Что такое ОС?

Операционная система (ОС) – представляет собой основу какого-то устройства, это комплекс программ, решающих 2 важные задачи:

- интерфейса между устройствами вычислительной системы и прикладными программами.
- управления устройствами и рабочими процессами, эффективно распределяя системные ресурсы.

Устройства ввода / вывода?

+) Ввод/вывод считается одной из самых сложных областей проектирования операционных систем, в которой сложно применить общий подход из-за изобилия частных методов.

+) Сложность возникает из-за огромного числа устройств ввода/вывода разнообразной природы, которые должна поддерживать ОС. При этом перед создателями ОС встает очень непростая задача — не только обеспечить эффективное управление устройствами ввода/вывода, но и создать удобный и эффективный виртуальный интерфейс устройств ввода/вывода, позволяющий прикладным программистам просто считывать или сохранять данные, не обращая внимание на специфику устройств и проблемы распределения устройств между выполняющимися задачами. Система ввода/вывода, способная объединить в одной модели широкий набор устройств, должна быть универсальной. Она должна учитывать потребности существующих устройств, от простой мыши до клавиатур, принтеров, графических дисплеев, дисковых накопителей, компакт-дисков и даже сетей. С другой стороны, необходимо обеспечить доступ к устройствам ввода/вывода для множества параллельно выполняющихся задач, причем так, чтобы они как можно меньше мешали друг другу.

+) Поэтому самым главным является следующий принцип: любые операции по управлению вводом/выводом объявляются привилегированными и могут выполняться только кодом самой ОС.

Вычислительный процесс?

+) процесс преобразования входных данных в выходные под управлением программы, а программа работает по управлением ОС

Когда мы имеем дело с ОС?

ОС - промежуточное звено между аппаратурой и прикладным ПО

Поддержание высокоуровневого унифицированного интерфейса прикладного программирования к разнородным устройствам ввода-вывода является одной из наиболее важных задач ОС.

Здесь мы опять имеем дело со свойством операционной системы подменять реальную аппаратуру удобными для пользователя и программиста абстракциями.

Всегда ли нужна ОС?

На калькуляторе не нужна, в банкоматах своя ОС, в смарт-карте ОС для шифрования и расшифровывания данных

Зависить от цели использования и самого себя этого устройства???

Этапы развития ЭВМ

1 поколение - ламповые

2 поколение - транзисторовые (дискретные), компы меньше и оптимизированнее

- перфокарты (как на печатной машинке - выбиваем дырки)

- машину нужна настраивать перед загрузкой пользовательских перфокарты

- для этого нужны перфокарты для инициализации

3 поколение - интегральные схемы, монитор, клавиатура (консоль) - обработка нажатий клавиш

- 1 машина - много людей (чтобы комп не простаивал, пока чел "тупит")

- многозадачность = мультипрограммирование

- Unix

4 поколение - интегральные схемы высокой степени интеграции

- микропроцессоры

- эпоха ПК

- MS-DOS

Assembler, Fortran

Модульное программирование

ООП

ОС

- **общего назначения**
- **специального назначения**
  - ОС на спутнике без графического дизайна
  - Криптопроцессор в банкомате, графическая карта не нужна
  - На телефоне нужна многозадачность
  - на квадрокоптерах система реального времени

Цель ОС - предоставить пользователю возможность осуществлять решение вычислительных и задач по обработке данных

Что должен сделать проектировщик ОС прежде чем создать ОС?

- Цели и задачи определить, далее составить спецификацию

Свойства ОС

- **Надежность**
  - любую ошибку ОС должна обрабатывать (продолжить работу)
- **Защита**
  - несколько процессов, они не должны лезть в адресное пространство друг друга
  - один записывает данные, остальные должны ждать, если им необходимо манипулировать этими данными
  - в си можно читать! за границами массива
- **Предсказуемость**
  - одна и та же последовательность действий - один и тот же результат
  - у квантовых компов вероятностный результат
    - bash: cp file1 dir/dir1 - копирование файла
    - windows: левая кнопка - перетаскивание иконки - перенос (если на разных дисках)
    - копирование (на разных дисках) левая - непредсказуемвая

- **Удобство**
  - для пользователя
  - для программиста (API, абстракции, системные вызовы)
- **Эффективность**
  - Ресурсы (память, процессорное время) эффективно использовать
    - Процессорное время: Процесс на 3 часа не должен тормозить процесс на 1 секунду
    - Память: Есть много памяти, но вся дифрагментирована, нет непрерывного участка - записать данные невозможно
  - Минимальные простои
  - Минимальная фрагментация памяти
- **Общесистемные услуги**
  - Набор услуг по работе с аппаратурой
    - копирование, удаление
  - Сервисы для обслуживания функционирования системные
  - Устранение дефрагментации
- **Гибкость**
  - Разные условия, разная аппаратура
  - Генерируемость
    - новая мышка - ОС гибко настраивается (адаптируется)
      - при необходимости скачивает драйвер
- **Расширяемость**
  - Новая плата, новая функциональность
  - Установка драйверов
    - желательно все динамически (без перезагрузки ОС)
    - если ядро generic - оптимальное
    - раньше надо было ядро перекомпилировать
- **Ясность**
  - Должно быть ясно, что происходит

- Прогрессбар, когда программа долго работатает

- пользователь должен знать, что программа не зависла

- Нажимаем на иконку - запускается программа

1. Если ОС работает только с 1 программой, нужно ли защита?

- Нет второго пользовательского процесса, который бы залез в адресное пространство первого

2. Является ли оконный интерфейс современных ОС удобным?

- Нет понятия "интуитивно понятного" интерфейса, есть понятие привычного интерфейса

- Norton commander для кого-то удобнее

- Vim, emacs для кого-то удобнее функциональных текстовых редакторов

- Весь графический интерфейс можно заменить на командную строку

3. Можно ли назвать современные ОС эффективными?

- Помимо железа, на эффективность влияете оптимальность ОС

- iOS поэтому быстрее Android

Основные принципы построения ОС

1.Частотный принцип

основан на выделении в алгоритмах программ и в обрабатываемых массивах действий и данных по частоте использования. Для действий, которые наиболее часто используются в работе ОС , обеспечиваются условия их быстрого выполнения.

2.Принцип модульности.

Модуль предполагает легкий способ его замены на другой при наличии заданных интерфейсов. Важное значение при построении ОС имеют параллельно используемые или реентабельные модули. Такие модули можно использовать одновременно несколькими программами.

Зам : Тоненбаум - будущее за микроядром,
модули
- для работы с процессами
- для работы с памятью
...

3. Принцип функциональной избирательности.

В ОС выделяется некоторая часть важных модулей, которая должна быть под рукой для эффектной организации вычислительного процесса, эту часть называют ядром.

Зам : нельзя реализовать все функции разом, только самын важные

4.Принцип генерируемости. (=гибкость)

Этот принцип подразумевает такой способ исходного представления ОС, который позволял бы ее настраивать исходя из конкретной конфигурации конкр. машины и круга решаемых задач. Эту часть ОС называют процессом инсталяции.

Зам : работа с широким набором оборудования

5. Принцип функциональной избыточности.

Этот принцип учитывает возможность проведения одной и той же работы различными средствами.

Зам : ОС должна не только то что нужно, но и немного больше

6.Принцип по умолчанию. (=ясность)

Применяется для облегчения организации связей системы как на этапе генерации, так и в процессе работы ОС. Принцип основан на хранении в системе некоторых базовых описаний структур процессов, структур

модулей, конфигураций оборудования и данных, определяющих прогнозируемые объемы оперативной памяти, времени счета программ и т.д., которые характеризуют пользовательские программы и условия их выполнения.

Зам : Переносим файл (папку) - копируется файл (папка) - одинаковый

7.Принцип перемещаемости.

Этот принцип подразумевает построение модулей, исполнение которых не зависит от места расположения в оперативной памяти.

Зам : 1 программа может быть в разных областях памяти
В абсолютных адресах не адресуемся

8.Принцип защиты .

Этот принцип определяет необходимость разработки мер ограждающих программы и данные пользователей от искажений и нежелательных влияний друг на друга, а также пользователей на ОС и наоборот.

9. Принцип независимости программ от внешних устройств.

Этот принцип позволяет одинаково осуществлять операции управления внешними устройствами независимо от физических характеристик.

Зам : Unix: любое устройство - это файл
Надо передать файл: открываем сокет, пишем в него как файл
Надо получить файл: открываем сокет, читаем как из файла
У microsoft все в виде объектов

10. Принцип открытой и наращиваемой ОС. (=расширяемость)

Открытая ОС доступна для анализа пользователей. Наращиваемая ОС позволяет не только использовать возможности генерации, но и вводить в ее состав новые модули, совершенствовать существующие.

Всем ли принципам должна отвечать ОС? Какие принципы могут нарушаться?

Не все должна отвечать ОС.

Зачем нам на спутнике **генерируемость** (никто туда флешку не вставит)

Концептуальные основы ОС

- **Процесс**
- **Ресурс**
  - память
  - процессорное время
  - устройство ввода-вывода (периферийные устройства)
  - программное и математическое обеспечение
- **Виртуализация**
  - виртуальная память
  - виртуальное адресное пространство
- **Пользовательский интерфейс** (взаимодействие с пользователем)
- **распределения ресурсов**
- **прерывания**

Другие концепции

(

Перемещаемость

Реитерабильность (функция использует гл. переменную, 2 запуска - разные результаты)

Прерывания

Каналы, процессоры, устройства ввода-вывода

Компоновщики, загрузчики (загружает программу в ОЗУ)

Языки системного программирование

Микропрограммирование

Структурное и модульное программирование

)

Подсистемы работают с разными ресурами

- Менеджер памяти работает (с памятью)
- Система распределение ОЗУ
- Система управление процессами (с процессами)
- Система управление устройствами ввода-вывода
- Система управления данными

Много программ работает одновременно -
мультипрограммирование = многозадачность
(Один процессор переключается с одной программы на другую)

4 области использования ОС

- Банкомат, Вычисл. кластер, ТВ, мобильный телефон

Какие требования к каждой области?

1. Расширяемость - **для ТВ** - новые кодеки!
2. **Для банкомата**
   - **надежность**
     - пользователь не может сделать то, что не должен
     - нельзя подключить usb, клавиатура
     - наружу торчит только то, что нужно
     - у пользователя нет прав админа
   - **защита**
     - одна программа - защита не нужна
     - но вирусы нужно ограничивать
       - запуск в защищенной памяти (песочница, e.g. java машина)
   - **предсказуемость**
   - **удобство**
     - вопрос привычности
   - **эффективность**
     - должно хорошо работать на имеющемся железе
   - **общесистемные услуги**
     - для пользователя не нужны, для администратора нужны
   - **гибкость**
     - поддерживать разные картоприемники, системы выдачи денег, клавиатуры и тд
   - **расширяемость**
     - больше нет, чем да
   - **ясность**
     - да

(переплыви тихий океан! ЯСНО, но для моего набора команд НЕПОНЯТНО) - ?

пользователь - тот, кто пользуется системой
- логин, вход
- права (открыть, закрыть файл, …)
- группа пользователей (по правам) – admin, user1, user2, …

многопользовательский (2, 4) и многозадачный режим (3, 4)
1. Одна программа - один пользователь
2. Одна программа - много пользователей
3. Много программ - один пользователь
4. Много программ - много пользователей

Типы ОС – ko có chi tiết trong leksi

Универсальные ОС
- Linux, Windows, Android
- широкий спектор оборудования
- функции клиента и сервера

ОС специального назначения:
- ОС реального времени
  - используются на производстве
  - чаще всего, встраиваемые системы
  - e.g. QNX
- сетевые ОС
- защищенные системы
  - все современные ОС сетевые и защищенные
  - на военное предприятие нужно поставить специализированную систему - (более защищенную)
    - сертифицирована
    - ограниченные набор софта

ОС реального назначения

Где используется:
- автоматизация промышленных процессов
- транспорт

Реальное время:

- жесткое реальное время (hard real-time)
  - на выполнение процесса есть строгое ограничение по времени
  - если превышает, процесс уничтожается
- мягкое реальное время
  - ограничение времени с диапазоном отклонения

Вопросы:

1. Почему в мультипрограммных ОС сложно обеспечить жесткое реальное время?

- много программ - сложно расчитать время для процесса

2. Какого типа реальное время нужно использовать в ОС для беспилотного автомобиля ?

- жесткое реальное время
- т.к. Операционная система должна обеспечить требуемое время выполнения задачи реального времени даже в худших случаях

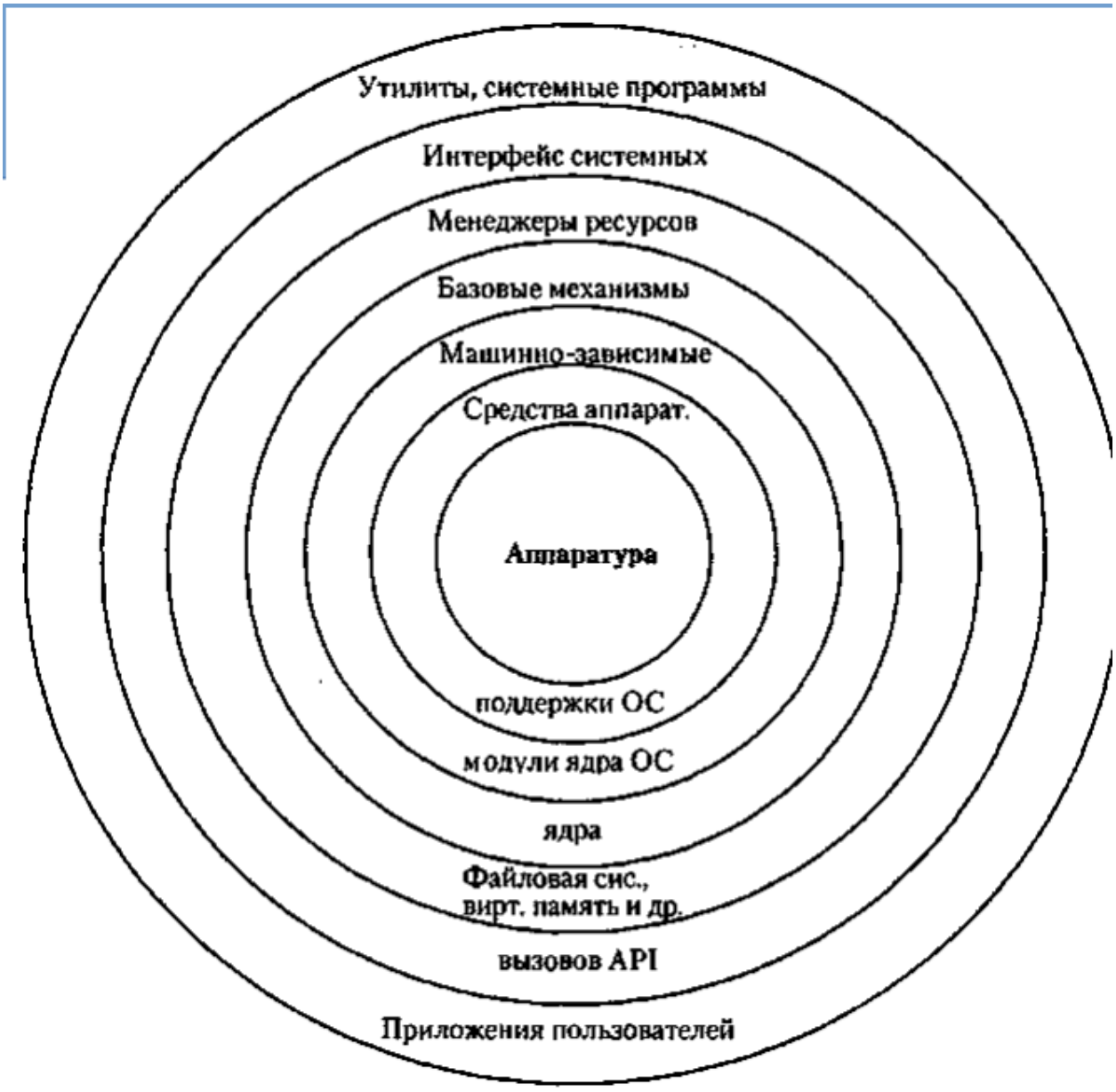
Многослойная структура ОС

- утилиты, системные программы, приложения пользователей
  - то, ради чего создана ОС
- интерфейс системных вызовов, API
  - WinAPI, либы Linux (open, fork, exec)
- менеджеры ресурсов
  - файловая система
  - вирт. память
  - ...
- базовые механизмы ядра
  - менеджер процессор, управления памяти и др
- машинно-зависимые модули ядра
  - код на ассемблере (работа с регистрами)

- средства аппаратной поддеркжи ОС
  - драйвера
  - BIOS (минимальная ОС, ввод \ вывод)

- аппаратура

- поддержки ОС
- модули ядра ОС
- Ядра
- файловая систем, виртуальный память, ...
- вызовов API
- приложения пользователей



Типовые средства аппаратной поддержки ОС

- средства поддержки привилегированного режима
- средства трансляции адресов
- средства переключения процессов
- система прерываний
- системный таймер
- средства защиты областей памяти

В чем отличие систменого таймера и системными часами?

Системный таймер зависит от частоты процессора

- разная частота - разная скорость игры

Системные часы зависят от времени на системе

Функциональные компоненты ОС (какие ресурсы, такие и функции управления)

- управление процессами
- управление памятью
- управление файлами и внешними устройствами
- защита данных и администрирование
  - *sudo apt-get ...*
    - *в логах присутствует конкретное имя пользователя*
    - *а не "root"*
- интерфейс прикладного программирования
  - *API, системные вызовы, библиотеки*
- пользовательский интерфейс

Типы пользовательских интерфейсов:

- командная строка;
- графический (оконный);

Новые интерфейсы с появлением новых устройства

- шлем который читает мысли и пишет текст
- очки которые фиксируют движения глаз и пишут статьи

Разновидности командной строки (оболочки)

- bash (born again shell)
- bshell
- cshell (си программирование)
- tshell (bshell + cshell)
- zshell (tshell + bash)
- powershell

Какие слои структуры ОС изменятся, если необходимо осуществить перенос ОС с персонального компьютера на мобильное устройство ?

- изменяются все нижние, немного затрагивает базовые механизмы ядра, остальное не изменятся
- ОС смартфона - система реального времени (фоновые приложения и звонок)

Генерация ОС - адаптация имеющийся ОС к имеющимся оборудованиям

- **Инсталляция** (установка ОС) – лучший момент для настройки ОС под конкретную вычислительную машину
- **Загрузка** – неплохой момент для настройки ОС под конкретную машину, но не лучший (потому что просканировать все оборудование каждый раз при загрузке - долго)
- **BIOS/EFI/UEFI** - базовая конфигурация
    - подпрограмма, запускается при включении компа
    - проводит тестирование системы
        - обнаруживает все устройства, определяет память

(раньше задавали размер страницы памяти, в которую может загружаться процесс - важный момент для настройки при загрузке)

Модульность - разбиение функциональных частей на модули

Уровни абстракции ОС

* **Пользовательский режим (защищен)**

  - Приложения пользователей

* **Режим ядра (не защищен, можно залезть в чужую память)**

  - Системные вызовы

  - Ядро системы

    - менеджер памяти

    - менеджер процессов

    - подсистема ввода-вывода

    - драйвера - работают не с аппаратурой, а с HAL

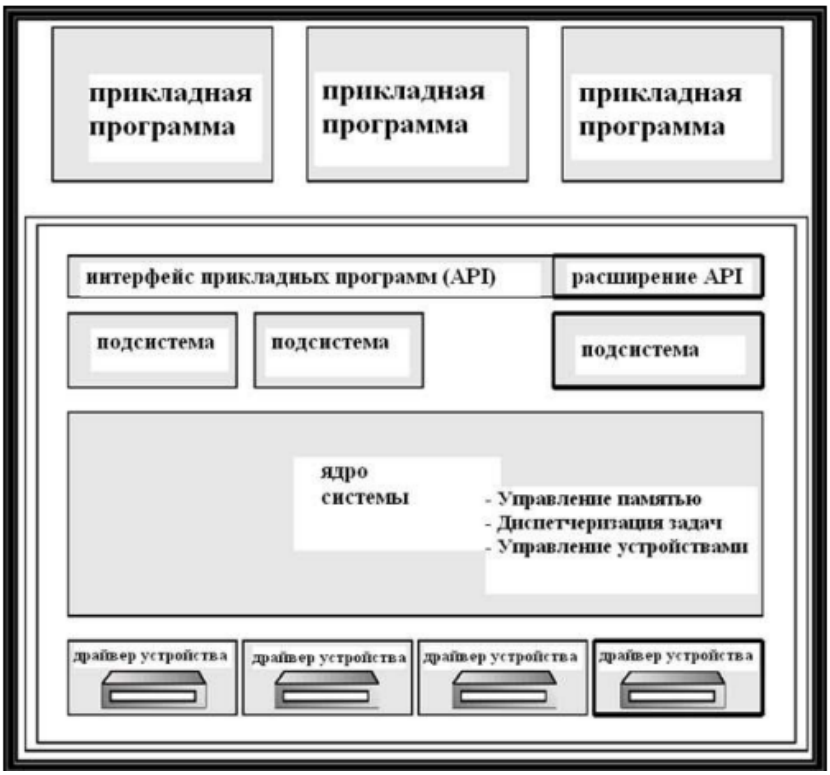
      - не "установи 3.5В", а просто "установи логическую 1"

  - HAL (Hardware Abstraction Layer) - промежуточный слой

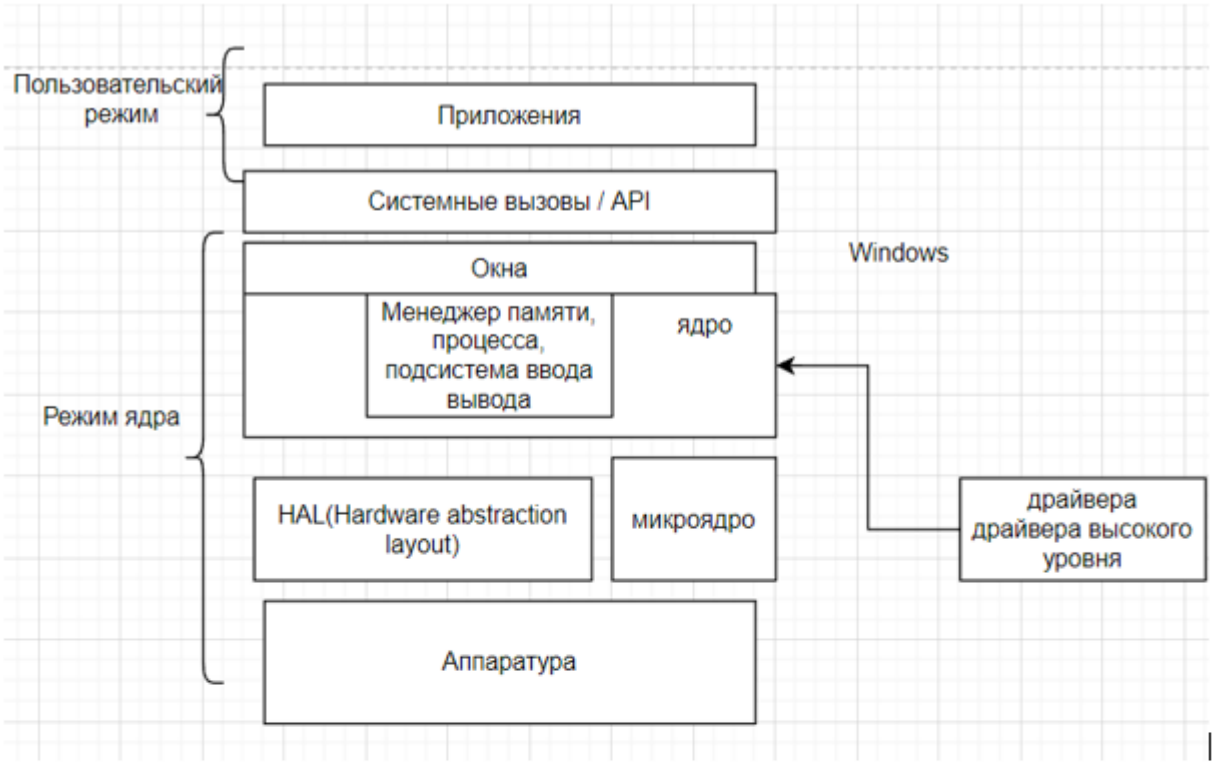
    - уровень абстракции программного обеспечения

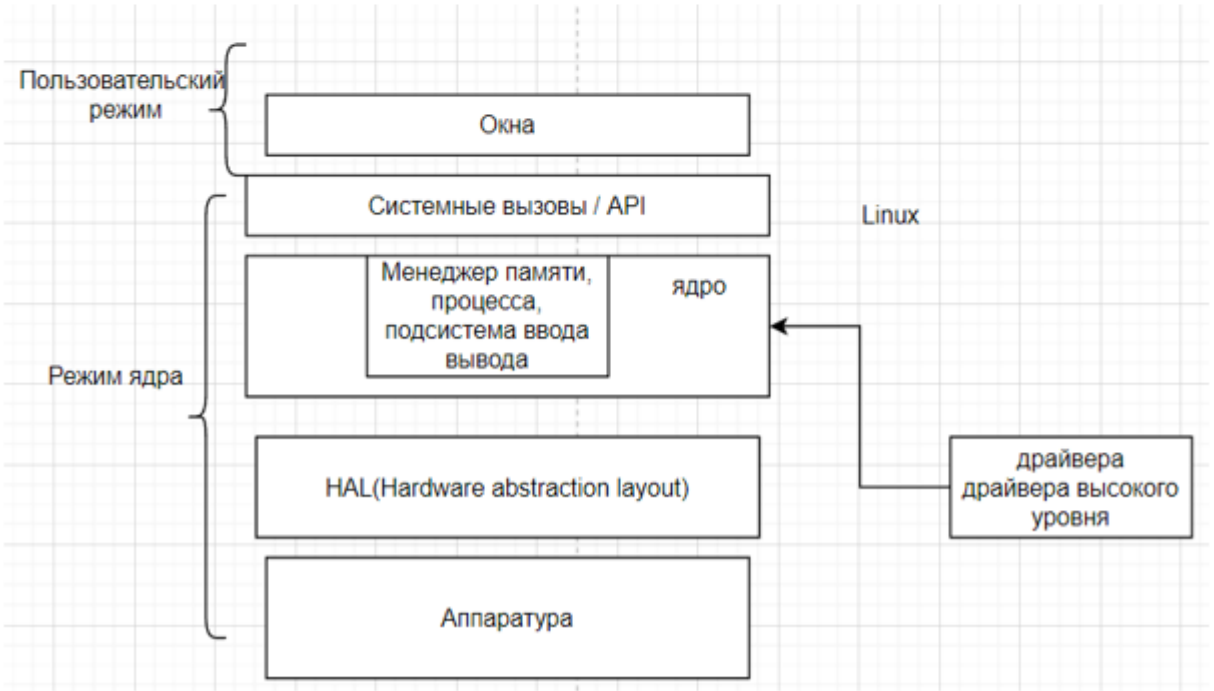
    - 0В - логический 0, больше 3.5В - логическая 1

* **Аппаратура**



Уровни абстракции ОС для Windows, Linux





Примеры

1. Процесс в памяти останавливаем, чтобы запустить другой

- записываем старый процесс на диск

- функционирует менеджер процессов

- должен иметь доступ к менеджеру памяти и подсистеме ввода-вывода (чтобы записать на диск)

- т.е. менеджер процессов имеет доступ к другим менеджерам

- режим ядра не защищен

2. Оконная (графическая) подсистема в UNIX расположена в пользовательском режиме

- в Windows в режиме ядра (чтобы не тормозило)

- чтобы не было переключений между режимами

- ведем мышкой

- прерывание устройства ввода

- процесс останавливается

- вызываем обработчик прерывания

- расчет координат мыши

- переходим в пользовательский режим

- в подсистему "Окна" передаем новые координаты мышки

- однако оконная система может быть не недежна

- залезть в область памяти других менеджеров и что-то записать

- -> синий экранd

- частота современных процессоров устранила тормоза (на прерывания)

Перенос ОС с ПК на Смартфон - изменяем HAL (Hardware Abstraction Layer) - промежуточный слой

- уровень абстракции программного обеспечения

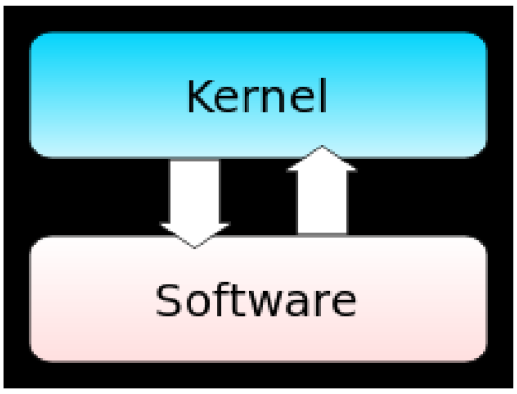
- 0В - логический 0, больше 3.5В - логическая 1

Два принципа построения ядра ОС:

- **монолитное ядро (Linux)**

Приложения

Ядро (службы, менеджеры)

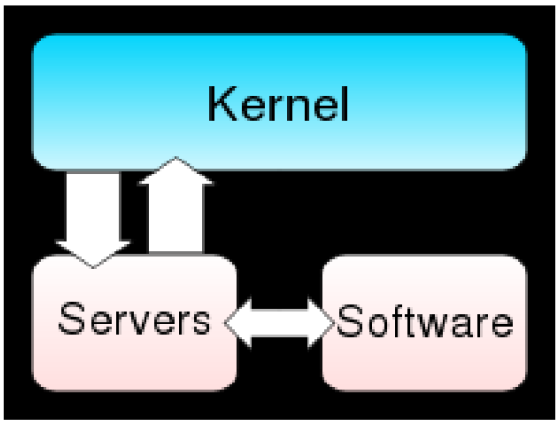


- **микроядро (Minix)**

Приложерия

Сервер1 Сервер2 Сервер3 (службы в пользовательском режиме)

Ядро (мин. функция - обмен сообщениями)



Процедура остановки процесса и запись на диск

- переключения между серверами
- постоянно переходим в режим ядра
- медленно (500 МГц -> 300 МГц - большая разница)

Другие подходы:

- гибридное ядро;
- экзоядров;
- наноядро;

Пример ядра: Iphone – XNU

Переносимость принципы:

1) Язык программирования
2) минимизация ассемблерного кода
3) создание API
4) создание абстракций

Переносимость ОС зависит от

- **архитектура процессора**
  - Гарвардская
    - отдельно данные, отдельно инструкции
  - Стэндфордская (архитектура фон неймана)
    - принцип совместного хранения команд (инструкций) и данных в одной памяти (в одних шинах)
- **архитектура вычислительной системы** (ВС)

Переносимость

- Win - HAL заменить
- Linux - ядро перекомпилировать

Первый пример переносимости – UNIX

- максимум кода на СИ
- минимум на asm
- создание API (одинаковое везде, разная реализация)
- создание абстракций

UNIX: все есть файл

клава - файл из которого можно читать

экран - файл в который можно записывать

(в графическом режиме - эмулятор терминала

в консольном режиме - терминал

echo Hello > /dev/tty3 (запись "Hello" в файл)

- в третьей консоли теперь Hello)

еще пример абстракции

(копилятор minGW - gcc на Windows

прослойка - преобразует системные вызовы Unix в системные вызовы Win

код для Linux

вызывает системные вызовы Windows

постоянная ретрансляция = медленно)

Вычислительная система (ВС) - совокупность взаимосвязанных и взаимодействующих процессоров или ЭВМ, периферийного оборудования и программного обеспечения, предназначенную для сбора, хранения, обработки и распределения информации

взаимосвязанные и взаимодействующие - обязательные условия

- два компа рядом - не ВС

- ВС - 2 компа по сети

- софт для распределение задачи (MPI - пересылка сообщений друг другу)

- мини вычислительный кластер

сбор, хранение, обработка (обязательно для ВС) и распределение информации

определение информационной системы (ИС)

Пример: Switch (коммутатор) - не ИС (ничего не обрабатывает) - получил пакет, передал пакет

Различают вычислительные системы (ВС):

- **однопрограммные и многопрограммные**

(в зависимости от количества программ, одновременно находящихся в оперативной памяти);

- **индивидуального и коллективного пользования**

(в зависимости от числа пользователей, которые одновременно могут использовать ресурсы ВС);

- **с пакетной обработкой** (обучение моделей ML) **и разделением времени** (разделение времени между процессами)

(в зависимости от организации и обработки заданий);

- **однопроцессорные, многопроцессорные и многомашинные** (в зависимости от числа процессоров);

- многомашинные = кластер

- **Сильносвязная, среднесвязная, слабосвязная** (в зависимости от степени связности элементов)

- ядра без общей памяти - слабосвязная система

- ядра с общей памятью - сильносвязная система

- и общая и раздельная - среднесвязная система

- **сосредоточенные, распределенные** (вычислительные сети) **и ВС с теледоступом** (в зависимости от территориального расположения и взаимодействия технических средств);

- **работающие или не работающие в режиме реального времени** (в зависимости от соотношения скоростей поступления задач в ВС и их решения);

- **универсальные, специализированные и проблемно-ориентированные** (в зависимости от назначения, e.g. Leonhard А.Ю.Попова).

Типы связанности:

1) Слабо (отдельный компьютер)
2) Средне (одна плата, но у каждого есть своя память)
3) Сильносвязанные (на одной плате)

Режимы работы вычислительных систем

1. Мультипрограммирование

2. Режим реального времени

3. Однопрограммный режим работы вычислительной системы (ВС)

4. Мультипрограммный режим работы вычислительной системы (ВС)

5. Режим пакетной обработки

6. Режим коллективного доступа

Виды ресурсов ВС

Под **ресурсом** понимают какой-либо объект, который может распределяться внутри вычислительной системы (ВС) между конкурирующими за него процессами.

Ресурс выделяется процессу на определенный интервал времени.

Ресурсы запрашиваются, используются и освобождаются процессами.

Ресурсы:

1) *Процессорное время*
2) *Устройства*

Классификация ресурсов ВС

1. По форме реализации:

- аппаратные ресурсы (Hard);
- программные ресурсы (Soft);
- информационные ресурсы.

2. По способу выделения ресурса различают:

- неделимые ресурсы – предоставляются процессу в полное распоряжение;

- делимые ресурсы – предоставляются процессу в соответствии с запросом на требуемое количество ресурса.

3. По реальности существования различают:

- физический ресурс – реально существует и при распределении обладает всеми присущими ему свойствами;

- виртуальный ресурс – программноаппаратная модель физического ресурса.

4. По месту размещения ресурса различают:

- локальные ресурсы – принадлежат автономному компьютеру;

- удаленные ресурсы – принадлежат рабочим станциям или серверам, входящим в состав сети.

ОС выполняет две группы функций:

- предоставление пользователям и программистам вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобней работать и легче программировать;

- повышение эффективности использования компьютера путем рационального управления его ресурсами.

С какими ресурсами имеет дело ОС с точки зрения пользователя в соотвествие с каждой группой функций? (? K biết đây có phải trả lời cho câu hỏi này ko @@)

Есть индивидуальное и совместное использование:

1) Индивидуальное – использование только для себя (принтер, печатает только один файл, другие в очереди)
2) Совместное – (память на диске, оперативная)
   Индивидуальное, коллективное, в каком времени работают и тд.

Управление ресурсами включает решение следующих задач:

- планирование ресурса, то есть определение, кому, когда, а для делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;

- выделение ресурса;

- отслеживание состояния ресурса, то есть поддержание оперативной информации о том, занят или не занят ресурс, а для делимых ресурсов – какое количество ресурса уже распределено, а какое свободно;

- освобождение ресурса.

плюсы и минусы Линукс и Винд.

Linux

Плюсы	Минусы
Бесплатное свободно распространяемое ПО	Поддержка не большого ассортимента компьютерного оборудования
Практически отсутствует вредоносное ПО	Значительно меньшее, чем для Windows, количество прикладных программ
Независимость от разработчика	Не большое количество хороших специалистов

Windows

Плюсы	Минусы
Поддержка очень большого ассортимента компьютерного оборудования	Сравнительно высокая стоимость
Огромное количество прикладных программ	Очень большое количество вредоносных программ
Большое количество специалистов	Жесткая зависимость от разработчика

Что такое процесс?

Процесс – выполняемая программа. В один момент времени процессор занят одной задачей, если он не многоядерный.

Особенность пакетной – хотим обработать как можно быстрее пакет.

Интерактивные – нужно отвечать пользователю.

Типы систем с точки зрения одновременно выполняемых процессов:

- однозадачная;

- многозадачная.

!!! В каждый конкретный момент времени процессор выполняет только одной процесс

Атрибуты процесса

- Счетчик команд

- Регистры

- Переменные

Что такое счетчик команд?

Счётчик кома́нд (также **PC** = *program counter*, **IP** = *instruction pointer*, **IAR** = *instruction address register*, **СчАК** = *счётчик адреса (машинных) команд*) — регистр процессора, который указывает, какую команду нужно выполнять следующей[1].

Что такое регистры ЦП? Какие бывают регистры?

Регистр процессора — поле заданной длины во внутрипроцессорной сверхбыстрой оперативной памяти (СОЗУ). Используется самим процессором, может быть как доступным, так и недоступным программно. Например, при выборке из памяти очередной команды она помещается в регистр команд, обращение к которому программист прописать не может.

Граф состояний процесса

Состояния:

Выполнение (работа)

Готовность

Блокировка

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:

ВЫПОЛНЕНИЕ - активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

ОЖИДАНИЕ - пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса;

ГОТОВНОСТЬ - также пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.

В ходе жизненного цикла каждый процесс переходит из одного состояния в другое в соответствии с алгоритмом планирования процессов, реализуемым в данной операционной системе. Типичный граф состояний процесса показан на рисунке 2.1.

В состоянии ВЫПОЛНЕНИЕ в однопроцессорной системе может находиться только один процесс, а в каждом из состояний ОЖИДАНИЕ и ГОТОВНОСТЬ - несколько процессов, эти процессы образуют очереди соответственно ожидающих и готовых процессов. Жизненный цикл процесса начинается с состояния ГОТОВНОСТЬ, когда процесс готов к выполнению и ждет своей очереди. При активизации процесс переходит в состояние ВЫПОЛНЕНИЕ и находится в нем до тех пор, пока либо он сам освободит процессор, перейдя в состояние ОЖИДАНИЯ какого-нибудь события, либо будет насильно "вытеснен" из процессора, например, вследствие исчерпания отведенного данному процессу кванта процессорного времени. В последнем случае процесс возвращается в состояние ГОТОВНОСТЬ. В это же состояние процесс переходит из состояния ОЖИДАНИЕ, после того, как ожидаемое событие произойдет.