Введение в операционные системы

- В составе компьютера выделяют три основных блока: процессор, память, устройства ввода-вывода
- Написание программ, которые корректно и оптимально работают с этим блоками, является крайне трудной задачей
- Операционная система — специальный уровень программного обеспечения, отвечает за управление всеми устройствами компьютера и обеспечивает пользователя простыми, доступными средствами для работы с аппаратурой

Уровни абстракций ЭВМ

- На нижнем уровне находятся физические устройства — микропроцессоры как наборы интегральных схем, микросхемы доступа к памяти, физические устройства ввода/вывода и т.п.
- Далее следует микроархитектурный уровень, на котором физические устройства рассматриваются в виде блоков, выполняющих элементарные операции, рассмотренные на цифровом логическом уровне
- Для того, чтобы пользователь получил доступ к функциональным блокам уровня микроархитектуры, предлагается набор инструкций, или набор команд, называемый машинным языком

Структура компьютера и ПО

- Программы и приложения на верхнем уровне обеспечивают потребности людей, которые используют компьютеры для решения своих задач
- Промежуток между этими уровнями достаточно большой и без наличия дополнительных уровней абстракций решение практических задач становится очень и очень трудным
- Дальнейшее изложение курса будет посвящено уровню операционных систем

Назначение ОС

- **Управление ресурсами компьютера**. ОС обеспечивает организованное и контролируемое распределение ресурсов компьютера между различными приложениями (процессами).
  - Ресурсы разделяются во времени
  - Ресурсы разделяются в пространстве
- **Расширение возможностей ЭВМ**. Работа с оборудованием на уровне машинного языка примитивна и трудоемка. Программистам гораздо проще и удобнее использовать абстракции более высокого уровня. Операционная система предоставляет такого уровня абстракции.

История ОС: I поколение

- Программы на первых ЭВМ реализовывались в виде коммутационных панелей и представляли собой схемы для прямых численных вычислений
- На смену коммутационным панелям пришли перфокарты
- Отсутствуют языки программирования, даже ассемблер
- Уровень развития оборудования пока не позволяет говорить об операционных системах

История ОС: II поколение

- Транзисторы: существенно выросли возможности ЭВМ. Появляются большие ЭВМ, называемые мейнфреймами. Складывается четкое разделение труда между проектировщиками, сборщиками, операторами, программистами и обслуживающим персоналом.
- Вместо машинных кодом применяют мнемонические обозначения и используют специальные программы — сборщики программ из небольших фрагментов кода — ассемблеры.
- Появляется FORTRAN — язык программирования высокого уровня, имеющий транслятор.
- Процесс разработки программы
  - Программист записывал задание на бумаге
  - Затем переносил задание на перфокарты
  - Колода перфокарт передавалась оператору для ввода.
  - По окончании работы программы результаты распечатывались на принтере.
  - Если в процессе расчетов был необходим компилятор языка FORTRAN, то оператор загружал его отдельно.

История ОС: системы пакетной обработки

- Пакет — набор заданий (различных программ).
- С перфокарт на недорогих компьютерах задания переписывались на магнитную ленту.
- Затем магнитная лента передавалась для выполнения на «большом», производительном компьютере — мейнфрейме.
- Выходные данные также записывались на ленту, после окончания задания данные с этой ленты распечатывались на принтере на отдельной недорогой ЭВМ.
- Для формирования задания использовались управляющие перфокарты: $ЗАДАНИЕ, $FORTRAN, $ЗАГРУЗИТЬ, $ЗАПУСТИТЬ, $КОНЕЦ.
- Управляющие перфокарты — предшественники современных языков программирования и интерпретаторов команд.
- Fortran Monitor System. IBSYS

Ранняя система пакетной обработки: *а* — программист приносит карты для IBM 1401; *б* — IBM 1401 записывает пакет заданий на магнитную ленту; *в* — оператор приносит входные данные на ленте к IBM 7094; *г* — IBM 7094 выполняет вычисления; *д* — оператор переносит ленту с выходными данными на IBM 1401; *е* — IBM 1401 печатает выходные данные

История ОС: Fortran Monitor System

- **Fortran Monitor System** представлял собой набор небольших процедур, которые помещались между отдельными заданиями или частями заданий в колоде перфокарт или на магнитной ленте
- Процедуры позволяли записывать текущее задание или его результаты на диск, загружать следующее в очереди задание в память и передавать ему управление
- В состав **FMS** включался ассемблер, позволяющий создавать код программы, пригодный для загрузки и запуска и позволяющий работать совместно с кодом на языке Fortran

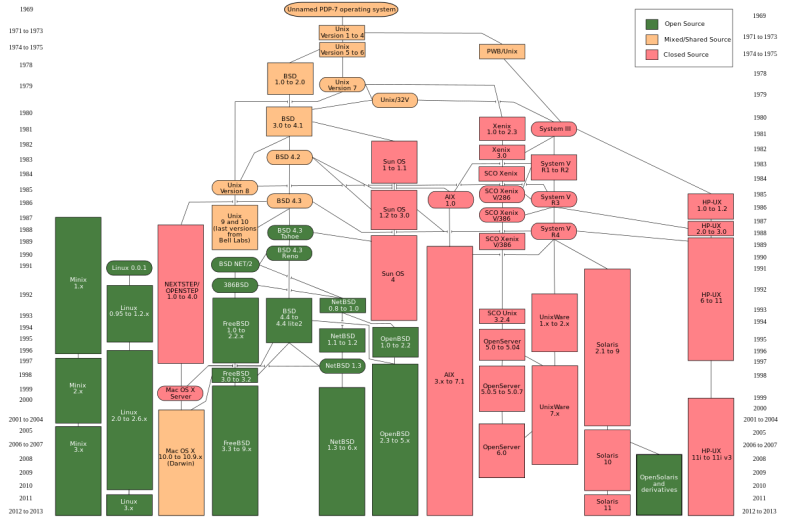
История ОС: III поколение

- Совместные периферийные операции в режиме подключения — способность считывания с перфокарт на диск по мере их поступления в машинный зал (подкачка данных, спулинг, spooling — SPOOL (Simultaneous Peripheral Operation On Line). Когда текущее задание заканчивалось, операционная система могла загрузить новое задание с диска в освободившийся раздел памяти.
- Многозадачность — возможность одновременного использования различных ресурсов машины (процессор, память, устройства ввода/вывода) разными приложениями.
- Системы разделения времени — вариант многозадачности, в которой у каждого пользователя есть свой терминал.
- OS/360
- CTSS (Compatible Time Sharing System), система с разделением времени, разработана в MIT

История ОС: IV поколение

- Появление больших интегральных схем привело к созданию микрокомпьютеров для персонального использования
- **CP/M** (Control Program for Microcomputers) — работа с диском, запуск программ (для систем на базе Intel 8080)
- **DOS**, **MSDOS** (Disk Operation System) — для систем на базе Intel x86
- Графический пользовательский интерфейс (окна, меню, мышь) — Даг Энгельбарт, 60-е годы двадцатого века
- **Apple Macintosh** — дружеский пользовательский интерфейс
- **Windows** —ОС с графическим пользовательским интерфейсом, сначала поверх MSDOS, затем самостоятельная

История ОС: дерево UNIX



Основные понятия и функции ОС

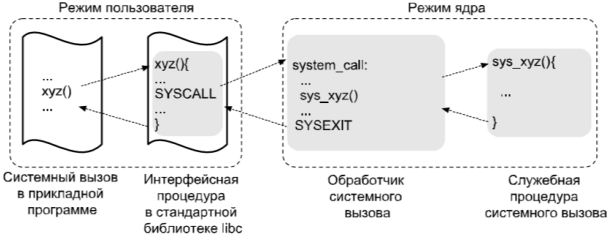
- Процесс — программа во время её выполнения (абстракция, описывающая выполняющуюся программу)
- Многозадачность — одновременное нахождение в памяти нескольких процессов
- Адресное пространство процесса — адреса памяти, которые использует/может использовать процесс
- Файл — именованный набор данных
- Файловая система — способ (метод) организации файлов, включая отображение файлов на физический носитель
- Управление процессами
- Управление памятью
- Управление устройствами ввода/вывода

Системные вызовы

- Интерфейс между пользовательскими программами и операционной системой стоится в основном на абстракциях
- Преимущества дополнительного программного слоя:
  - облегчается программирование, потому что программисты избавлены от необходимости изучать низкоуровневые характеристики аппаратных устройств
  - повышается безопасность системы, поскольку ядро может проверить корректность запроса на уровне интерфейса до выполнения этого запроса
  - программы становятся более переносимыми, позволяя компилировать и корректно выполнять их в каждом ядре операционной системы, предлагающем такой же набор интерфейсов
- Системный вызов — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции
- Приложения не могут получать доступ напрямую к системным ресурсам
- API (Application Programmer Interface, интерфейс прикладного программирования) представляет собой определение функции, описывающее, как получить определенный сервис или услугу. Системный вызов — непосредственное обращение к ядру ОС

Выполнение системного вызова

- Помещение параметров системного вызова на стек
- Помещение номера системного вызова в регистр
- Переключение в режим ядра
  - или прерывание int $0x80
  - или инструкция sysenter (>Pentium II)
- Проверка допустимости системного вызова
- Вызов служебной процедуры, ассоциированной с номером системного вызова, который хранится в регистре
- Возврат к вызывающей процедуре

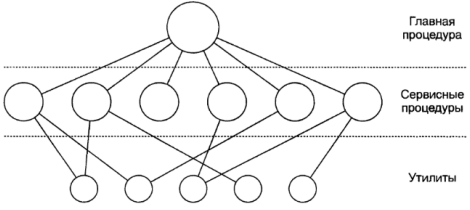


Категории системных вызовов

- Управление процессами
  - load
  - execute, end (exit)
  - abort/создание процесса (fork в Unix-like, NtCreateProcess в WindowsNT Native API)
  - get/set process attributes
  - wait время, события, signal события
  - allocate, free memory
- Работа с файлами
  - create file, delete file
  - open, close
  - read, write, reposition
  - get/set file attributes
- Управление устройствами
  - request device, release device
  - read, write, reposition
  - get/set device attributes
  - logically attach or detach devices
- Работа с информацией
  - get/set time or date
  - get/set system data
  - get/set process, file, or device attributes
- Связь, коммуникация
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

Структура ОС: монолитная

- В общем случае структура отсутствует, ОС реализуется в виде набора процедур, вызывающих друг друга
- Некоторая структура наблюдается при реализации системных вызовов: выполняется специальная инструкция перехвата управления (вызов ядра или вызов супервизора), переключает машину в режим ядра
- Такая реализация позволяет выделить следующие уровни:
  - Главная программа вызывает требуемую сервисную процедуру
  - Набор сервисных процедур, выполняющих системные вызовы
  - Набор утилит, обслуживающих служебные процедуры
- Для каждого системного вызова существует реализующая его сервисная процедура

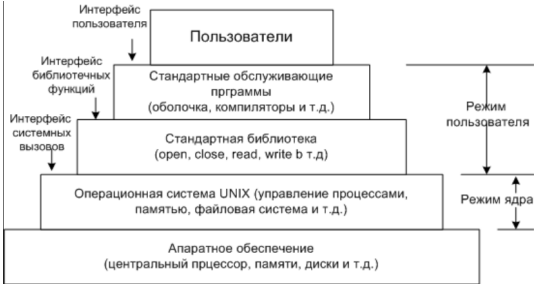


Структура ОС: многоуровневая

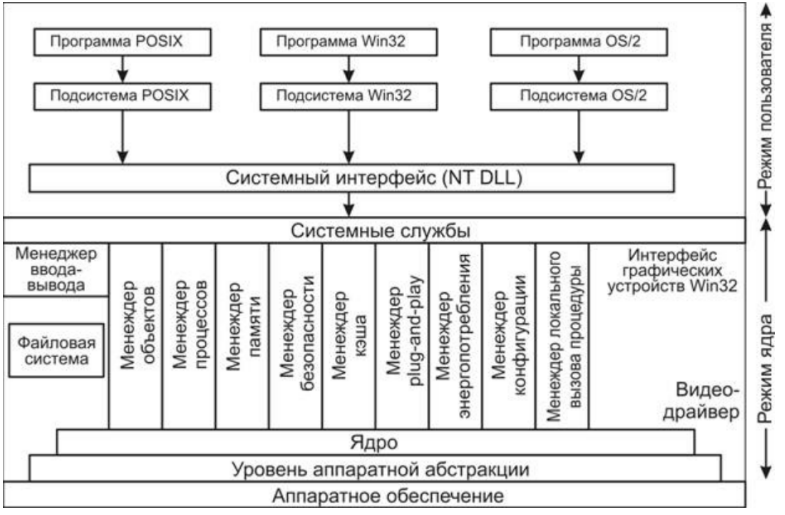
- Организация ОС в виде иерархии уровней
- Первая ОС с такой структурой: THE (Э.Дейкстра)
  - 5 Оператор
  - 4 Пользовательские программы
  - 3 Управление вводом/выводом
  - 2 Взаимодействие «оператор-процесс» (между консолью оператора и процессами)
  - 1 Управление памятью и барабаном — выделение процессам пространства в ОЗУ и магнитном барабане (виртуальная память)
  - 0 Выделение процессора и многозадачность

- MULTICS
  - Уровни — серия концентрических колец
  - Внутренние кольца более привилегированные, чем внешние
  - Вызов внешним кольцом процедуры внутреннего осуществляется эквивалентно системному вызову
- THE: многоуровневая схема только как конструктивное решение, все части системы собраны в одном файле
- MULTICS: механизм разделения колец действовал во время исполнения на аппаратном уровне, можно запустить программу в нужном кольце (при наличии привилегий)

Структура ОС: многоуровневая, UNIX

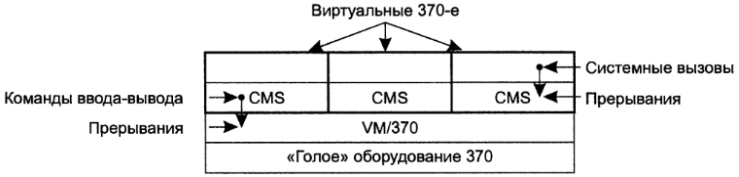


Структура ОС: многоуровневая, WINDOWS 2000



Структура ОС: виртуальные машины VM/370

- CP/CMS (позже VM/370) система с разделением времени для IBM/360
  - многозадачность
  - расширенная машина с более удобным интерфейсом доступа к оборудованию
- Монитор виртуальной машины: предоставляет верхнему уровню не одну, а несколько виртуальных машин
- Виртуальная машина не является расширенной, а предоставляет точную аппаратную копию, включая ввод/вывод, прерывания и т.п.
- На разных виртуальных машинах могут запускаться разные ОС (OS/360 — пакетные задания, CMS — Conversational Monitor System, система диалоговой обработки)
- В настоящее время — VMWare, VirtualBox, ...



Структура ОС: экзоядро

- В VM/370 каждый процесс пользователя получает точную копию настоящей машины
- На Pentium, в режиме виртуальной машины 8086, каждый пользовательский процесс получает точную копию машины
- Развитие идеи: система, в которой каждый пользователь получает абсолютную копию машины, но со своим подмножеством ресурсов (память, диск, ...)
- На нижнем уровне в режиме ядра работает экзоядро (exokernel) — программа для распределения ресурсов для виртуальных машин и проверки их использования.
- Преимущества:
  - нет уровня отображения ресурсов
  - отделение многозадачности (в экзоядре) от операционной системы пользователя осуществляется с меньшими затратами, необходимо лишь не допускать вмешательства работы виртуальных машин
- Идея осталась на уровне исследовательского проекта (ExOS, Nemesis)

Структура ОС: клиент-сервер

- В современных ОС существует тенденция переноса кода на верхние уровни и минимизация ядра
- Решение большинства задач операционной системы перекладывается на пользовательские процессы
- Пользовательский процесс (клиент) посылает запрос серверному процессу, который его обрабатывает и высылает ответ обратно
- Задача ядра — управление взаимодействием между клиентами и серверами
- Некоторые функции невозможно выполнить из пространства пользователя
  - возможно запускать такие процессы в режиме ядра с сохранением модели взаимодействия клиент-сервер (ранние версии MINIX, драйверы компилировались в ядро, но запускались как отдельные процессы)
  - встраивание в ядро минимальных механизмов обработки, но вынос «политических» решений в пользовательское пространство (например, операции чтения/записи с диска в ядре, а проверка доступа – в пространстве пользователя). MINIX3, драйверы находятся в пользовательском пространстве и посылают ядру специальные вызовы.