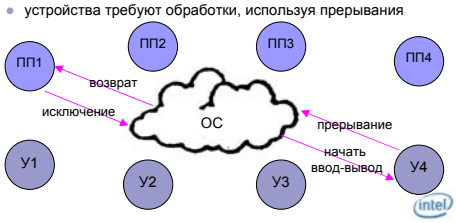
Лекция 3: Архитектура и компоненты операционных систем

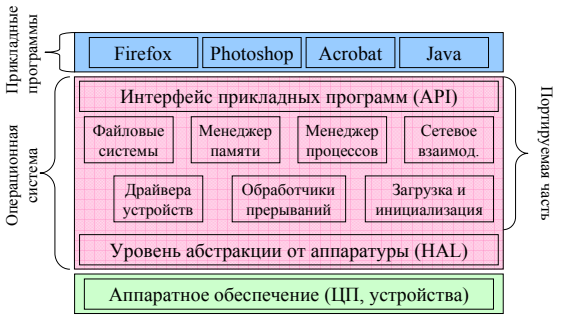
Алексей Линёв
Александр Мощук
Кирилл Погорельский

somes slides are adapted from the OS course at the University of Washington

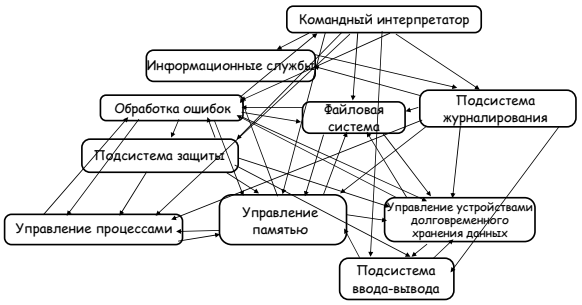
Архитектура ОС

- ОС расположена между прикладными программами и аппаратным обеспечением
  - ОС выступает посредником при доступе и предоставляет удобные абстракции
  - программы запрашивают сервис, инициируя исключения (ловушки или ошибки – traps or faults)
  - устройства требуют обработки, используя прерывания





Архитектура ОС



Основные компоненты ОС

- Процессы
- Оперативная память
- Ввод-вывод
- Устройства долговременного хранения данных
- Файловые системы
- Защита
- Журналирование (обработка статистических данных)
- Пользовательские оболочки (командный интерпретатор или интерфейс пользователя ОС)
- Графический интерфейс пользователя
- Сетевое взаимодействие

Управление процессами

- В ОС исполняются множество видов активностей:
  - пользовательские приложения
  - пакетные программы или скрипты
  - системные программы
    - очереди печати, сервера имен, файловые сервера, другие сетевые сервисы,…
- Каждая из этих активностей инкапсулирована в *процесс*
  - понятие процесса включает *контекст* выполнения
    - программный счетчик, регистры, виртуальная память, ресурсы ОС (например, открытые файлы) и т.д.
    - выполняющаяся программа (код и данные)
  - одна из компонент ОС управляет этими процессами
    - создает, разрушает, выделяет им ЦП и т.д.

Программа/Процессор/Процесс

- Запомните: программа полностью пассивна
  - это просто набор байтов на диске, в которых закодированы выполняемые инструкции
- Процесс – это экземпляр программы, исполняемый процессором (реальным или виртуальным)
  - могут существовать несколько процессов, выполняющих одну и ту же программу (например, редактор)
  - эти процессы обособлены и (обычно) независимы
  - Linux: ps aux отображает список всех процессов

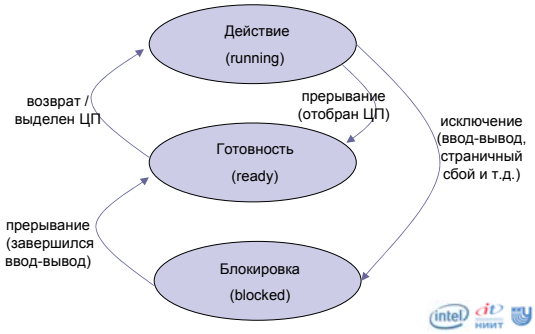
Процесс A

код стек IP регистры	таблицы страниц ресурсы

Процесс B

код стек IP регистры	таблицы страниц ресурсы

Состояния пользовательских процессов



Операции над процессами

- ОС предоставляет следующие операции над процессами (интерфейс работы с процессами)
  - создание процесса
  - уничтожение процесса
  - приостановление выполнения процесса
  - возобновление выполнения процесса
  - создание копии процесса
  - средства передачи данных между процессами
  - средства синхронизации процессов
  - создание/удаление *подпроцесса* (child process, subprocess))

Управление памятью

- Основная память (оперативная память, RAM) – хранилище данных, непосредственно доступное ЦП
  - исполняющиеся программы должны находиться в RAM
  - RAM обладает низким временем доступа (10 нс)
    - но ее содержимое теряется при выключении ЭВМ
- ОС должна
  - выделять программам оперативную память (явно и неявно)
  - перераспределять память при необходимости
  - поддерживать отображение виртуальной памяти на физическую
    - посредством таблиц страниц
  - определять, сколько памяти выделить каждому процессу
    - на основании некоторой стратегии (policy decision)
  - определять момент выгрузки процесса из оперативной памяти
    - также должна существовать некоторая стратегия

Ввод-вывод

- Большая часть ядра ОС связана с вводом-выводом
  - сотни тысяч строк кода в Windows NT
- ОС предоставляет системным и прикладным программам стандартный интерфейс доступа к устройствам
  - файловая система (жесткий диск), сокеты (сеть), кадровый буфер (видео)
- *Драйвера устройств* – это совокупности функций, взаимодействующих с устройствами различных типов
  - *инкапсулируют* специфику взаимодействия с устройством
  - например, как инициализировать устройство, начать ввод-вывод, обработать прерывания или ошибки
  - примеры: драйвера SCSI-устройств, Ethernet-адаптеров, видеоадаптеров, звуковых карт,…
- Примечание. Windows содержит ~35000 драйверов устройств!

Внешняя память (secondary storage)

- Внешняя память (диск, лента) – постоянная память
  - часто энергонезависимый носитель магнитного типа
- Функции, взаимодействующие с дисками, как правило, расположены на очень низком уровне ОС
  - используются множеством компонент
    - подсистемы управления файлами, виртуальной памятью и т.д.
  - управляют
    - последовательностью выполнения дисковых операций
    - движением головок
    - обработкой ошибок
    - часто – распределением свободного дискового пространства

Внешняя память (secondary storage)

- Как правило, функции, взаимодействующие с дисками, отделены и независимы от подсистемы управления файлами
  - в частных случаях между ними может быть более тесное взаимодействие
  - использование подсистемой управления файлами информации о внутреннем устройстве внешней памяти может повысить производительность
    - например, размещение связанных файлов близко на диске

Файловые системы

- Непосредственное использование устройств внешней памяти очень неудобно
  - напр., "записать блок размером 4096 байт в сектор 12"
- Файловая система – удобная абстракция
  - определяет логические объекты, такие как *"файлы"* и *"каталоги"*
    - скрывает подробности фактического размещения объектов на жестком диске
  - определяет операции над объектами
    - например, чтение и запись
      - чтение/запись логических последовательностей байт, а не блоков жесткого диска

Файловые системы

- *Файл* – основной объект долговременного хранения
  - файл – уникально именованный набор долговременно хранящихся данных
- *Каталог* – специальный тип объекта файловой системы
  - каталог – файл, содержащий имена других файлов и метаданные, относящиеся к этим файлам (например, размер)
- Замечание: использование последовательного потока данных – это только один из возможных вариантов!

Операции файловых систем

- Интерфейс подсистемы управления файлами определяет набор стандартных операций:
  - создание и удаления файла или каталога
  - чтение, запись, изменение размера, переименование, изменение атрибутов защиты
  - копирование
  - блокировка (lock)
- Файловые системы также предоставляют высокоуровневые сервисы
  - журналирование операций и квотирование (accouning, quotas)
  - резервное копирование (backup)
    - (должно быть инкрементным и выполняться в реальном времени)
  - (иногда) индексирование или поиск
  - (иногда) поддержка использования версий файлов

Подсистема защиты

- Защита – основополагающий механизм, пронизывающий всю ОС
  - все ресурсы должны быть защищены
    - оперативная память
    - процессы
    - файлы
    - устройства
    - время ЦП
    - …
  - механизмы защиты помогают обнаруживать и пресекать некорректные действия, являющиеся следствием как неумышленных ошибок, так и злонамеренных вторжений

Командный интерпретатор (shell)

- Специальная программа, интерпретирующая команды пользователя и помогающая управлять процессами
  - пользовательский ввод может приниматься
    - с клавиатуры (интерфейс командной строки)
    - из файлов-скриптов
    - с мыши (графический интерфейс)
  - позволяет пользователям запускать программы и управлять ими
- В одних случаях командный интерпретатор может являться неотъемлемой частью ОС (MS DOS, Apple II)
- В других случаях – это обычная непривилегированная программа, предоставляющая пользовательский интерфейс
  - например, bash (или csh, tcsh, zsh, ash, sh) в UNIX
- Существуют системы, в которых не предусмотрен интерфейс командной строки
  - например, MacOS

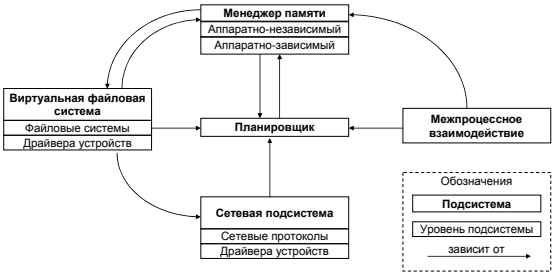
Журналирование (Accounting)

- Отслеживание статистики использования ресурсов
  - для реализации квотирования (enforce quotas)
    - "Вы превысили ограничение по занимаемому дисковому пространству"
  - для выставления счетов
    - при использовании систем разделения времени, таких как мейнфреймы
    - при представлении ресурсов для размещения сервисов

графический интерфейс … сеть … и т.д.

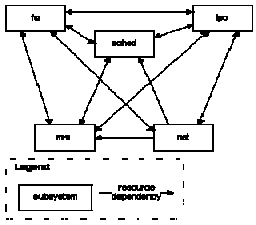
Архитектура ОС

- Все еще не ясно, как связать компоненты ОС вместе:



Conceptual Organization
(Linux 1998)
(from http://plg.uwaterloo.ca/~itbowman/CS746G/a2/#_Toc411854285)

Архитектура ОС



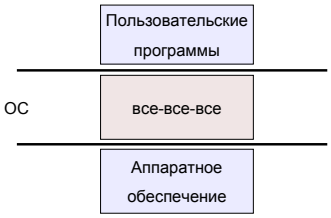
Реальная архитектура
(Actual Organization)
(from http://plg.uwaterloo.ca/~itbowman/CS746G/a2/#_Toc411854285)

Архитектура ОС

- ОС включает все перечисленные подсистемы, а также
  - множество других компонент
  - системные программы (привилегированные и непривилегированные)
    - например, код начальной загрузки, программа init (UNIX),…
- Основной вопрос:
  - как объединить все это?
  - какие программные модули используются и где они расположены?
  - как они взаимодействуют?
- Задача проектирования и разработки огромного объема
  - необходимо спроектировать большую и сложную программу таким образом, чтобы она
    - обладала хорошей производительностью
    - была надежна
    - расширяема
    - обратно совместима,…

Ранние архитектуры: Монолитная структура

- Согласно традиции, ОС (такие как UNIX) строились в виде монолитной сущности



Монолитная архитектура

- Главное преимущество:
  - низкая стоимость межмодульного взаимодействия (фактически реализуемого в виде вызовов функций других модулей)
- Недостатки:
  - сложно понять
  - сложно изменять
  - ненадежность (отсутствие защиты модулей друг от друга)
  - сложно поддерживать
- Существуют ли альтернативы?
  - необходимо найти способ упорядочить архитектуру ОС – это упростит ее разработку и реализацию

Слоеные/многоуровневые системы (Layering)

- Традиционный подход к упрощению архитектуры – вертикальная композиция или разбиение на слои/уровни
  - ОС реализуется в виде набора уровней
  - каждый уровень предоставляет вышележащему "виртуальную машину"
  - сложность и функциональность "виртуальных машин" при этом нарастает

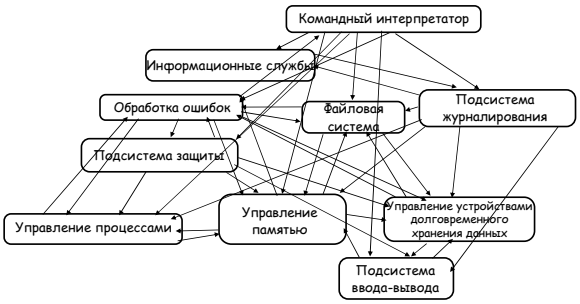
Слоеные/многоуровневые системы (Layering)

- Первое описание данного подхода дал Дийкстра в системе THE
  - Уровень 5: Менеджеры задач
    - Выполняют программы пользователей
  - Уровень 4: Менеджеры устройств
    - Управляют устройствами и выполнят буферизацию
  - Уровень 3: Менеджер консоли
    - Обеспечивает функционирование виртуальных консолей
  - Уровень 2: Менеджер страниц
    - Обеспечивает виртуальное адресное пространство для каждого процесса
  - Уровень 1: Ядро
    - Обеспечивает выделение каждому процессу виртуального процессора
  - Уровень 0: Аппаратное обеспечение
- Каждый уровень может разрабатываться и отлаживаться независимо от остальных

Проблемы многоуровневой архитектуры

- Устанавливают иерархическую структуру
  - но реальные системы более сложны
    - подсистема управления файлами требует сервиса со стороны подсистемы виртуальной памяти (буфера)
    - подсистема виртуальной памяти может захотеть использовать файл в качестве пространства подкачки
  - строгое разделение на уровни – недостаточно гибкий подход
- Низкая производительность
  - переход через каждый уровень порождает дополнительные *накладные расходы* (*overhead*)
- Имеет место противоречие между моделью и реальностью
  - системы моделируются как многоуровневые, но их реализации таковыми не являются

Архитектура ОС



Микроядерная архитектура (Microkernels)

- Была популярна в конце 80-х – начале 90-х
  - происходит возрождение интереса
- Основная цель:
  - минимизировать обработку в ядре
  - реализовать остаток ОС в виде пользовательских процессов
- Результаты:
  - надежность (компоненты ОС изолированы)
  - упрощение расширения и настройки
  - низкая производительность (из-за частых пересечений рубежа между уровнями ядра и пользователя)
- Первая микроядерная система – Hydra (CMU, 1970)
  - Последователи: Mach (CMU), Chorus (французская UNIX-like OS), OS X (Apple), в некоторой степени – Windows NT (Microsoft)

Микроядерная архитектура