**Классификация и архитектура ОС, процессы и их состояния**

Файл – объект, предназначенный для хранения информации.

Процессы и нити (потоки)

**Классификация ОС**

1. Многозадачность:

* Одна задача (MS-Dos)
* Много задач (Windows, Linux, OS2, UNIX)

Многозадачные ОС решают задачи распределения ресурсов и конкуренции. Идея разделения времени (выделение каждому процессору часть процессорного времени)

Классическая машина – одноядерная (пока что)

1. Поддержка многопользовательского режима:

* Однопользовательские (пользователь один, вся информация принадлежит ему, не нужно защищать информацию от кого-либо)
* Многопользовательские (много пользователей, присутствует необходимость в безопасности информации)

1. Многопроцессорная обработка:

* Симметричные (все ядра под управлением 1 ядра, процесс может выполняться на одном из ядер)
* Ассиметричные (ядра неравноправные; есть 1 главное ядро, а остальные являются подчинёнными, находятся под его управлением; главное ядро управляет загрузкой всех остальных ядер)

ОС реального времени. Система реального времени – особый класс систем, в котором регламентируется время отклика (обработки) системы, вне зависимости от их количества.

**Архитектурная особенность ОС**

1. Монолитное ядро (все компоненты ОС являются не самостоятельными модулями, а составными частями одной большой программы)

ОС представляет собой набор процедур, каждая из которых может вызывать каждую (обращаться напрямую) Они используют общую структуру данных и взаимодействуют друг с другом. Ядро ОС совпадает со всей ОС. Для таких ОС присуще понятие «Сборка ядра»

Компьютер состоит из разных компонентов, каждый из которых может работать по-своему. Поэтому перед установкой ОС выбирается набор компонентов, из которых состоит компьютер, и из них происходит сборка ядра. При замене деталей делается пересборка. Монолитное ядро – старейший способ построения ОС.   
Плюсы: быстрота работы   
Минусы: Сложность разработки и проектировки.

1. Многоуровневая система

Структурируя такую систему, можно разбить её на ряд уровней с хорошо определёнными связями между ними. Таким образом, что бы объекты уровня N могли вызывать только объекты уровня N-1. Нижний уровень – аппаратура, верхний – интерфейсы пользователя.

Пример уровневой системы (снизу вверх):  
Аппаратная составляющая, планирование задачи и процессов, управление памятью, драйвер устройства связи оператора и консоли, управление вводом-вводом, интерфейс пользователя.

Другим примером (не ОС) является модель сетей с 7 уровнями. Данные системы хорошо реализуются, т.к. разрабатывать их можно постоянно, (каждый слой имеет свою структуру). Они хорошо тестируются, но сложно проектируются (изначальное разбиение на слои, проектирование и т.п.) Данная система менее эффективна, чем монолитное ядро, т.к. информация идёт по слоям => более низкая скорость обработки.   
Плюсы: Хорошая реализация за счёт постоянной разработки, хорошее тестирование.  
Минусы: Сложная проектировка, низкая скорость обработки

1. Виртуальные машины

ВМ представляют собой 1 физический компьютер, который разбивается на несколько ВМ, каждая из которых представляет собой «новый, чистый» компьютер. У каждой такой машины есть свой процессор, память, устройства ввода/вывода и т.п. Но физически машина одна.   
Плюсы: на каждую такую машину можно поставить разные ОС и работать с ними.   
Минусы: небольшая производительность машин.

1. Микроядерная архитектура

В последние десятилетия большая часть системного кода переносится на уровень пользователя. Как следствие – минимизация ядра ОС. Большинство компонентов ОС – отдельные самостоятельные программы, которые взаимодействуют между собой через специальный модуль (микроядро). Только микроядро работает в привилегированном режиме, обеспечивая взаимодействие между всеми программами.   
Плюсы: высокая степень модульности ядра ОС, что позволяет легко добавлять новые компоненты в систему. Существенно упрощается отладка систем и отказоустойчивость. Компоненты ОС ничем не отличаются от разработки обычных программ.   
Минусы: накладные расходы (компоненты взаимодействуют через ядро, а не напрямую друг с другом)   
Для того, чтобы минусы были минимальными, требуется тщательное проектирование системы (так, чтобы взаимодействия между ними были минимальными)

1. Смешанные системы

Большинство ОС – комбинированные (Linux – монолитная система с элементами микроядерной архитектуры; Windows – микроядерная архитектура и монолитное ядро)

**Процессы**

Процесс – это некоторая совокупность набора исполняющихся команд, ассоциированных с этим процессом ресурсов (адресное пространство, стеки, открытые файлы, зарезервированный устройства ввода\вывода и т.д.) и текущего момента его выполнения (значение регистров, значение счётчиков команд, текущее состояние содержимого его стека, текущее значение его переменных), находящихся под управлением ОС.

Процесс – динамическая программа.

Не существует однозначного взаимосоответствия между процессами и программами, которые обрабатываются в ВС.

**Состояния процесса**

Исполняется/не исполняется (работает/не работает; активный/не активный)

* Процесс может остановиться по 2-м причинам:  
  а) Для его дальнейшей работы потребовалось некоторое событие (завершение операции ввода/вывода)  
  б) Истёк квант времени, выделенный ему ОС
* Не исполняется в двух состояниях:  
  а) ожидание  
  б) готовность
* Выйти из состояния исполнения может по 3-м причинам:  
  а) ОС прекращает его деятельность  
  б) процесс не может продолжать свою работу, пока не произойдёт некоторое событие (ОС переводит его в состояние ожидания)  
  в) приостановка (в результате возникновения прерывания в ВС (АП; самое частое – это прерывание таймера) данный процесс возвращают в состояние готовности)

В частности процесс не появляется в системе моментально, для его «нормального» проявления ему нужно пройти через особое состояние «рождение». Там он должен получить адресное пространство, ресурсы для работы, значение программного счётчика и т.д. Только потом он переходит в состояние готовности.

Процесс не может моментально завершиться, поэтому он попадает в состояние «завершения работы», когда ресурсы возвращаются системе. После этого происходит уничтожение процессов.