Тема 1 История, назначение, функции и виды операционных систем

История ОС

Первые (1945-1955г.г.) компьютеры работали без операционных систем, как правило, на них работала одна программа.

Появились первые системы **пакетной обработки** (1955-1965г.г.), которые просто автоматизировали запуск одной программ за другой и тем самым увеличивали коэффициент загрузки процессора.

История ОС

- **Многозадачность** (1965-1980) - это способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются несколько задач. Пока одна задача выполняет операцию ввода-вывода, процессор не простаивает, как это происходило при последовательном выполнении задач , а выполняет другую задачу.
- **Спулинг** (spooling-подкачка) в то время задания считывались с перфокарт на диск в том темпе, в котором они появлялись в помещении вычислительного центра, а затем, когда очередное задание завершалось, новое задание с диска загружалось в освободившийся раздел.
- **Системы разделения времени** - вариант многозадачности, при котором у каждого пользователя есть свой диалоговый терминал. Это было сделано, чтобы каждый программист мог отлаживать свою программу в реальном времени. Фактически это была многопользовательская система. Естественно стали возникать проблемы защиты задач друг от друга.

История ОС

- В это время была разработана многопользовательская система MULTICS, которая должна была обеспечивать одновременную работу сотни пользователей.
- В это время также стали бурно развиваться мини-компьютеры (первый был выпущен в 1961г.), на которые была перенесена система MULTICS. Эта работа в дальнейшем развилась в систему **UNIX.**
- Появилось много разновидностей несовместимых UNIX, основные из них были System V и BSD. Чтобы было возможно писать программы, работающие в любой системе UNIX, был разработан стандарт **POSIX.** Стандарт POSIX определяет минимальный интерфейс системного вызова, который должны поддерживать системы UNUX.
- В 1974г. был выпущен центральный процессор Intel 8080, для него была создана операционная система **CP/M.** В 1977г. она была переработана для других процессоров, например Z80.

История ОС

- В начале 80-х была разработана система **MS-DOS**, и стала основной системой для микрокомпьютеров.
- В 80-х годах стало возможным реализовать **графический интерфейс пользователя** (GUI - Graphical User Interface), теория которого была разработана еще в 60-е годы. Первой реализовала GUI корпорация Macintosh.
- С 1985 года выпустили **Windows**, в то время она была графической оболочкой к MS-DOS вплоть до 1995г., когда вышла **Windows 95.**

История ОС

- Уже тогда было ясно, что DOS с ее ограничениями по памяти и по возможностям файловой системы не может воспользоваться вычислительной мощностью появляющихся компьютеров. Поэтому IBM и Microsoft начинали совместно разрабатывать операционную систему **OS/2**. Она должна была поддерживать вытесняющую многозадачность, виртуальную память, графический пользовательский интерфейс, виртуальную машину для выполнения DOS-приложений. Первая версия вышла 1987г.
- В дальнейшем Microsoft отошла от разработки OS/2, и стала разрабатывать **Windows NT.** Первая версия вышла в 1993г.

История ОС

- В середине 80-х стали бурно развиваться сети персональных компьютеров, работающие под управлением сетевых или распределенных операционных систем.
- **Сетевая операционная система** не имеет отличий от операционной системы однопроцессорного компьютера. Она обязательно содержит программную поддержку для сетевых интерфейсных устройств (драйвер сетевого адаптера), а также средства для удаленного входа в другие компьютеры сети и средства доступа к удаленным файлам.
- **Распределенная операционная система**, напротив, представляется пользователям простой системой, в которой пользователь не должен беспокоиться о том, где работают его программы или где расположены файлы, все это должно автоматически обрабатываться самой операционной системой.

История ОС

- В 1987г. была выпущена операционная система **MINIX**, она была построена на схеме **микро ядра**.
- В 1991г. была выпущена **LINUX**, в отличии от микроядерной MINIX она стала монолитной.
- Чуть позже вышла **FreeBSD** (основой для нее послужила BSD UNIX).

Linux

Назначение ОС. ОС как виртуальная машина

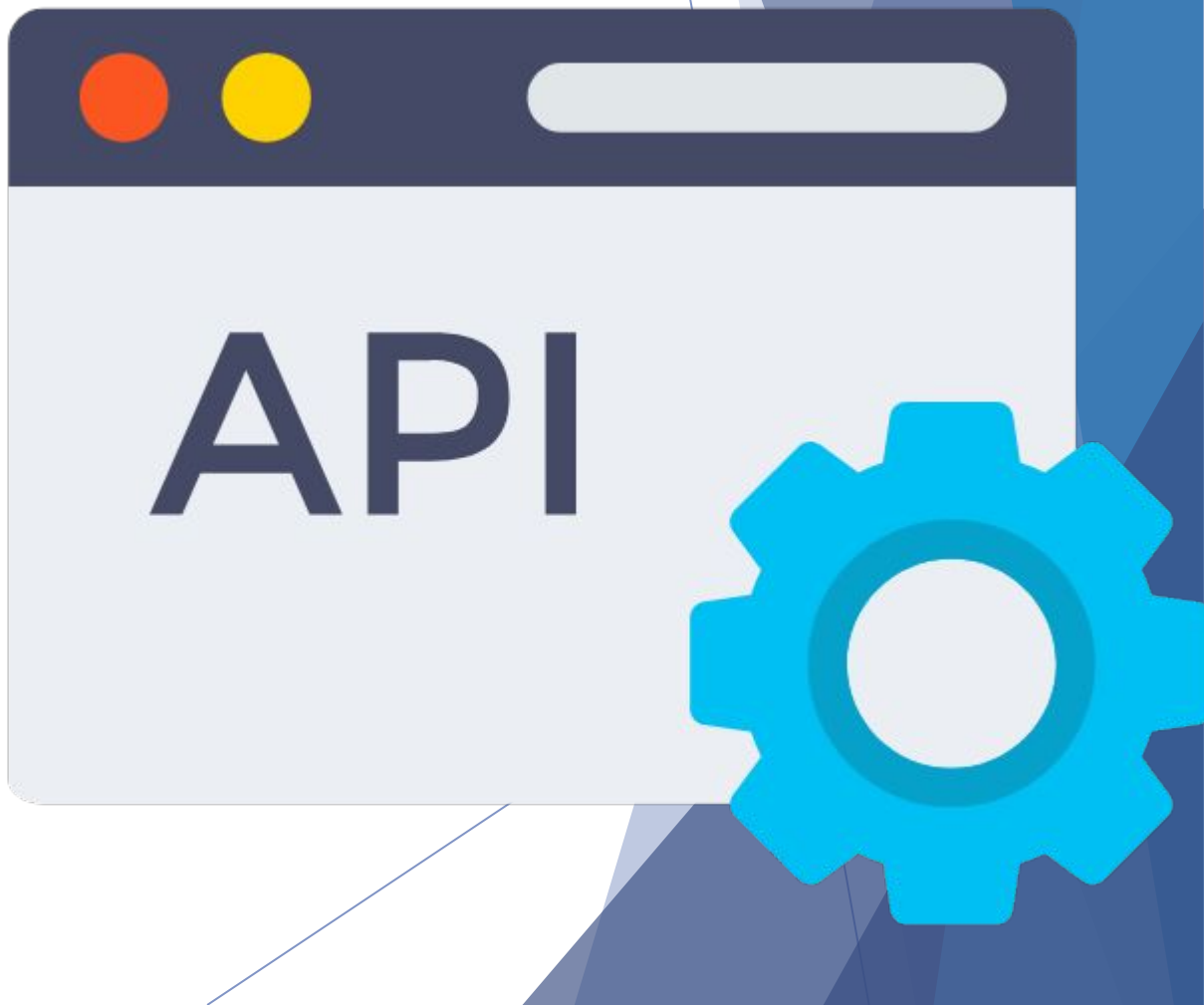
- ОС предоставляет пользователю **виртуальную машину**, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину.
- Например, чтобы считать или записать информацию на дискету, надо:
  - Запустить двигатель вращения дискеты
  - Управлять шаговым двигателем перемещения головки
  - Следить за индикатором присутствия дискеты
  - Выбрать номер блока на диске
  - Выбрать дорожку
  - Выбрать номер сектора на дорожке
  - и т.д.
- Все эти функции берет на себя операционная система.

Назначение ОС. ОС как система управления ресурсами

- Чтобы несколько программ могло работать с одним ресурсом (процессор, память), необходима **система управления ресурсами**.
- Способы распределения ресурса:
  - **Временной** - когда программы используют его по очереди, например, так система управляет процессором.
  - **Пространственный** - программа получает часть ресурса, например, так система управляет оперативной памятью и жестким диском.

Интерфейс прикладного программирования

- **API (Application Programming Interface)** - интерфейс прикладного программирования, .
- Интерфейс между операционной системой и программами определяется набором **системных вызовов**.
- Например, если пользовательскому процессу необходимо считать данные из файла, он должен выполнить команду системного вызова, т.е. выполнить прерывание с переключением в режим ядра и активизировать функцию операционной системы для считывания данных из файла.



Интерфейс прикладного программирования

- Рассмотрим наиболее часто применяемых системных вызовов стандарта POSIX. В POSIX существует более 100 системных вызовов.
- **fork** - создание нового процесса
- **exit** - завершение процесса
- **open** - открывает файл
- **close** - закрывает файл
- **read** - читает данные из файла в буфер
- **write** - пишет данные из буфера в файл
- **stat** - получает информацию о состоянии файла
- **mkdir** - создает новый каталог
- **rmdir** - удаляет каталог
- **link** - создает ссылку
- **unlink** - удаляет ссылку
- **mount** - монтирует файловую систему
- **umount** - демонтирует файловую систему
- **chdir** - изменяет рабочий каталог

Интерфейс прикладного программирования

- Рассмотрим вызовы Win32 API, которые подобны вызовам стандарта POSIX.
- **CreatProcess** (fork) - создание нового процесса
- **ExitProcess** (exit) - завершение процесса
- **CreatFile** (open) - открывает файл
- **CloseHandle** (close) - закрывает файл
- **ReadFile** (read) - читает данные из файла в буфер
- **WriteFile** (write) - пишет данные из буфера в файл
- **CreatDirectory** (mkdir) - создает новый каталог
- **RemoveDirectory** (rmdir) - удаляет каталог
- **SetCurrentDirectory** (chdir) - изменяет рабочий каталог

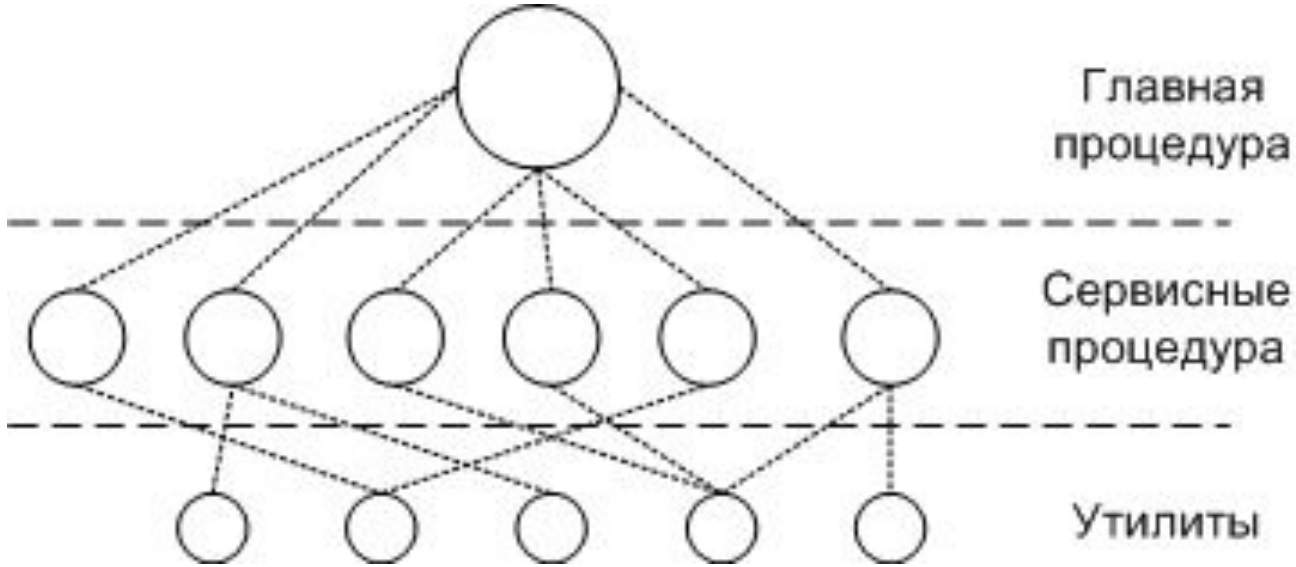
Интерфейс прикладного программирования

Интерфейс Win32 API позволяет программам работать почти на всех версиях Windows

Структура операционных систем. Монолитная система

- Структура системы:

1. Главная программа, которая вызывает требуемые сервисные процедуры.
2. Набор сервисных процедур, реализующих системные вызовы.
3. Набор утилит, обслуживающих сервисные процедуры.



Структура операционных систем. Монолитная система

- В этой модели для каждого системного вызова имеется одна сервисная процедура (например, читать из файла). Утилиты выполняют функции, которые нужны нескольким сервисным процедурам (например, для чтения и записи файла необходима утилита работы с диском).
- Этапы обработки вызова:
  - Принимается вызов
  - Выполняется переход из режима пользователя в режим ядра
  - ОС проверяет параметры вызова для того, чтобы определить, какой системный вызов должен быть выполнен
  - После этого ОС обращается к таблице, содержащей ссылки на процедуры, и вызывает соответствующую процедуру.

Структура операционных систем. Многоуровневая структура ОС

- Обобщением предыдущего подхода является организация ОС как иерархии уровней.
- Уровни образуются группами функций операционной системы - файловая система, управление процессами и устройствами и т. п. Каждый уровень может взаимодействовать только со своим непосредственным соседом - выше- или нижележащим уровнем. Прикладные программы или модули самой операционной системы передают запросы вверх и вниз по этим уровням.

Структура операционных систем. Многоуровневая структура ОС

Пример структуры многоуровневой системы

Уровни	Функции				
7	обработчик системных вызовов				
6	файловая система 1	...	файловая система n		
5	виртуальная память				
4	драйвер 1	драйвер 2	...	...	драйвер n
3	управление потоками				
2	обработка прерываний, управление памятью				
1	сокрытие низкоуровневой аппаратуры				

- Преимущества:
  - Высокая производительность
- Недостатки:
  - Большой код ядра, и как следствие большое содержание ошибок
  - Ядро плохо защищено от вспомогательных процессов

Структура операционных систем. Многоуровневая структура ОС

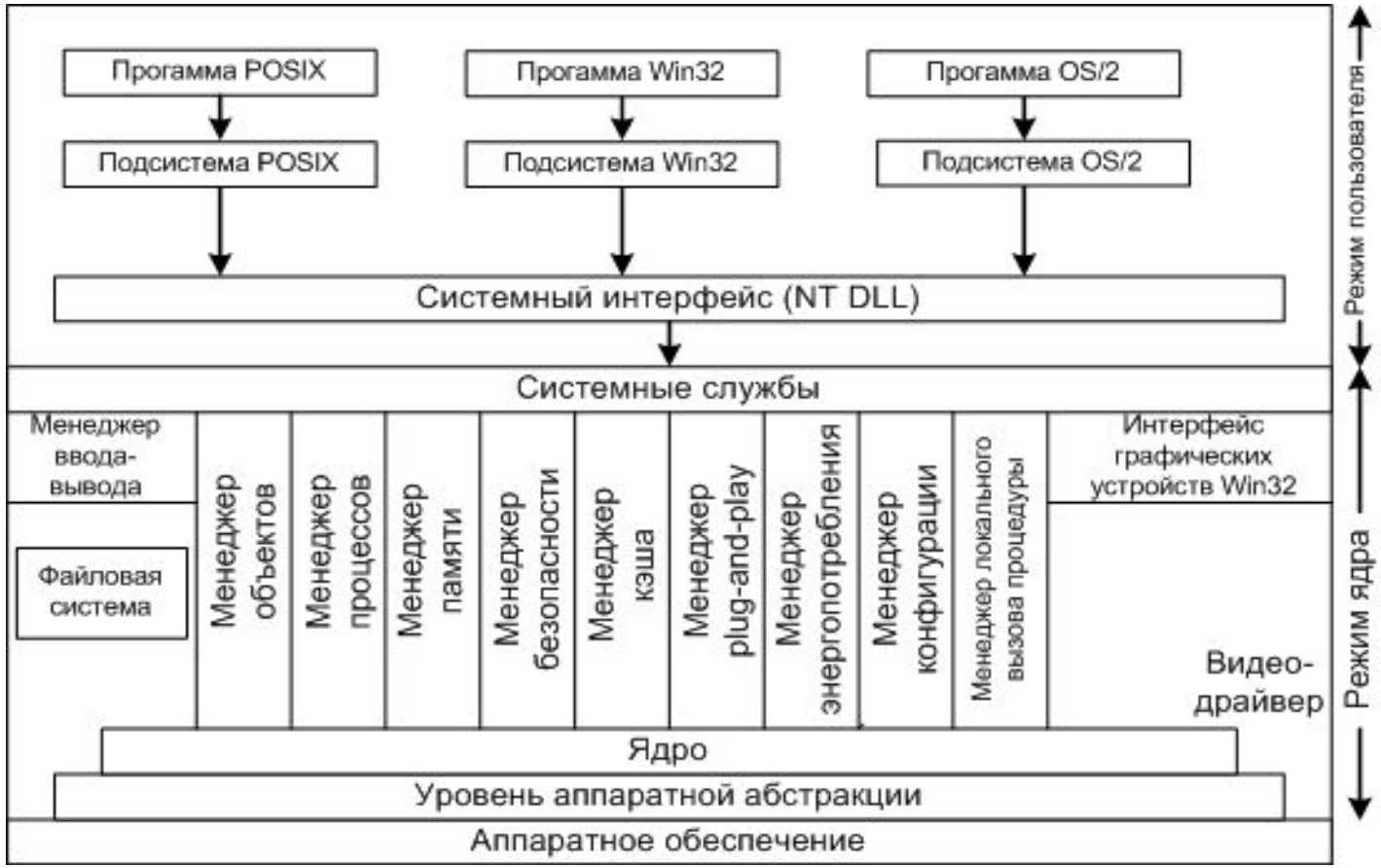
Пример реализации многоуровневой модели UNIX.

Структура операционных систем. Многоуровневая структура ОС

Ядро ОС UNIX

Структура операционных систем. Многоуровневая структура ОС

Пример реализации многоуровневой модели Windows



Структура операционных систем. Модель экзоядра

- Если предыдущие модели брали на себя максимум функций, принцип экзоядра - все отдать пользовательским программам.
- Например, зачем нужна файловая система? Почему не позволить пользователю просто читать и писать участки диска защищенным образом? Т.е. каждая пользовательская программа сможет иметь свою файловую систему.
- Такая операционная система должна обеспечить безопасное распределение ресурсов среди соревнующихся за них пользователей.

Структура операционных систем. Микроядерная архитектура (модель клиент-сервер)

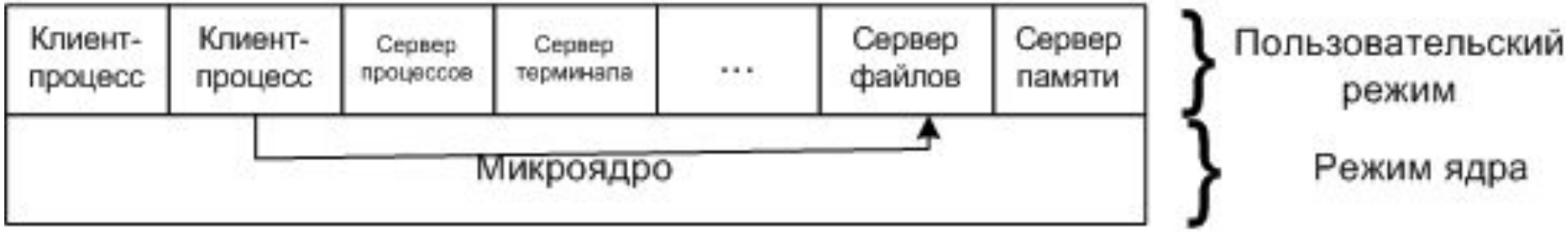
- Эта модель является средним между двумя предыдущими моделями.
- В развитии современных операционных систем наблюдается тенденция в сторону дальнейшего переноса задач из ядра в уровень пользовательских процессов, оставляя минимальное микроядро.
- В этой модели вводятся два понятия:

1. Серверный процесс (который обрабатывает запросы)
2. Клиентский процесс (который посылает запросы)

- В задачу ядра входит только управление связью между клиентами и серверами.

Структура операционных систем. Микроядерная архитектура (модель клиент-сервер)

Модель клиент-сервер



- Преимущества:
  - Малый код ядра и отдельных подсистем, и как следствие меньшее содержание ошибок.
  - Ядро лучше защищено от вспомогательных процессов.
  - Легко адаптируется к использованию в распределенной системе.
- Недостатки:
  - Уменьшение производительности.

Структура операционных систем. Обобщение сравнения моделей

Сравнения моделей.