СТРУКТУРА И ПРИНЦИПЫ ПОСТРОЕНИЯ ОПЕРАЦИОННЫХ СИСТЕМ

2.2.1 Особенности методов построения ОС

При описании операционной системы часто указываются особенности ее структурной организации и основные концепции, положенные в ее основу. К таким базовым концепциям относятся:

- Способы построения ядра системы – монолитное ядро или микроядерный подход.

- Построение ОС на базе объектно-ориентированного подхода;
- Наличие нескольких прикладных сред;
- Распределенная организация операционной системы.

2.2.2 Монолитное ядро и микроядро

Большинство ОС использует монолитное ядро, которое компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения из привилегированного режима в пользовательский и наоборот.

Альтернативой является построение ОС на базе микроядра, работающего также в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой, в то время как функции ОС более высокого уровня выполняют специализированные компоненты ОС – серверы, работающие в пользовательском режиме.

При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, зато система получается более гибкой – ее функции можно наращивать, модифицировать или сужать, добавляя, модифицируя или исключая серверы пользовательского режима. Кроме того, серверы хорошо защищены друг от друга, как и любые пользовательские процессы.

Часть ОС, работающая в привилегированном режиме и ответственная за небольшой набор системных функций (управление процессами, обработка прерываний, управление виртуальной памятью, пересылка сообщений) называется **микроядром**.

Все остальные высокоуровневые функции ядра разрабатываются в виде приложений, работающих в пользовательском режиме – **серверы ОС**.

Взаимодействие между обычными приложениями и серверами ОС осуществляется через механизм обращений. **Клиентское приложение** отправляет запрос к серверу ОС через микроядро ОС. Такой механизм обеспечивает защиту работы приложений.

Микроядерная архитектура

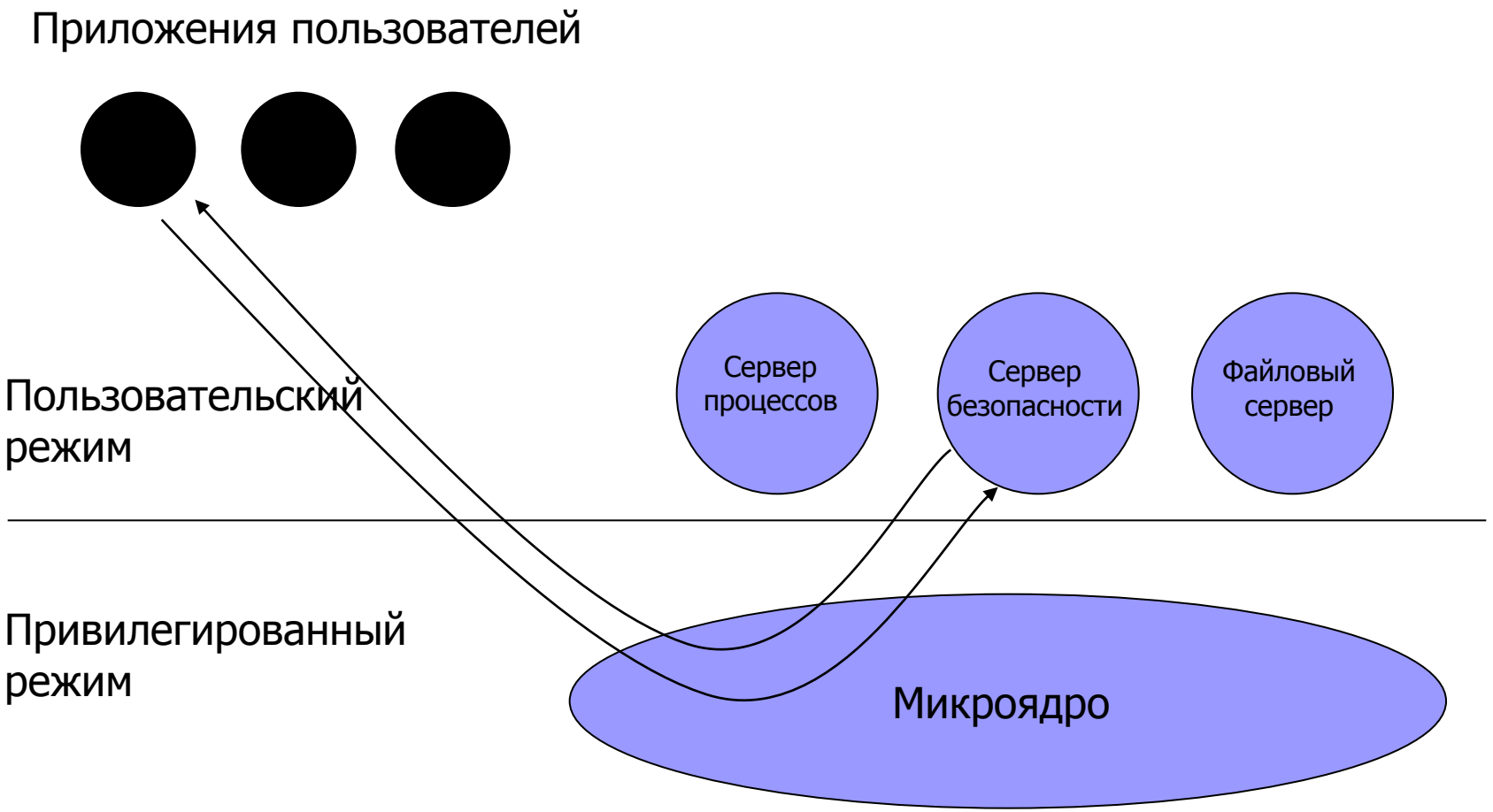


Рис. Клиент-серверная архитектура

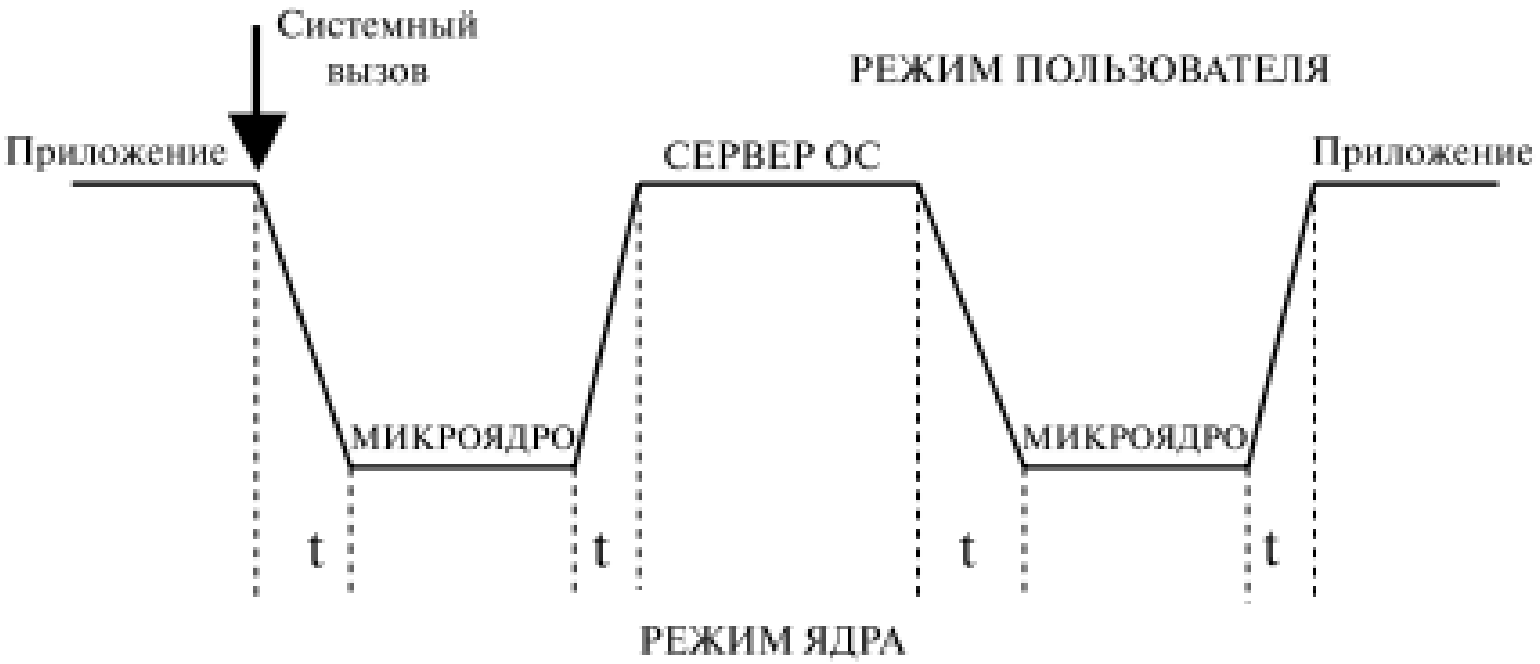


Рис. Обработка системного вызова в микроядерной архитектуре

Операционные системы, основанные на микроядерной архитектуре обладают рядом преимуществ, предъявляемых к современным ОС:

- Переносимость (обусловлена малым числом модулей в аппаратно-зависимом микроядре)
- Расширяемость (добавление новых функций связано с включением новых серверов ОС)
- Надежность (обусловлена изолированностью процессов)
- Поддержка распределенных вычислений (используется механизм взаимодействия приложений аналогичный взаимодействию в распределенных системах)

Недостаток

- Производительность (обладают меньшей производительностью)

К микроядерным ОС относится ОСРВ QNX, а к монолитным – Windows 9x и Linux. Для ОС Windows 9x пользователь не может изменить ядро, так как не располагает исходными кодами и программой сборки ядра. Для ОС Linux такая возможность предоставлена, пользователь может сам собрать ядро, включив в него необходимые программные модули и драйверы.

Ядра, которые занимают промежуточные положение между монолитными и микроядрами, называют **гибридными** (hybrid kernel).

Примеры различных типов ядер:

- монолитное ядро – MS-DOS, Linux, FreeBSD;
- микроядро – Mach, Symbian, MINIX 3;
- гибридное ядро – NetWare, BeOS, Syllable.

Для надежного управления работой приложений ядро ОС должно обладать некоторыми привилегиями по отношению к остальным приложениям.

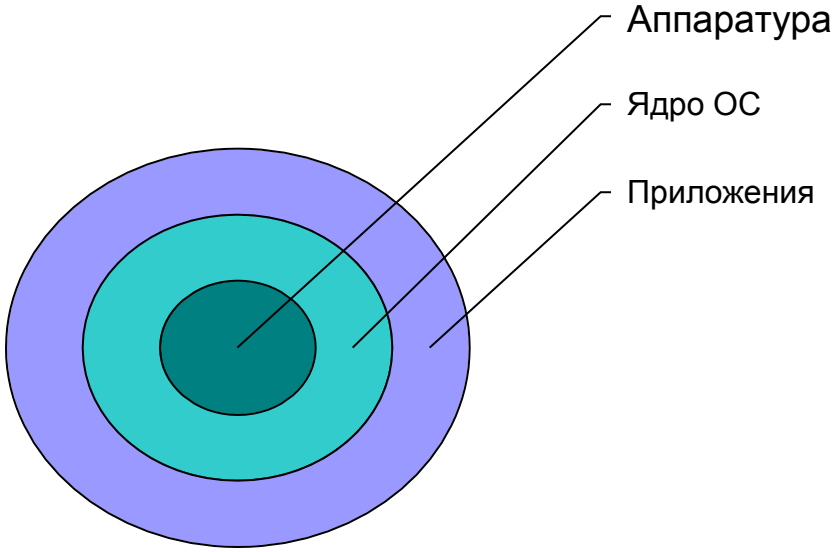
Обеспечивается привилегированный режим специальными средствами аппаратной поддержки. Процессор компьютера поддерживает как минимум два режима работы – **пользовательский** (user mode) и **привилегированный** (kernel mode).

Приложения в пользовательском режиме не могут выполнять некоторые критичные команды (переключение процессора с задачи на задачу, доступ к механизму выделения и защиты областей памяти и т.п.).

Многослойная структура (состав) ОС

Вычислительную систему под управлением ОС можно рассматривать как состоящую из нескольких слоев:

- Нижний слой – аппаратура;
- Средний – ядро ОС;
- Верхний – утилиты, приложения и т.п.



При такой организации операционной системы приложения не могут непосредственно взаимодействовать с аппаратурой, а только через слой ядра.

Обычно пользователь вычислительной системы работает только с верхнеуровневым слоем операционной системы. Программисты же используют более глубокие уровни операционной системы, например системные библиотеки, а иногда и даже низкоуровневые функции ядра (например, через низкоуровневые драйверы).

Рис. Структура ядра ОС

Ядро может состоять из следующих слоев.

- **Средства аппаратной поддержки операционной системы**.

К операционной системе относят не все аппаратные устройства компьютера, а только те, которые прямо участвуют в организации вычислительных процессов: средства поддержки привилегированного режима, систему прерываний, средства переключения контекстов процессов, средства защиты областей памяти.

- **Машинно-зависимые компоненты ОС**.

Этот слой образуют программные модули, в которых отражается специфика аппаратной платформы компьютера. Этот слой экранирует вышележащие слои ядра от особенностей аппаратуры.

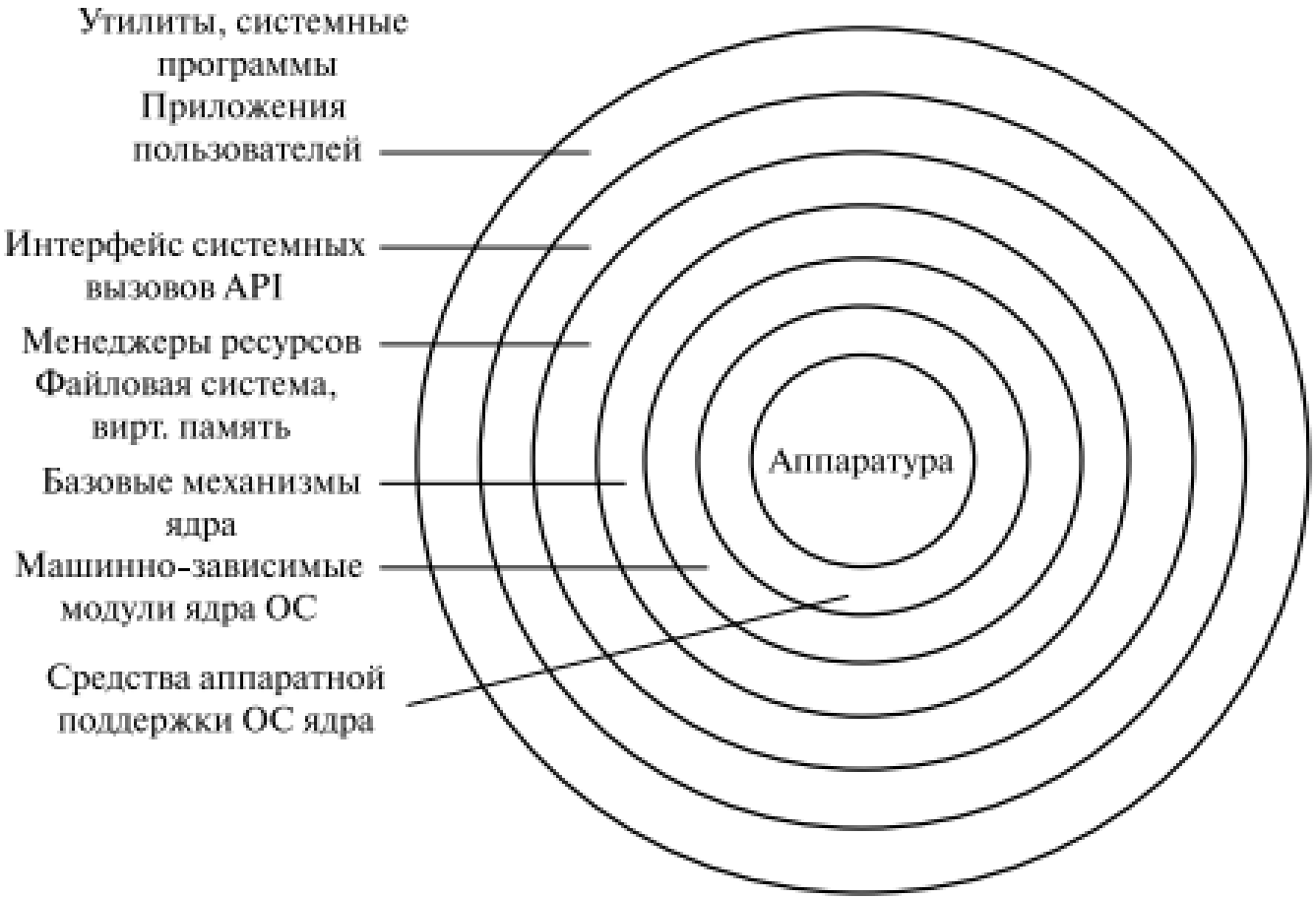
Это позволяет разрабатывать вышележащие слои на основе машинно-независимых модулей, существующих в единственном экземпляре для всех типов аппаратных платформ, поддерживаемых данной операционной системой. Примером экранирующего слоя может служить слой HAL ОС Windows .

- **Базовые механизмы ядра**.

Этот слой выполняет операции ядра, такие как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно. Модули данного слоя не принимают решений о распределении ресурсов — они только отрабатывают принятые «наверху» решения.

- **Менеджеры ресурсов**. Этот слой состоит из мощных функциональных модулей, реализующих стратегические задачи по управлению основными ресурсами вычислительной системы. Обычно на данном слое работают менеджеры (так называемые диспетчеры): менеджеры процессов, менеджеры 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.

- **Интерфейс системных вызовов.** Этот слой является самым верхним слоем ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы. Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей, их физического расположения. Например, в операционной системе UNIX с помощью системного вызова open приложение открывает некий файл, хранящийся в каталоге, а с помощью системного вызова read читает из этого файла в область своего адресного пространства некоторое количество байт.


Утилиты, системные программы
Приложения пользователей
Интерфейс системных вызовов API
Менеджеры ресурсов
Файловая система, вирт. память
Базовые механизмы ядра
Машинно-зависимые модули ядра ОС
Средства аппаратной поддержки ОС ядра
Аппаратура


РЕЖИМ ЯДРА
Утилиты ОС, приложения
Интерфейс системы ввода-вывода
Менеджер виртуальной памяти
Менеджер процессов
Базовые механизмы ядра
Машинно-зависимые модули
Средства аппаратной поддержки ОС
Аппаратура
Пользовательский режим
Приложения пользователей
Утилиты
API
Сервер безопасности
Файловая система
Драйвер устройств
МИКРОЯДРО
(режим ядра)
Средства аппаратной поддержки ОС
Аппаратура

Структура ОС носит модульный характер. Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:

- ядро – модули, выполняющие основные функции ОС;
- модули, выполняющие вспомогательные функции ОС.

Модули ядра ОС выполняют следующие базовые функции ОС: управление процессами, управление памятью, управление устройствами ввода-вывода. Функции выполняемые ядром ОС требуют высокой скорости выполнения и для этого размещаются постоянно в оперативной памяти (резидентные модули).

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**.

Резидентный модуль хранится в оперативной памяти постоянно после загрузки операционной системы.

В состав ядра входят функции, **решающие внутрисистемные задачи** организации вычислительного процесса (загрузка/выгрузка станиц из памяти, обработка прерываний). Эти функции недоступны для приложений.

Другой класс функций ядра служит для поддержки самих приложений, создавая для них прикладную программную среду. Приложения могут обращаться к ядру с запросами — **системными вызовами** — для выполнения тех или иных действий, например для открытия и чтения файла, вывода графической информации на дисплей. Функции ядра, которые могут вызываться приложениями, образуют **интерфейс прикладного программирования**, так называемый API.

Вспомогательные модули выполняют функции: архивирование информации; дефрагментация данных на диске; поиск необходимого файла и т.п. Вспомогательные модули операционной системы оформляются либо в виде приложений, либо в виде библиотек процедур.

Вспомогательные модули ОС условно разделяются на следующие группы:

- **утилиты** – приложения, решающие отдельные задачи управления и сопровождения ОС;

Утилиты — обслуживающие программы, которые предоставляют пользователю сервисные услуги.

Они, как правило, имеют полноэкранный, организованный в виде меню интерфейс взаимодействия с пользователем. Реже интерфейс организован в виде запросов.

- **системные обрабатывающие программы** – текстовые и графические редакторы, компиляторы, компоновщики и т.п.;
- **программы предоставления пользователю дополнительных услуг** – специальный вариант пользовательского интерфейса, калькулятор, игры и т.п.;
- **библиотеки процедур** – модули различного назначения, упрощающие разработку приложений.

Как и обычные приложения, для выполнения своих задач утилиты, обрабатывающие программы и библиотеки ОС, обращаются к функциям ядра посредством системных вызовов.

Модули ОС, оформленные в виде утилит, системных обрабатывающих программ и библиотек, обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются **транзитными.** Постоянно в оперативной памяти располагаются только самые необходимые коды ОС, составляющие ее ядро. Такая организация ОС экономит оперативную память компьютера.

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

Принципы построения ОС

Принцип функциональной избирательности

Некоторая часть модулей ОС должна находиться в памяти для более эффективной организации вычислительного процесса. Они называются ядром ОС, должны быть минимальными по требуемому объему памяти и наиболее часто используемым функциям. Сюда, как правило, относят модули управления системой прерывания, управления задачами, модули управления ресурсами. Остальные модули ОС называются транзитными. Они загружаются в память по необходимости, а при отсутствии свободного пространства памяти замещаются другими, более необходимыми в данный момент.

Принцип генерируемости

Суть его в том, что способ исходного представления ядра ОС должен позволять настройку его на непосредственную конфигурацию вычислительного комплекса, где ОС устанавливается на круг решаемых задач.

В большинстве современных ОС для персональных компьютеров конфигурирование под имеющийся состав оборудования производится на этапе инсталляции, а последующие изменения параметров ОС или состава драйверов производится редактированием файла конфигурации.

Единственной ОС, генерируемой в полном смысле является ОС Linux.

Принцип функциональной избыточности

Он учитывает возможность выполнения одной и той же работы различными средствами. Наличие возможности использования нескольких типов мониторов, систем управления файлами и т.д., позволяет быстро и адекватно адаптировать ОС к данной конфигурации вычислительной системы, эффективно загружать технические средства, получать максимальную производительность.

Принцип виртуализации

Он позволяет представить структуру системы в виде набора планировщиков процессов и распределителей ресурсов (мониторов) и использовать единую схему распределения ресурсов.

Наиболее полно принцип проявляется в понятии виртуальной машины, обладающей идеальными для пользователя архитектурными характеристиками.

Одним из примеров применения принципа виртуальности является имеющаяся во всех ОС Windows, защищенная подсистема, предоставляющая полную MS-DOS среду и консоль для выполнения DOS приложений.

Принцип независимости программ от внешних устройств

Суть его в том, что связь программы с внешним устройством устанавливается не на этапе трансляции, а в период планирования ее исполнения. Программа общается не с устройствами, а с ОС, сообщая ей о потребности в ресурсах для выполнения данной работы. Конкретное устройство, на котором эта работа будет выполнена, программу не интересует, это задача ОС.

Принцип совместимости

Суть принципа − в обеспечении совместимости ОС выполнять программы, написанные для других ОС или под другие аппаратные платформы.

Различают совместимость на уровне исходных текстов (текстовая совместимость) и на уровне кодов (двоичная совместимость).

Первая требует наличия транслятора, совместимость на уровне библиотек и системных вызовов.

Вторая требует совместимость на уровне архитектуры процессов и систем команд. Для реализации такой совместимости используются эмуляторы (прикладные среды).

Принцип открытости и наращиваемости

Открытая ОС доступна для анализа пользователем и системным программистом. Наращиваемая ОС позволяет вводить в состав новые модули, модернизировать существующие и т. д., не нарушая целостность системы. Пример наращиваемости демонстрируют структурированные ОС типа «клиент-сервер» на основе микроядерной технологии, когда ядро системы (привилегированная управляющая программа) сохраняется неизменяемым, а состав серверов (набор непривилегированных услуг) может модифицироваться. Примером открытой системы является ОС Linux и все UNIX-системы.

Принцип мобильности (переносимости)

Принцип требует, чтобы ОС легко устанавливалась с одного процессора на другой, с одной аппаратной платформы на другую. Для этого ОС в основном должна быть написана на языке, имеющимся на всех платформах, куда ее планируется переносить (предпочтительно на С). Кроме того, в ней должны быть минимизированы средства взаимодействия с аппаратурой. Не исключенные аппаратно зависимые части кода должны быть изолированы в хорошо локализируемых модулях. Тогда при переносе меняются (или подстраиваются) только эти локальные данные и функции взаимодействия с ними.

Принцип безопасности вычислений

Правила безопасности защищают ресурсы одного пользователя от другого и устанавливают квоты на ресурсы для предотвращения захвата всех ресурсов одним пользователем.

Основы стандартов в области безопасности вычислений были заложены в документе под названием «Критерии оценки наземных компьютерных систем». В соответствии с ним, безопасной считается система, посредством специальных механизмов контролирующая доступ к информации так, что доступ к ней получают только лица с соответствующими полномочиями или процессы, выполняющиеся от их имени.

В соответствии с требованиями **«Оранжевой книги»** безопасной считается такая 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 «Оранжевой книге», выделяет четыре уровня (D, C, B, A) и 6 классов безопасности внутри уровней (C1, C2, B1, B2, B3, A1).

В уровень D попадают системы, оценка которых выявила их несоответствие требованиям безопасности. Уровень С обеспечивает произвольное управление доступом; уровень В – принудительное управление доступом; уровень А – верифицируемую безопасность. В каждом классе от С1 к А1 требования по безопасности расширяются.

Коммерческие системы обычно соответствуют уровню С. **Требования безопасности этого уровня:** 1) наличие защищенных средств секретного входа, обеспечивающих идентификацию пользователя путем ввода логина и пароля; 2) избирательный (дискреционный) контроль доступа, позволяющий владельцу ресурса определить, кто имеет доступ к ресурсу и что он может с ним делать (права доступа пользователю или группе пользователей); 3) средства учета и наблюдения, обеспечивающие возможность обнаружить и зафиксировать события (попытки создать, получить доступ и удалить системные ресурсы; 4) защита памяти, обеспечивающая инициализацию перед повторным использованием.

Системы уровня В реализуют мандатный (принудительный) контроль доступа. Каждому пользователю присваивается рейтинг защиты и он может получать доступ к данным только в соответствии с этим рейтингом.

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.

2.3 Вопросы к лекции

- 2.3.1 Каковы базовые концепции, положенные в основу ОС?
- 2.3.2 Что характерно для монолитного ядра и микроядра?
- 2.3.3 Охарактеризуйте работу ядра в привилегированном режиме.
- 2.3.4 Из каких модулей состоят ОС?
- 2.3.5 Охарактеризуйте принципы построения ОС.