безопасностью, или любые попытки создать, получить доступ или удалить системные ресурсы;
- защита памяти, заключающаяся в том, что память инициализируется перед тем, как повторно используется.

- Системы уровня В основаны на помеченных данных и распределении пользователей по категориям, то есть реализуют мандатный контроль доступа. Каждому пользователю присваивается рейтинг защиты, и он может получать доступ к данным только в соответствии с этим рейтингом. Этот уровень в отличие от уровня С защищает систему от ошибочного поведения пользователя.

Принцип обеспечения безопасности вычислений

- Уровень А является самым высоким уровнем безопасности, он требует в дополнение ко всем требованиям уровня В выполнения формального, математически обоснованного доказательства соответствия системы требованиям безопасности.
- А-уровень безопасности занимает своими управляющими механизмами до 90 % процессорного времени.
- Более безопасные системы не только снижают эффективность, но и существенно ограничивают число доступных прикладных пакетов, которые соответствующим образом могут выполняться в подобной системе. Например, для ОС Solaris (версия UNIX) есть несколько тысяч приложений, а для ее аналога В-уровня — только около ста.

Микроядерные операционные системы

- Микроядро — это минимальная стержневая часть операционной системы, служащая основой модульных и переносимых расширений.
- В микроядре содержится и исполняется минимальное количество кода, необходимое для реализации основных системных вызовов. В число этих вызовов входят передача сообщений и организация другого общения между внешними по отношению к микроядру процессами, поддержка управления прерываниями, а также ряд некоторых других функций.
- Остальные функции, обеспечиваются как модульные дополнения-процессы, взаимодействующие главным образом между собой и осуществляющие взаимодействие посредством передачи сообщений.
- Микроядро является маленьким, передающим сообщения модулем системного программного обеспечения, работающим в наиболее приоритетном состоянии компьютера и поддерживающим остальную часть операционной системы, рассматриваемую как набор серверных приложений.

Микроядерные операционные системы

- Микроядро включает только те функции, которые требуются для определения набора абстрактных сред обработки для прикладных программ и для организации совместной работы приложений в обеспечении сервисов и в действии клиентами и серверами. В результате микроядро обеспечивает только пять различных типов сервисов:
  - управление виртуальной памятью;
  - задания и потоки;
  - межпроцессные коммуникации (IPC - inter-process communication, межпроцессные коммуникации);
  - управление поддержкой ввода/вывода и прерываниями;
  - сервисы набора хоста (host - главный компьютер. Сейчас этим термином обозначают любой компьютер, имеющий IP-адрес) и процессора.

Микроядерные операционные системы

- Наиболее ярким представителем микроядерных ОС является ОС реального времени QNX. Микроядро QNX поддерживает только планирование и диспетчеризацию процессов, взаимодействие процессов, обработку прерываний и сетевые службы нижнего уровня. Микроядро может быть целиком размещено во внутреннем кэше даже таких процессоров, как Intel 486. Разные версии этой ОС имели и различные объемы ядер — от 8 до 46 Кбайт.
- Чтобы построить минимальную систему QNX, требуется добавить к микроядру менеджер процессов, который создает процессы, управляет процессами и памятью процессов.
- Чтобы ОС QNX была применима не только во встроенных и бездисковых системах, нужно добавить файловую систему и менеджер устройств.

Монолитные операционные системы

- В монолитной ОС, несмотря на ее возможную сильную структуризацию, очень трудно удалить один из уровней многоуровневой модульной структуры. Добавление новых функций и изменение существующих для монолитных ОС требует очень хорошего знания всей архитектуры ОС и чрезвычайно больших усилий.
- При поддержке монолитных ОС возникает ряд проблем, связанных с тем, что все функции макроядра работают в едином адресном пространстве:
  1. это опасность возникновения конфликта между различными частями ядра;
  2. сложность подключения к ядру новых драйверов.

ОС реального времени

- Система реального времени (СРВ) должна давать отклик на любые непредсказуемые внешние воздействия в течение предсказуемого интервала времени. Для этого должны быть обеспечены следующие свойства:
  - *Ограничение времени отклика,* то есть после наступления события реакция на него гарантированно последует до предустановленного крайнего срока. Отсутствие такого ограничения рассматривается как серьезный недостаток программного обеспечения.
  - *Одновременность обработки:* даже если наступает более одного события одновременно, все временные ограничения для всех событий должны быть выдержаны. Это означает, что системе реального времени должен быть присущ параллелизм. Параллелизм достигается использованием нескольких процессоров в системе и/или многозадачного подхода.

ОС реального времени

- Различают системы «мягкого» и «жесткого» реального времени. Различие между жесткой и мягкой СРВ зависит от требований к системе — система считается жесткой, если «нарушение временных ограничений не допустимо», и мягкой, если «нарушение временных ограничений нежелательно».
- Нет мягких или жестких ОСРВ. ОСРВ может только служить основой для построения мягкой или жесткой СРВ. Сама по себе ОСРВ не препятствует тому, что ваша СРВ будет мягкой.

ОС реального времени

- Перечислим основные требования к ОСРВ:
  - Мультипрограммность и многозадачность
  - Наличие приоритетов задач (потоков)
  - Наследование приоритетов
  - Синхронизация процессов и задач
  - Предсказуемость

Мультипрограммность и многозадачность ОСРВ

- ОС должна быть мультипрограммной и многозадачной (многопоточной — multi-threaded) по принципу абсолютного приоритета (прерываемой) и активно использовать прерывания для диспетчеризации.
- Планировщик должен иметь возможность прервать любой поток и предоставить ресурс тому потоку, которому он более необходим.
- ОС (и аппаратура) должны также обеспечивать прерывания на уровне обработки прерываний.

Наличие приоритетов задач (потоков) в ОСРВ

- В ОС должно существовать понятие приоритета потока. В идеальной ситуации ОСРВ отдает ресурс потоку или драйверу с ближайшим крайним сроком (это называется управлением временным ограничением, deadline driven OS). Чтобы реализовать это временное ограничение, ОС должна знать сколько времени требуется каждому из выполняющихся потоков для завершения.
- ОС, построенных по этому принципу, практически нет, так как он слишком сложен для реализации.

Наследование приоритетов в ОСРВ

- ОС должна существовать система наследования приоритетов.
- Комбинация приоритетов тредов и разделения ресурсов между ними приводит к проблеме инверсии приоритетов. Это можно проиллюстрировать на примере, когда есть как минимум три треда. Когда тред низшего приоритета захватил ресурс, разделяемый с тредом высшего приоритета, и начал выполняться поток среднего приоритета, выполнение треда высшего приоритета будет приостановлено, пока не освободится ресурс и не отработает тред среднего приоритета. В этой ситуации время, необходимое для завершения треда высшего приоритета, зависит от нижних приоритетных уровней, — это и есть инверсия приоритетов.

Наследование приоритетов в ОСРВ

- Чтобы устранить такие инверсии, ОСРВ должна допускать наследование приоритета, то есть повышение уровня приоритета треда до уровня треда, который его вызывает. Наследование означает, что блокирующий ресурс тред наследует приоритет треда, который он блокирует (разумеется, это справедливо лишь в том случае, если блокируемый тред имеет более высокий приоритет).

Синхронизация процессов и задач в ОСРВ

- ОС должна обеспечивать мощные, надежные и удобные механизмы синхронизации задач.
- Так как задачи разделяют данные (ресурсы) и должны сообщаться друг с другом, должны существовать механизмы блокирования и коммуникации. Необходимы механизмы, гарантированно предоставляющие возможность параллельно выполняющимся задачам и процессам оперативно обмениваться сообщениями и синхросигналами.
- Эти системные механизмы должны быть всегда доступны процессам, требующим реального времени. Следовательно, системные ресурсы для их функционирования должны быть распределены заранее.

Предсказуемость ОСРВ

- Поведение ОС должно быть известно и достаточно точно прогнозируемо.
- Времена выполнения системных вызовов и временные характеристики поведения системы в различных обстоятельствах должны быть известны разработчику. Поэтому создатель ОСРВ должен приводить следующие характеристики:
  - латентную задержку прерывания (то есть время от момента прерывания до момента запуска задачи): она должна быть предсказуема и согласована с требованиями приложения. Эта величина зависит от числа одновременно «висящих» прерываний;
  - максимальное время выполнения каждого системного вызова. Оно должно быть предсказуемо и не зависимо от числа объектов в системе;
  - максимальное время маскирования прерываний драйверами и ОС.

СПАСИБО ЗА ВНИМАНИЕ