

Учебный курс

Операционные среды, системы и оболочки

Лекция 2

Лекции читает

доктор технических наук, профессор
Назаров Станислав Викторович

1.3. Назначение, состав и функции ОС

Назначение

1. Обеспечение удобного интерфейса [приложения, пользователь] - компьютер за счет предоставляемых сервисов:

- 1.1. Инструменты для разработки программ
- 1.2. Автоматизация исполнения программ
- 1.3. Единообразный интерфейс доступа к устройствам ввода-вывода
- 1.4. Контролируемый доступ к файлам
- 1.5. Управление доступом к совместно используемой ЭВМ и ее ресурсам
- 1.6. Обнаружение ошибок и их обработка
- 1.7. Учет использования ресурсов

2. Организация эффективного использования ресурсов ЭВМ

- 2.1. Планирование использования ресурса
- 2.2. Удовлетворение запросов на ресурсы
- 2.3. Отслеживание состояния и учет использования ресурса
- 2.4. Разрешение конфликтов между процессами, претендующими на одни и те же ресурсы

3. Облегчение процессов эксплуатации аппаратных и программных средств вычислительной системы

3.1. Широкий набор служебных программ (утилит), обеспечивающих резервное копирование, архивацию данных, проверку, очистку, дефрагментацию дисковых устройств и др.

3.2. Средства диагностики и восстановления работоспособности вычислительной системы и операционной системы:

- диагностические программы для выявления ошибок в конфигурации ОС;**
- средства восстановления последней работоспособной конфигурации;**
- средства восстановления поврежденных и пропавших системных файлов и др.**

4. Возможность развития

4.1. Обновление и возникновение новых видов аппаратного обеспечения

4.2. Новые сервисы

4.3. Исправления (обнаружение программных ошибок)

4.4. Новые версии и редакции ОС

Состав компонентов и функции операционной системы:

1. Управление процессами
2. Управление памятью
3. Управление файлами
4. Управление внешними устройствами
5. Защита данных
6. Администрирование
7. Интерфейс прикладного программирования
8. Пользовательский интерфейс

1.4. Архитектуры операционных систем

Архитектура - это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы. [IEEE 1471]

ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ АРХИТЕКТУРЫ ОПЕРАЦИОННЫХ СИСТЕМ:

1. Концепция многоуровневой иерархической вычислительной системы (виртуальной машины) с ОС многослойной структуры.
2. Разделение модулей ОС по функциям на две группы: ядро – модули, выполняющие основные функции ОС, и модули, выполняющие остальные (вспомогательные) функции.
3. Разделение модулей ОС по размещению в памяти вычислительной системы: резидентные, постоянно находящиеся в оперативной памяти, и транзитные, загружаемые в оперативную память только на время выполнения своих функций.
4. Реализация двух режимов работы вычислительной системы: привилегированного режима (режима ядра – kernel mode) или режима супервизора (supervisor) и пользовательского режима (user mode) или режима задача (task mode).
5. Ограничение функций ядра (а, следовательно и числа его модулей) до минимально необходимых функций.

6. Модульное строение (однократно используемые – при загрузке ОС) и повторно используемые (привилегированные – не допускают прерываний, реентерабельные – допускают прерывания и повторный запуск, повторновходимые – допускают прерывания после завершения секций).
7. Параметрическая универсальность. Возможность генерации ОС и создания нескольких рабочих конфигураций.
8. Функциональная избыточность.
9. Функциональная избирательность.
10. Открытость, модифицируемость, расширяемость (возможность получения текстов исходных модулей).
11. Мобильность – возможность переноса на различные аппаратные платформы.
12. Совместимость – возможность выполнения приложений, рассчитанных на другие ОС.
13. Безопасность – защита от несанкционированного доступа, защита легальных пользователей друг от друга, аудит, возможность восстановления ОС после сбоев и отказов.

Модульно – интерфейсный подход (структурный подход)

1. Декомпозиция системы на модули по структурному или функциональному признаку.
2. Модули и их взаимные связи образуют абстракцию системы высокого уровня.
3. Описывается каждый модуль и определяется его интерфейс.
4. Проводится декомпозиция каждого модуля и т. д.

Спецификации модулей и их интерфейсов дают структурную основу для проектирования каждого модуля и всей системы в целом.

Правильное определение и выделение модулей представляет собой сложную задачу. Тесно связанные между собой части системы должны входить в один и тот же модуль.

Разработчики программного обеспечения начинают работу с очень грубого и неполного наброска схемы системы и преждевременно обращают внимание на детали отдельных модулей. Поэтому решения, влияющие на систему глобальным образом, принимаются не из тех предпосылок, из которых нужно и без ясного понимания их последствий.

Преждевременная реализация приводит к неустойчивости программного обеспечения, которая часто требует огромных усилий по поддержанию системы.

Многослойная (иерархическая) структура операционной системы и метод проектирования «сверху вниз» и «снизу вверх»

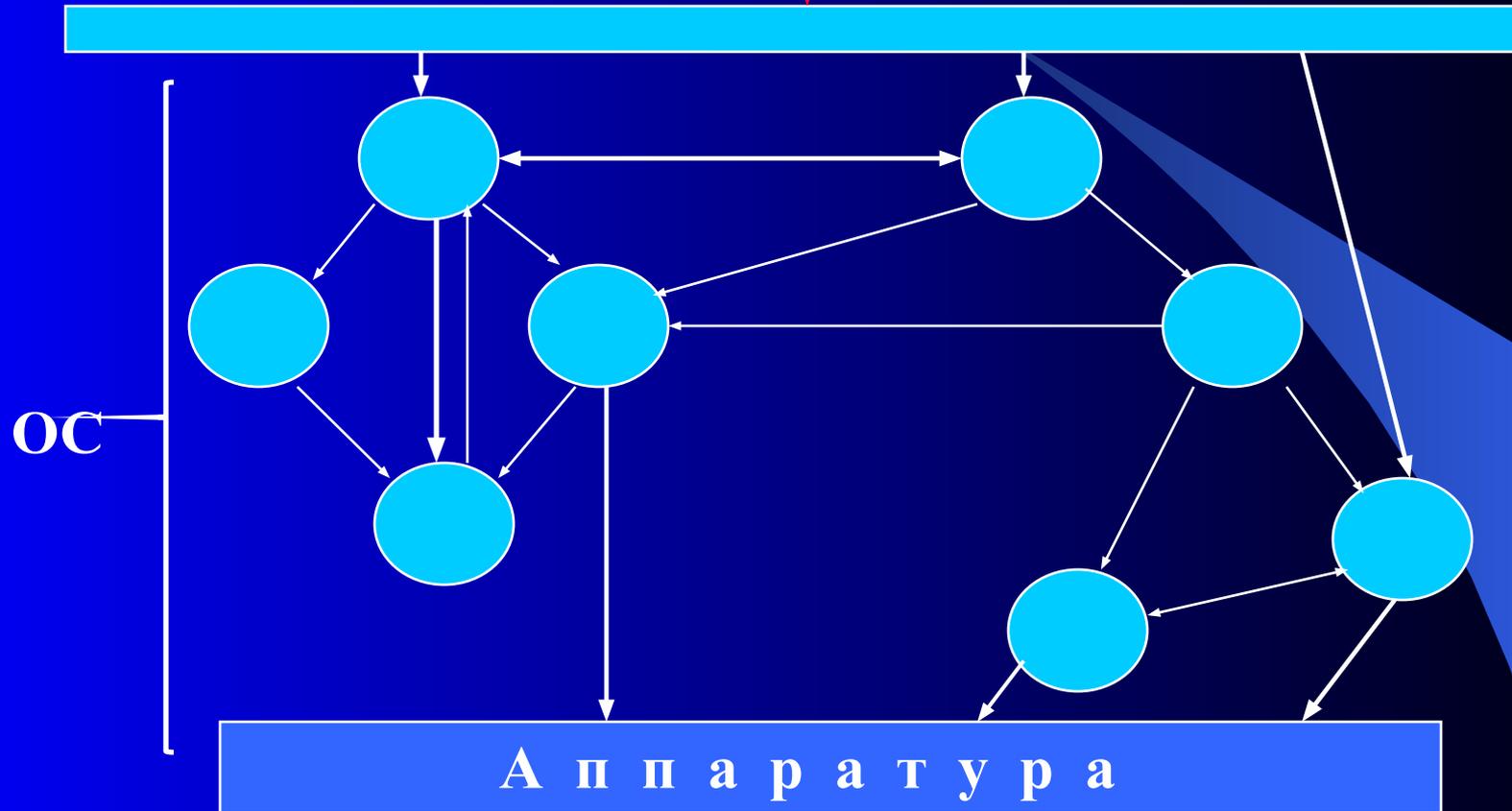
1. Операционная система представляется в виде иерархии слоев.
2. Верхний слой определяет виртуальную машину с желаемыми свойствами.
3. Каждый следующий слой детализирует вышележащий, выполняя для него некоторый набор функций.
4. Межслойные интерфейсы подчиняются строгим правилам. Связи внутри слоя могут быть произвольными.
5. Отдельный модуль слоя $L(i)$ может выполнить работу самостоятельно или последующим вариантам: обратиться только к слою $L(i-1)$; обратиться к некоторой команде определенного слоя $L(q)$, который выполняет требуемую функцию ($i-2 \leq q \leq 0$); обратиться к любому последующему слою $L(s)$, ($i-2 \leq s \leq 0$).

Достоинства:

1. Между уровнями можно организовать четкий интерфейс.
2. Систему можно спроектировать методом «сверху вниз», а реализовать методом «снизу вверх».
3. Уровни реализуются в соответствии с их порядком, начиная с аппаратуры и далее вверх.
4. Каждую новую виртуальную машину можно детально проверить, после чего продолжать дальнейшую работу.
5. Любой слой достаточно просто модифицировать, не затрагивая другие слои и не меняя межслойные интерфейсы.

Монолитная архитектура операционной системы

От приложений
системный интерфейс



Пример: ранние версии ядра UNIX, Novell NetWare. Каждая процедура имеет хорошо определенный интерфейс в терминах параметров и результатов и может любую другую для выполнения нужной работы.

Смена режимов при выполнении вызова функции ядра



Недостатки иерархической организации ОС:

1. Значительные изменения одного из уровней могут иметь трудно предвидимое влияние на смежные уровни.
2. Многочисленные взаимодействия между соседними уровнями усложняют обеспечение безопасности.

Микроядерная архитектура ОС

