

## **1. Операционная система. Процесс. Поток.**

**ОС** – организованная совокупность программ, которая действует, как интерфейс к аппаратной части ЭВМ. ОС обеспечивает пользователя набором средств, позволяющих упростить программирование, отладку и сопровождение программ.

**Процесс** – система действия, реализующая определённую функцию в вычислительной системе. Это логическая единица работы ОС. ОС выполняет решение задач, связанных с процессами, таких как управление, планирование, взаимодействие, синхронизация, распределение ресурсов и др. В современных ОС процесс воспринимается как динамический объект. Процессы в ОС реализуются по-разному и различаются своим представлением, способами защиты, именованиями и отношениями между ними. Процессы могут быть последовательными, параллельными и комбинированными.

**Поток** – единица выполнения. Это сущность внутри процесса, подлежащая планированию. Это отдельный счётчик команд. Поток отображает одну из возможно многих подзадач процесса.

## **2. Многозадачность. Многопоточность. Мультипроцессорная обработка. Ресурсы.**

**Многозадачность** – совместное использование процессора потоками, создание для пользователя иллюзии одновременного выполнения всех потоков.

**Многопоточность** – поддержка нескольких потоков внутри одного процесса.

**Мультипроцессорная обработка** – исполнение одного и того же кода ОС различными процессами, как на однопроцессорных, так и на многопроцессорных машинах.

**Ресурсы** – различные средства и возможности, необходимые для организации и поддержания процесса. Основная цель ОС – обеспечить простой, эффективный и бесконфликтный способ распределения ресурсов между пользователями.

## **3. Ресурсы. Классификация ресурсов. Категории ресурсов.**

**Ресурсы** – различные средства и возможности, необходимые для организации и поддержания процесса. Основная цель ОС – обеспечить простой, эффективный и бесконфликтный способ распределения ресурсов между пользователями.

**Классификация ресурсов:**

- физические (реально существующие) или виртуальные (мнимые, некая модель физического ресурса, не существующая в том виде, в котором она проявляет себя пользователю);
- пассивные или активные (способ выполнения действий над другими ресурсами);
- временные или постоянные.

**Категории ресурсов:**

- Процессорное время;
- Память (оперативная, дисковая, виртуальная);
- Периферийные устройства;
- Мат. обеспечение (функции для работы с данными, сервисные программы по управлению файлами, программы обслуживания ввода/вывода и др.).

## **4. Примеры операционных систем и их основные характеристики.**

**MS-DOS** – 16-тиразрядная, поддерживает только одного пользователя, однозадачный режим, программа полностью захватывает процессор, отсутствует пользовательский интерфейс.

**OS/2** – 32-х разрядная, совместная разработка IBM и MS, некоторые версии поддерживают приложения MS-DOS и Windows, применяется вытесняющая многозадачность, поддерживается многопоточность, позволяющая одновременно выполнять множество операций (для коммуникационных и мультимедийных программ, координирующих аудио и видео, текст и речь). Вначале была только для платформы Intel, затем была перенесена на RISC-процессоры.

**Windows 95** – поддерживает многозадачность в двух режимах: кооперативная многозадачность (для старых 16-тиразрядных приложений Windows) и многозадачный режим с вытеснением (для 32-х разрядных приложений Windows 95/NT), функционирует на платформе Intel и является однопроцессорной ОС.

**WINDOWS NT (Work Station)** многозадачный процесс с вытеснением. 32-х разрядный (гарантия, что каждая программа получит свою долю процессорного времени). Приложение своей изолированной области виртуальной памяти. Реализует многопроцессорную обработку (может распределять процессоры между программами).

**Mac OS** – графическая многооконная среда, работает только со своими приложениями, но есть эмулирующие программы для запуска приложений MS-DOS и Windows. Последние версии поддерживают кооперативную и вытесняющую многозадачность. Совместно с IBM, Motorola и Apple шла работа над созданием общей аппаратной платформы.

**Unix** – обычно работает только со специальным программным обеспечением, но есть средства для эмуляции среды Dos, Windows, Mac. Используется вытесняющая многозадачность, в последнее время реализована многопоточность. Работает на различных аппаратных платформах. Существуют RISC-процессоры, спроектированные специально для Unix.

**Windows 2000 (NT5)** – содержит лучшие черты 95, 98 и NT. Средства защиты NT (защита на уровне файлов, учётные записи) сочетаются с совместимостью Windows 98. Высокая аппаратная и программная совместимость, но поддерживается меньше устаревших устройств и программ. Минимальные ресурсы: P166 MHz, 32 МБ и выше. Основная файловая система NTFS5. Но может читать диски отформатированные в FAT32.

**Linux** – ориентирована на пользователей-программистов и относится к семейству Unix.

## 5. Эволюция операционных систем.

**Этапы:-ОС, ориентированные на перфокарты** – в каждый момент времени ЭВМ используется только для одной прикладной программы. Процессор простаивает, когда данные вводятся с перфокарт. Низкая производительность из-за устройства ввода/вывода.

**-ОС, ориентированные на магнитную ленту** – сократилось время ввода/вывода, но всё остальное осталось.

**-ОС пакетной обработки данных** – повышение производительности за счёт выполнения заданий без внешнего пользователя. Пакет – совокупность программ и данных, разделённых специальными метками. Согласование темпа поступления данных с темпом их обработки. Появляется оператор, который формирует пакет, и программа, которая считывает пакеты, запускает их на выполнение, отслеживает аварийные ситуации и другие функции. Эту программу можно назвать простейшей ОС. Производительность возросла, но пользователю приходилось ждать, пока будет обработан весь пакет.

**-ОС мультипрограммной пакетной обработки** – обеспечивает эффективное использование ресурсов несколькими пользователями. В оперативную память помещается несколько пользовательских программ. Время процессора разделяется между программами. Параллельно с работой процессора происходит обмен данными с несколькими внешними устройствами. Осуществляется автоматическая одновременная работа процессора, оперативной памяти и других устройств. Пользователь не имеет непосредственного доступа к ЭВМ. Появляется задача, какие функции возложить на аппаратуру, а какие – на ОС.

**-ОС с разделением времени** – пользователь имеет непосредственный доступ к ЭВМ. Основная цель – обслужить каждого пользователя, обеспечить допустимое время реакции ЭВМ на запрос пользователя. Вводится детерминизм (схема по мультиплексированию центральной памяти процессора среди программ, готовых на выполнение), то есть каждой программе отводится свой интервал времени.

**-Многoproцессорные ОС, многопоточные ОС** – ресурсы могут быть как сосредоточенными, так и распределёнными. Часто используются алгоритмы с вытеснением плюс всё, что раньше.

**-Микроядерные ОС** – в микроядре изолирована вся машинно-зависимая часть ОС(всё в перспективе).

## 6. Основные функции операционных систем.

**-Управление процессором** – решение задачи планирования, синхронизации, взаимодействия.

**-Управление ресурсами** – организация доступа к процессору, создание эффективного механизма деления времени.

**-Управление оп. памятью** – распределение, организация.

**-Управление периферийными устройствами.**

**-Организация доступа ко всей системе (защита математического обеспечения)**

**Функции с точки зрения пользователя:**

- организация интерфейса с одним или несколькими пользователями;
- поддержка операционного окружения пользовательских задач;
- обеспечение совместимости с другими системами;
- защита и безопасность информации.

## 7. Типы и свойства операционных систем.

**Типы:-ОС для ЭВМ общего назначения.**

**-ОС реального времени** – управление датчиками, широкий спектр устройств ввода/вывода, упрощённые алгоритмы обработки.

**-ОС портативных ЭВМ.**

**-ОС ЭВМ специального назначения.**

Наименование	Netware 4.1	Windows NT Server 4.0	Unix
Многозадачность	кооперативная	вытесняющая	вытесняющая
Защита памяти отдельного процесса	нет	есть	есть
Многопоточность	есть	есть	есть
Сертификация по C2	есть	рабочая станция по C2	разные варианты для версий
Поддержка алфавитно-цифрового терминала	нет	нет (?)	есть
Сетевой графический интерфейс	нет	у независимых разработчиков	есть
Логическая организация ресурсов	служба каталогов	домены	домены NIS
Быстродействие сетевой файловой системы	отличное	очень хорошее	низкое

**Свойства ОС:**

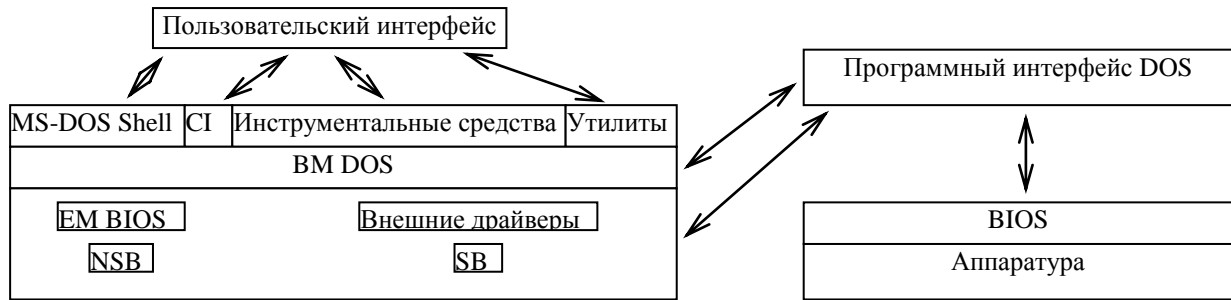
**-Надёжность** - ОС должна быть также надёжна, как и аппаратура, на которой она работает. Средства определения, диагностирования и исправления ошибок.

**-Защита** (внутренняя) от взаимного влияния пользователей друг на друга - минимизация порчи программ и данных.

**-Предсказуемость** – реакция ОС на запросы пользователя должна быть предсказуемой и не варьироваться слишком сильно.

- Удобство** – ОС должна облегчать работу пользователя и проектироваться с учётом факторов человеческой психологии.
- Эффективность** – эффективное распределение системных ресурсов.
- Общие системные услуги** – должны быть такими, чтобы пользователь при решении задач как можно реже обращался к дополнительным возможностям системных услуг.
- Гибкость** – при определении (установке) ресурсов для пользователя, гибкость при настройке системных операций для конкретного пользователя.
- Расширяемость** – возможность добавления новых модулей.
- Ясность (прозрачность)** – пользователь должен иметь возможность знать об ОС всё, что он захочет.

## 8. Структура ОС на примере MS-DOS. Назначение основных модулей.



**BIOS** - находится в ПЗУ и может считаться компонентом ОС. Содержит драйверы стандартных периферийных устройств, некоторые программы аппаратуры. Осуществляет инициализацию векторов прерываний нижнего уровня и считывает в память NSB.

**NSB** – Non-System Bootstrap, внесистемный загрузчик. Стартовый сектор физического жёсткого диска, является вторичным загрузчиком, считывает в память SB.

**SB** – стартовый сектор каждого логического диска, считывает в память EM BIOS, BM DOS и запускает EM BIOS.

**EM BIOS** – Extension Module BIOS. По существу, это файл «io.sys». Осуществляет определение состояния оборудования, конфигурацию DOS по «config.sys», инициализацию и переустановку некоторых векторов прерываний нижнего уровня. Запускает BM DOS.

**BM DOS** – Basic Module. Центральный компонент DOS, реализующий управление ресурсами и программой. Основу составляют обработчики прерываний верхнего уровня. Модуль считывает в память и запускает интерпретатор команд.

**Внешние драйверы устройств** – отдельные файлы для управления периферийными устройствами.

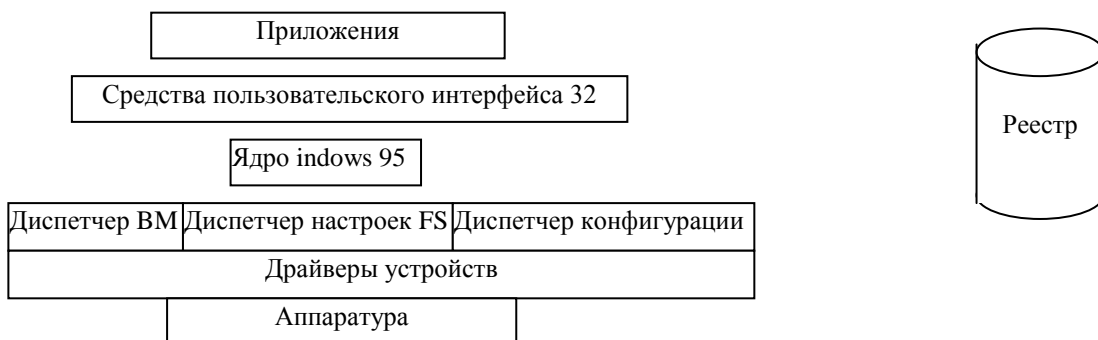
**CI** – интерпретатор команд. Файл «command.com». Отвечает за поддержку пользовательского интерфейса, осуществляет выполнение файла «autoexec.bat», состоит из двух модулей: резидентный (обработчики прерываний и код подгрузки транзитного модуля) и транзитный (может перекрываться в оперативной памяти выполняемыми подпрограммами, содержит исполнитель внутренних команд и загрузчик программы в оперативную память для выполнения).

**Утилиты** – обслуж. программы, предоставляющие пользователю сервисные услуги. Бывают диалоговые и недиалоговыми.

**MS-DOS Shell** – оболочка. Надстройка над CI, внешне напоминающая Windows.

**Инструментальные средства** – включают систему программирования MS-DOS (Qbasic, отладчик для тестирования и отладки исполнимых файлов)

## 9. Структура ОС на примере Windows 95. Назначение основных модулей.



В сравнении с Windows 3.1 появились новые 32-хразрядные драйверы устройств, файловая система, подсистема памяти, подсистема связи и мультимедиа и само ядро, включая управление памятью, процессами и т.д.

**Реестр** – центральная информационная база данных. Она упрощает структуру ОС, так как в принципе отпадает необходимость в файлах «autoexec.bat» и «config.sys». Реестр осуществляет централизованное хранение всей информации об аппаратных средствах.

**Диспетчер FS** – включает VFAT (виртуальную таблицу размещения файлов), CDFS (файловая система для CD-ROM), сетевой репозиторий.

**Ядро** – состоит из трёх компонент: USER (управляет клавиатурой, мышью, окнами, меню и т.д.), GDI (интерфейс графических устройств) и Kernel (обеспечивает базовые возможности ОС)

**Диспетчер конфигурации** – даёт гарантию, что каждое устройство сможет воспользоваться линиями запроса прерывания (IRQ) и другими ресурсами без конфликтов.

**Драйверы устройств** – применяется архитектура «универсальный драйвер – минидрайвер». Универсальный драйвер включает общую часть кода для конкретного класса устройств. Минидрайвер более простой и содержит инструкции для конкретного устройства.

## 10. Структура ОС на примере Windows NT. Назначение основных модулей.

Структура операционной системы на примере Windows NT. Назначение основных модулей.



Структура Windows NT состоит из двух частей:

### ❖ Защищённые подсистемы (серверы)

**Сервер** – отдельный процесс, память которого защищена от других процессов с помощью системы виртуальной памяти. Они предоставляют исполнительной системе пользовательский и программный интерфейсы, обеспечивает среду для выполнения приложений различных типов.

**Интерфейс прикладных программ (API)** – это набор процедур, которые вызываются прикладной программой для осуществления низкоуровневых операций. API реализуется на отдельном сервере Win32, OS/2 и других, что позволяет

устранить конфликты и дублирование в исполнительной системе.

**Подсистема Win32** предоставляет прикладным программам API Win32, реализует графический интерфейс и управляет вводом/выводом. Остальные подсистемы имеют свои API, но используют для получения пользовательского ввода и отображения результатов подсистему Win32. Подсистема защиты регистрирует правила контроля доступа на локальный компьютер, ведёт базу данных учётных записей пользователей.

❖ **Исполнительная система** – сама по себе является законченной ОС и выполняет функции низкого уровня. Имеет два набора функций: системные сервисы и внутренние процедуры. Компоненты исполнительной системы поддерживают независимость друг от друга.

- **Диспетчер объектов** – создаёт, поддерживает и уничтожает объекты, предоставляет системные услуги.
- **Справочный монитор защиты** – оберегает ресурсы ОС, обеспечивает защиту объектов и ведёт аудит (возможность обнаружения и регистрации важных событий, относящихся к защите) во время выполнения.
- **Диспетчер процессов** – создаёт, завершает и выводит информацию о процессах и потоках.
- **Средства локального вызова процедур (LPC)** – передаёт сообщения между клиентскими и серверными процессами, расположенными на одном и том же компьютере.
- **Диспетчер виртуальной памяти** – выделяет и управляет виртуальной памятью, обеспечивает собственное адресное пространство каждому процессу, отвечает за подкачку страниц в память, поддержку VDM и др.
- **Ядро** – реагирует на прерывания, направляет потоки на выполнение, осуществляет межпроцессорную синхронизацию, скрывает процессорные различия от остальной части ОС.
- **Диспетчер ввода/вывода** – реализует средства ввода/вывода, не зависящие от типа устройства.
- **Файловая система** – драйверы, принимающие запросы файлового ввода/вывода конкретного устройства.
- **Сетевой редиректор** – драйверы, принимающие запросы ввода/вывода для удалённых файлов, почтовых ящиков (именованных каналов), и пересылающие запросы сетевому серверу на другую машину.
- **Драйверы устройств** – применяется архитектура «универсальный драйвер/мини-драйвер». Универсальный драйвер включает общую часть кода для всех устройств конкретного класса, мини-драйвер содержит низкоуровневые инструкции для конкретного устройства.
- **Диспетчер кэша** – использует средства подкачки страниц диспетчера виртуальной памяти для автоматической записи информации на диск в фоновом режиме. Это повышает производительность

файлового ввода/вывода.

- **Слой, абстрагирования от оборудования (HAL)** – динамически подключаемая библиотека (DDL). Она изолирует исполнительную систему от особенностей аппаратных платформ разных производителей.
- **Реестр** – центральная информационная БД (начиная с Win95). Упрощает структуру ОС (заменяя AUTOEXEC.BAT, CONFIG.SYS, INI – файлы. Хранит в себе профили всех пользователей, данные о программах и типах документов, значения свойств для папок и значков программ, конфигурацию оборудования и данные об используемых портах. Имеет иерархическую древовидную структуру, состоящую из разделов, подразделов, кустов и записей.

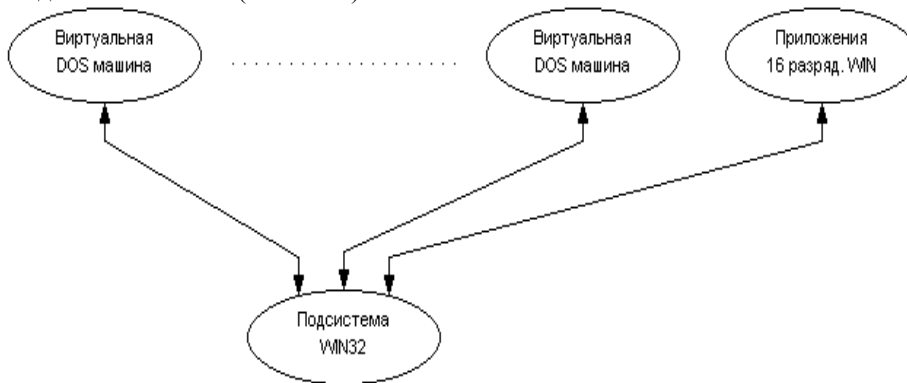
**В WINDOWS 2000** – новшества по структуре:

Распространяются не на ядро и не на пользовательский интерфейс, а на многочисленные важные подсистемы и службы.

**В WINDOWS 2003** – новшества касаются в основном подсистемы защиты.

## 11. Подсистема Win32. Виртуальные DOS машины. Схема VDM.

Подсистема Win32 (см. выше)



### Виртуальная DOS-машина (VDM)

– создаётся каждый раз, когда пользователь запускает приложение MS-DOS. Для каждого приложения создаётся собственное виртуальное адресное пространство, содержащее весь код MS-DOS и драйверы MS-DOS. В сущности, VDM – это виртуальная ОС MS-DOS, исполняющаяся на виртуальном компьютере с процессором типа Intel x86. Диспетчер обеспечивает

совместное использование всеми VDM одной копии 32-хразрядного кода.

NT работает с приложением MS-DOS, как и с другими потоками.

Блок обработки команд 32-хразрядной эмуляции DOS, драйверы	16Mб
Приложение MS-DOS	640Kб
16-тиразрядная эмуляция MS-DOS	
Исходный код MS-DOS	

Эмуляция 16-ти разрядной Win под Win32

Блок обработки команд
32-х – разрядный диспетчер окон и заглушки GDI
32-х – разрядная эмуляция DOS
Драйверы виртуальных устройств

16 Мб

Приложение Win 3.x
Диспетчер окон и заглушки GDI
Ядро Win 3.1
16-ти – разрядная эмуляция DOS

0

## 12. Подсистема Win32. Схема 16-тиразрядной Windows в виртуальном адресном пространстве.

Подсистема Win32 (см. выше)

Блок обработки команд	
32-хразрядный диспетчер окон и заглушки GDI	
32-хразрядная эмуляция MS-DOS	
Драйверы виртуальных устройств	
Приложения для Windows 3.x	16Mб
Диспетчер окон и заглушки GDI	
Ядро Windows 3.1	
16-тиразрядная эмуляция MS-DOS	

### 13. Процесс. Основное понятие. Дескриптор процесса. Виды групп информации дескриптора.

**Процесс** – для ОС представляет собой заявку на потребление системных ресурсов. Основной дейей процесса является способ управления программами в ходе их выполнения. Процесс создаётся, когда начинается выполнение задания пользователя, и разрушается при его завершении. Процесс, как логическая единица, предполагает два аспекта: выполняет операции, является носителем данных. То есть процессу присущи две части: программа, по которой он будет развиваться в активном состоянии и дескриптор процесса.

**Дескриптор процесса** – представляет собой информационную структуру, в которой сосредоточена управляющая информация, необходимая для системы планирования и управления процессами.

**Контекст процесса** – информация о процессе, необходимая непосредственно в активном состоянии.

**Группы информации (по функциональному назначению):**

- *Информация по идентификации* – содержит уникальное имя процесса, необходимое для реализации операций управления процессами, как поименованными объектами;
- *Информация о ресурсах*;
- *Информация о состоянии процесса* – необходима для определения возможности перехода в следующее состояние;
- *Информация о родственных связях* – используется для конкретного объекта для правильного окончания выполнения процесса, необходима для указания, какие ресурсы используются совместно, а какие – автономно;
- *Информация, необходимая для учёта и планирования процесса* – содержит ссылки на средства синхронизации между процессами, а также приоритет или место в соответствующей очереди.

**Очередь процессов** – дескрипторы отдельных процессов, объединённые в списки.

### 14. Граф существования процесса. Основные состояния процесса. Условия перехода из одного состояния в другое.



**Порождение** – подготавливаются все условия для выполнения.

**Готовность** – предоставляются все ресурсы, но процесс не исполняется, из-за внешних, по отношению к нему, обстоятельств.

**Активное состояние** – непосредственное использование процессора.

**Ожидание** – процесс может быть прерван по ряду причин: попытка получения ресурса или отка от ресурса, порождение, уничтожение или другие действия по отношению к другим процессам, возникновение прерывания (арифметическое переполнение, обращение к защищенной области оперативной памяти и др.), общая необходимость синхронизации между параллельными процессами.

**Окончание** – нормальное или аварийное завершение работы.

### 15. Планирование процессов. Планировщик. Двухуровневая система управления процессами. Типы планировщиков.

В мультипрограммных ОС на ресурсы могут претендовать сразу несколько пользователей, то есть существует множество независимых процессов, поэтому ОС должна осуществлять планирование.

**Планирование процессов** – управление распределением ресурсов между различными процессами путём передачи им управления согласно опеределённой стратегии.

**Диспетчеризация процессов** – выбор процесса и передачу на него управления.

**Диспетчер процессов** – часть ОС, отвечающая за диспетчеризацию процессов.

**Планировщик процессов** – набор функциональных модулей, выполняющих операции, необходимые для управления процессом. Он отвечает за постановку процессов в очередь на выполнение и управляет структурой этой очереди.

**Двухуровневая система управления процессами** (используется в большинстве ОС):

- *Долгосрочное планирование* – верхний уровень. На этот уровень выносятся действия, редкие в системе, но требующие больших системных затрат. процесс рассматривается как совокупность состояний по использованию программы на виртуальной машине.  
Состояние порождения для данного уровня – создание планировщиком требуемой виртуальной машины. Особенность данного уровня в том, что источник требований на порождение работы является внешним относительно процессора. При порождении осуществляются следующие действия: резервируются все необходимые ресурсы, резервируется память, создаётся структура данных.  
Состояние готовность – предоставлены все ресурсы виртуальной машины, кроме виртуального процессора.  
Состояние окончание – освобождены все ресурсы, которые были использованы для построения виртуальной машины.
- *Краткосрочное планирование* – нижний уровень. На этом уровне моделируется на процессоре деятельность виртуального процессора.  
Состояние активность – выполнение работы на виртуальном процессоре. Заявка на нижнем уровне ... на

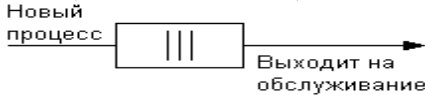
верхнем. Доступ любого задания к процессору осуществляется через системные программы планировщика и диспетчера.

#### Типы планирования:

- *Единый планировщик* - встроенный в ядро ОС, используется для всех заданий.
- *Разделённый планировщик* – планировочный модуль помещён в адресную часть каждой программы пользователя. Затем процесс осуществляет подпрограмму вызова, для постановки самого себя в очередь на исполнение. Это позволяет каждой программе иметь собственную стратегию планирования.

### 16. Классические дисциплины обслуживания очереди на исполнение процесса.

**FIFO(First In – First Out)** – минимизация дисперсии времени ожидания.



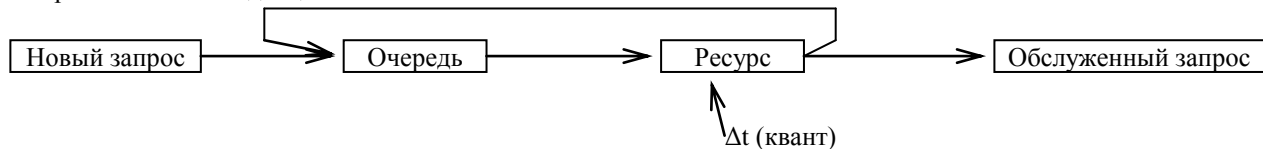
**LIFO (Last In – First Out)** – проста в реализации, является основой для построения стековой памяти.

Общим для LIFO и FIFO является то, что время ожидания запросов в очереди является одинаковым, независимо от характеристик процессора. Все процессы будут ожидать в очереди одинаково.



### 17. Алгоритм циклического планирования процессов.

Алгоритм основан на дисциплине FIFO.



Процессы выбираются из очереди и выполняются по порядку, начиная с первого. приоритет определяется линейным положением процесса в очереди. Недостаток: один процесс может занимать процессор длительное время. Для снятия этого недостатка каждому процессу выделяется интервал времени, квант. По истечении кванта процесс прерывается и помещается в конец очереди. Данный способ используется во многих ОС. Автоматически происходит дискриминация длинных и коротких запросов. Короткие запросы обслуживаются быстрее.

### 18. Алгоритм приоритетного планирования процессов. Статическое и динамическое приоритетное планирование.

**Приоритет** – число, характеризующее степень привилегированности процесса при использовании ресурсов (целое, дробное, больше нуля, меньше нуля).

Каждому процессу присваивается приоритет, который определяет его положение по отношению к другим процессам. Процесс с самым низким приоритетом называется холостым, так как он выполняет пустые инструкции. Приоритеты разбивают на группы ещё на этапе проектирования ОС. Количество групп выбирается таким образом, чтобы во время обработки не происходило окончание процессов в отдельных группах. Границы и число приоритетов могут быть различными.

**Статическое приоритетное планирование** – процессы при создании могут быть разделены по группам несколькими способами: исходя из запросов на ресурсы, согласно приоритету программы, которой принадлежит этот процесс, согласно оценке времени данной программы, по типу процесса, независимо от используемых ресурсов.

**Динамическое приоритетное планирование** – приоритет измеряется, как функция разницы между необходимой услугой и услугой практически полученной, то есть процесс перемещается по группам приоритетов в зависимости от израсходованного времени или ресурсов.

### 19. Алгоритм адаптивно-рефлексивного планирования процессов.

Алгоритм предполагает контроль над реальным использованием памяти. К началу планирования для каждого процесса устанавливаются ограничения на использование памяти и виртуального времени процессора. Далее ОС приспосабливается к рабочей области каждого процесса в течение всего времени его выполнения. Ограничения на память определяются оценкой текущего объёма памяти и оценкой вектора изменений этого объёма, полученного анализом работы процесса в течение предыдущего кванта времени. Если памяти достаточно, то выделяется временной интервал, причём его величина обратно пропорциональна максимальному объёму памяти, необходимой процессу. Идея подхода – ориентировать систему на процессы с минимальной рабочей областью.

## 20. Вытесняющие алгоритмы планирования процессов.

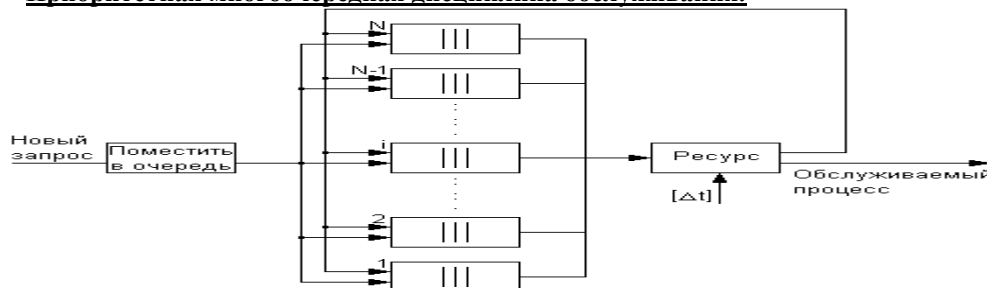
Алгоритм использует стратегию, при которой текущий процесс может быть вытеснен другим процессом. Например, после обработки прерывания на выполнение ставится процесс с более высоким приоритетом. При этом вытесненный процесс должен быть повторно обработан планировщиком. Стратегия с вытеснением может чередоваться со стратегией без вытеснения. Например, для каждого процесса вводится два флага: процесс может быть захвачен или нет, и процесс может захватить другой или нет.

## 21. Многоочередные дисциплины обслуживания процессов. Простая и приоритетная дисциплины.



Организуется N-очередей. Все запросы поступают в конец очереди. Первый процесс из очереди (i) поступает на обслуживание лишь тогда, когда все очереди от (i) до (i - 1) пустые, если кванта времени не хватило, то недообслуживаемый процесс поступает в конец очереди с номером (i+1). Если процесс выходит за пределы очереди N, то возможны два варианта: либо он обслуживается до конца, либо по циклическому алгоритму.

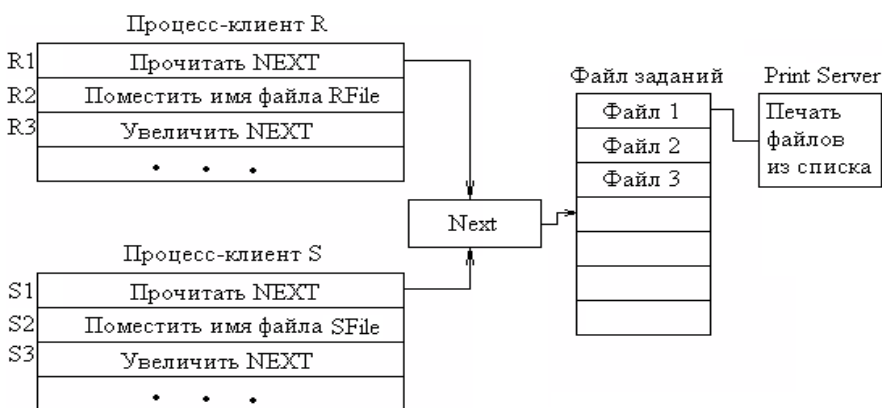
### Приоритетная многоочередная дисциплина обслуживания.



Поступающие процессы попадают в очередь в соответствии с имеющимися приоритетами. Эти приоритеты определяются параметрами процессов. Во многих ОС алгоритмы планировки построены, как с использованием квантования,

так и с использованием приоритетов. Например, в основе планирования может лежать квантование по величине или порядок выбора процесса из очереди определяется приоритетами процессов.

## 22. Проблемы, возникающие при взаимодействии процессов в мультипрограммных ОС.



Многие процессы находятся в зависимости от других. Поэтому необходимо обеспечить их взаимодействие. Необходимость в синхронизации возникает в программе печати файлов (Print Server).

Эта программа печатает по очереди все файлы, имена которых другие программы в порядке поступления записывают в специальный общедоступный файл заказов. Особая переменная NEXT также доступна всем процессам-клиентам и содержит номер первой свободной для записи

позиции файла заказов. Процессы-клиенты читают эту переменную, записывают в свободную позицию имя своего файла и увеличивают значение NEXT на 1.

**Эффект «гонок».** Пусть процесс R решил распечатать свой файл и прочитал NEXT=4, но поместить имя файла не успел, так как закончился квант времени. Очередной процесс S, желающий распечатать файл, прочитал то же самое значение переменной NEXT и поместил в 4 позицию имя своего файла и нарастил значение переменной на 1. После передачи управления процессу R он размещает в позицию 4 (это значение он прочитал раньше) свой файл и увеличивает переменную NEXT. В итоге не будет распечатан файл процесса S.

**Критическая секция** – это часть программы, в которой осуществляется доступ к разделенным данным.

**Взаимное исключение** – это исключение эффекта «гонок», обеспечивающее, чтобы в каждый момент в критической секции находилось не более одного процесса. Простейший способ его обеспечения - позволить процессу, находящемуся в критической секции, запрещать все прерывания. **Недостаток** – возможность крайне длительной загрузки системы процессом. Другой способ заключается в использовании блокирующих переменных. С каждым разделенным ресурсом связывается двоичная переменная (1 – ресурс свободен, 0 – ресурс занят). Однако операция проверки и установки блокирующей переменной должна быть неделимой, т.е. необходимо запретить прерывание на



протяжении всей этой операции. Недостаток – бесполезно тратится процессорное время при опросе блокирующей переменной.

**Взаимоблокировка** (дедлок (deadlock), клинч (clinch), тупик) – ситуация, когда конкуренция из-за ресурсов приводит к тому, что все процессы не могут продолжать работу. Для её избегания выделяют 3 метода восстановления: принудительная выгрузка ресурса, уничтожение процесса, откат.

Проблемы синхронизации и взаимного исключения могут быть решены путем использования управления событиями и памятью.

## 23. Механизмы синхронизации и взаимодействия процессов.

Процесс считается свободным, когда его последний поток завершается или заканчивается квант времени. Понятие свободен применяется и к другим объектам, которые имеют отношение к синхронизации. Объект поток свободен, когда завершается и исполняется поток. Объект события считается свободным, когда поток установлен потоком. Объект семафор считается свободным, когда счетчик семафора доходит до 0. Объект таймер – когда заданное время истекает за заданный интервал.

Взаимодействие между процессами может осуществляться с помощью обмена сообщениями или при помощи разделяемых (совместно используемых) переменных (участков памяти).

**Событие** – объект синхронизации, который используется для информирования потоков о том, что произошло некоторое событие. В ОС допускается одновременное возникновение нескольких событий.

Синхронизация подразумевает сигнализацию между процессами по определенному протоколу.

**Протокол** – набор правил и соглашений, не зависящий от времени и называемый событиями. В ОС допускается одновременное возникновение нескольких событий, каждому событию присваивают идентификатор, который называют флагом события. Этот флаг определяет возможность синхронизации. Количество флагов задается при проектировании. С каждым событием связано статическое слово и числовые значения. Если статус меньше 0, то это означает, что событие прекращено из-за ошибки и статическое слово содержит системный код ошибки. Если статус равен 0, то событие продолжается. Если статус больше 0, то это означает удачное завершение.

Простейший вариант синхронизации двух процессов:

А – процесс прародитель.

В – подчиненный процесс.

Процесс А инициирует процесс В с помощью события 1, а после завершения процесс В посылает статус и значение для события 1. При решении общих задач процессы должны иметь возможность обмениваться данными. Передаваемую последовательность данных называют сообщением. Обмен сообщениями между процессами имеет сложную структуру, поэтому такая связь между процессами создает новые проблемы, касающиеся адресации процессов, структуры планировщиков процесса и др. Асинхронность выполнения процессов усложняет взаимодействие, так как предполагается, что один процесс выполняется со скоростью, не зависящей от других. Принимающий сообщение процесс может быть не готов к приему и чтобы обеспечить взаимодействие, используют временный буфер. Обмен между процессами может быть разделен на два класса:

1 класс – разделяемые (совместно используемые) переменные: события и семафоры.

**Семафор** – счётчик числа доступных ресурсов (число сообщений в очереди, притом предельное значение семафора соответствует максимуму этой очереди). Это неотрицательная целая переменная, для которой определены две операции: Р и V. **Операция Р** – уменьшение семафора на 1, если это возможно. Если Р=0, то процесс, вызвавший операцию, ждет, пока уменьшение не станет возможным. **Операция V** – семафор увеличивается на 1.

2 класс – сообщения.

Используется в распределённых системах, когда процессы имеют собственную ОП и использовать распределённую память невозможно. Обмен информации между процессами имеет 2 ограничения со стороны ресурсов:

1. Объем сообщений не должен превышать емкости отведенного буфера.

2. Принимающий ресурс не может обрабатывать сообщение быстрее, чем они создаются передающим процессом.

**Возможно два подхода при обмене:**

1. Процесс может обмениваться только с процессами, имеющими общего прародителя.

2. Процесс может обмениваться с любыми процессами.

Также взаимодействие между процессами может быть реализовано с помощью использования общих участков памяти (в них записывается информация, которой процессы обмениваются). **Критическая секция** – это часть программы, в которой осуществляется доступ к разделенным данным. Необходимо обеспечить, чтобы в каждый момент времени к разделённой памяти обращалось (в критической секции находилось) **НЕ БОЛЕЕ** одной программы. Для этого существует **взаимное исключение** (запрет прерывания процесса в критической секции).

## 24. События. Семафоры. Сообщения. Их основное назначение.

Взаимодействие между процессами может осуществляться с помощью обмена сообщениями или при помощи разделяемых (совместно используемых) переменных (участков памяти).

**Событие** – объект синхронизации, который используется для информирования потоков о том, что произошло некоторое событие. В ОС допускается одновременное возникновение нескольких событий.

Синхронизация подразумевает сигнализацию между процессами по определенному протоколу.

**Протокол** – набор правил и соглашений, не зависящий от времени и называемый событиями. В ОС допускается одновременное возникновение нескольких событий, каждому событию присваивают идентификатор, который называют флагом события. Этот флаг определяет возможность синхронизации. Количество флагов задается при проектировании. С каждым событием связано статическое слово и числовые значения. Если статус меньше 0, то это означает, что событие прекращено из-за ошибки и статическое слово содержит системный код ошибки. Если статус равен 0, то событие продолжается. Если статус больше 0, то это означает удачное завершение.

Простейший вариант синхронизации двух процессов:

А – процесс прародитель.

В – подчиненный процесс.

Процесс А инициирует процесс В с помощью события 1, а после завершения процесс В посылает статус и значение для события 1. При решении общих задач процессы должны иметь возможность обмениваться данными. Передаваемую последовательность данных называют сообщением. Обмен сообщениями между процессами имеет сложную структуру, поэтому такая связь между процессами создает новые проблемы, касающиеся адресации процессов, структуры планировщиков процесса и др. Асинхронность выполнения процессов усложняет взаимодействие, так как предполагается, что один процесс выполняется со скоростью, не зависящей от других. Принимающий сообщение процесс может быть не готов к приему и чтобы обеспечить взаимодействие, используют временный буфер. Обмен между процессами может быть разделен на два класса:

1 класс – разделяемые (совместно используемые) переменные: события и семафоры.

**Семафор** – счётчик числа доступных ресурсов (число сообщений в очереди, притом предельное значение семафора соответствует максимуму этой очереди). Это неотрицательная целая переменная, для которой определены две операции: Р и V. **Операция Р** – уменьшение семафора на 1, если это возможно. Если Р=0, то процесс, вызвавший операцию, ждет, пока уменьшение не станет возможным. **Операция V** – семафор увеличивается на 1.

2 класс – сообщения.

Используется в распределённых системах, когда процессы имеют собственную ОП и использовать распределённую память невозможно. Обмен информации между процессами имеет 2 ограничения со стороны ресурсов:

1. Объем сообщений не должен превышать емкости отведенного буфера.

2. Принимающий ресурс не может обрабатывать сообщение быстрее, чем они создаются передающим процессом.

**Возможно два подхода при обмене:**

1. Процесс может обмениваться только с процессами, имеющими общего прародителя.

2. Процесс может обмениваться с любыми процессами.

Также взаимодействие между процессами может быть реализовано с помощью использования общих участков памяти (в них записывается информация, которой процессы обмениваются). **Критическая секция** – это часть программы, в которой осуществляется доступ к разделенным данным. Необходимо обеспечить, чтобы в каждый момент времени к разделённой памяти обращалось (в критической секции находилось) **НЕ БОЛЕЕ** одной программы. Для этого существует **взаимное исключение** (запрет прерывания процесса в критической секции).

## 25. Организация процессов в операционной системе UNIX.

Имеется два уровня – верхний и нижний. Распределение процессора производится по приоритетной схеме. Осуществление динамическим изменением приоритета в зависимости от текущих характеристик процесса. Например, на основе оценки соотношения времени тиспользуемому и тпрогнозируемому. Перераспределение времени происходит детерминировано с дискретностью один раз в секунду. Этот период называют квантом мультиплексирования процессора. Такой алгоритм обладает полезной отрицательной обратной связью (то есть, если процесс захватил много времени, то его приоритет снизится или наоборот). Для синхронизации используются события и специальные не поименованные пользователем файлы для передачи данных между двумя процессорами. Такие файлы могут наследоваться порожденными процессами. Файл прекращает свое существование, когда прекращают существовать все процессы, имеющий его дескриптор. Количество файлов таких не должно превышать максимально допустимое число дескрипторов. Файл используется только между родственными процессами. **Недостаток** – отсутствие взаимодействия между не связанными процессами.

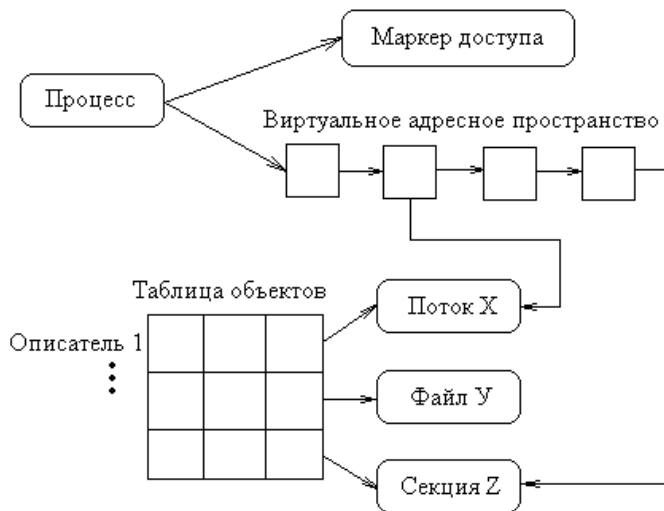
## 26. Процессы в Windows NT. Процесс как объект на высоком уровне абстракции. Атрибуты и сервисы процесса-объекта.

**Процесс** представляет собой один из типов объектов, который связан с другими объектами. К другим можно отнести: поток, секция (совместно используемая память), объект файл, объект порт (для передачи данных между процессами), объект маркер доступа (закодированный идентификатор с информацией о праве доступа), объект событие, объект семафор и др.

**На высоком уровне абстракции процесс состоит из:**

1. Исполнительной программы, которая определяет начальный код и данные процесса.
2. Закрытого адресного пространства (состоит из наборов адресов виртуальной памяти).
3. Системных ресурсов (семафоры, коммуникационные порты и файлы).

**Большинство процессов** – это процессы пользовательского режима. В режиме ядра в основном используется код ОС или осуществляется доступ к памяти.



Если процессу необходимо получить информацию о его маркере доступа, то он должен открыть описатель для своего объекта маркера. Виртуальное адресное пространство создаёт диспетчер. Из рисунка видно, что процесс открыл описатель одного из своих потоков (х), файла и секции совместно используемой памяти. Стрелки от адресного пространства означают виртуальные адреса, занятые стеком потока и объектом секциями. Процесс-объект имеет атрибуты и сервисы.

**Атрибуты** – идентификатор процесса, маркер доступа, базовый приоритет, процессорное сродство по умолчанию, размеры квот на ресурсы памяти, время выполнения, счетчик ввода-вывода, счетчик операций виртуальной памяти, коды завершения, порты исключения и отладки.

**Сервисы** – создать процесс, открыть процесс, запросить информацию процесса, установить

информацию процесса, текущий процесс, завершить процесс, выделить или освободить виртуальную память, защитить виртуальную память, чтение или запись в виртуальную память, блокировать и разблокировать виртуальную память, опросить виртуальную память и «сбросить» виртуальную память на диск.

Каждый процесс должен иметь как минимум 1 поток. Потоки 1го процесса могут выполняться независимо друг от друга.

## 27. Потоки в Windows NT. Роль потока в организации работы процесса. Назначение переключения контекста в многозадачной ОС.

**Процесс** – система действий, реализующая некоторую функцию в вычислительной системе. Это логическая единица работы ОС.

**Поток (нить)** – это сущность внутри процесса, подлежащая планированию. Она отображает 1 из возможно многих подзадач процесса.

**Многопоточность** – поддержка нескольких потоков внутри одного процесса.

Каждый процесс в Windows NT должен иметь как минимум 1 поток.

Пусть пользователь запустил приложения для работы с БД и сделал запрос на поиск данных с последующим сохранением отчета в файле. Пока идет выполнение, пользователь может сделать новый запрос. ОС представляет каждый из запросов как отдельные потоки внутри процесса приложения для работы с БД. Эти потоки могут выполняться независимо друг от друга.

Если число потоков превышает число процессов, то система переходит в режим многопоточности.

Контекст потока включает в себя:

- 1) уникальный идентификатор клиента
- 2) набор регистров состояния процессора
- 3) два стека для пользовательского режима и режима ядра
- 4) собственная область памяти

В NT используется вытесняющая многозадачность.

## **28. Вытесняющая многозадачность в Windows NT. Многопоточность и многозадачность.**

**Процесс** – система действий, реализующая некоторую функцию в вычислительной системе. Это логическая единица работы ОС.

**Поток** – это сущность внутри процесса, подлежащая планированию. Она отображает 1 из возможно многих подзадач процесса.

**Многозадачность** – это способ организации вычислительного процесса, при котором возможно совместное использование процессора потоками. Это создание для пользователя иллюзии одновременного выполнения всех потоков.

**Многопоточность** – поддержка нескольких потоков внутри одного процесса.

**Каждый процесс в WINDOWS должен иметь как минимум один поток.** Пусть пользователь запустил приложение для работы с БД и сделал запрос на поиск данных с последующим сохранением в файле. Пока идёт выполнение, пользователь может сделать новый запрос. ОС представляет каждый из запросов, как отдельные потоки внутри процесса приложения для работы с БД и эти потоки могут выполняться независимо друг от друга. Основные составляющие потока в исполнительной системе такие: **1-Уникальный идентификатор клиента. 2-Набор регистров в состоянии процессора. 3-Два стека (для пользовательского режима и режима ядра). 4-Собственная область памяти. Всё это контекст потока.**

Используется вытесняющая многозадачность, то есть поток может быть прерван по окончании кванта времени или когда на выполнение поступает поток с более высоким приоритетом. WINDOWS NT является ОС с симметричной мультипроцессорной обработкой, специально предназначенной для работы с ПК более, чем с одним процессором. Если число потоков превышает число процессоров, то оно поддерживает многозадачность. Использование двух процессоров для достижения многозадачности не всегда эффективно, например, в некоторых системах UNIX, когда один процесс создает другой процесс, система с одним процессором должна скопировать все адресное пространство первого процесса в адресное пространство второго. А в WINDOWS NT все содержимое не копируется, а используются механизмы совместного использования памяти.

## **29. Виды памяти. Основные функции управления оперативной памятью.**

Память в большинстве случаев является дефицитным ресурсом, который распределяется между ОС и пользователем. Особенности использования памяти:

1. Необходимый пользователю объём памяти часто превосходит имеющийся физический объём.
2. Каждый процесс считает, что отведённая ему память начинается с нулевого адреса и является непрерывной. Поэтому необходимо обеспечить возможность распределения физических адресов от 0 до (n) среди (k) процессов.

**Различают два вида памяти:**

- Физическая (соответствует реальной памяти ЭВМ).
- Логическая (соответствует набору ссылок или адресов, на которые может ссылаться программа).

**Основные функции управления памятью:**

**1. Отображение или перевод адресов логической памяти на адреса физической памяти.** Перевод осуществляется в следующих случаях:

- Абсолютная трансляция (выполняется компилятором или ассемблером при подготовке программы и генерировании ей абсолютных адресов).
- Статическая трансляция (проводится, если программа составлена в формате выполняемого модуля. Отдельные программы собираются вместе и им присваиваются адреса, установленные относительно некоторого установленного адреса памяти).
- Динамическая трансляция (Реальные адреса определяются ОС. Предполагается, что пространство памяти, отводимое процессу, может изменяться в ходе его работы).

**2. Расширение границ логического пространства памяти за границы физического пространства.**

Используется оверлейное программирование. Разные части программы используют одинаковый набор логических адресов. Существуют и другие методы, такие свопинг. В этих методах логическое адресное пространство программы соответствует её физическому адресному пространству.

**3. Разделение** (между программой пользователя и ОС) и распределение.

**4. Защита информации пользователя и ОС друг от друга.**

### 30. Система распределения оперативной памяти. Цели распределения. Основные решаемые задачи. Распределение памяти в двухуровневой ОС.

При распределении было освоено две цели:

1-В машинах, где ОП является дефицитным ресурсом целью является оперативная загрузка ОП.

2-Если обеспечение ОП решённый вопрос, то целью является достижение пользовательской эффективности. Стараются обеспечить удобный доступ к данным ОП. Необходимо отметить, что потребность пользователей всегда опережает реальные возможности. Решением является подход, при котором пользователи должны работать с ОП не на физическом уровне, а на некотором более высоком логическом (виртуальном) уровне. Если ОС двухуровневая на верхнем уровне память распределяется статически, а на нижнем динамически.

На каждом уровне решаются три взаимосвязанные задачи:

- Учёта памяти.
- Выделение памяти.
- Возврата памяти.

Ядро ОС обычно размещают в начальных адресах памяти. Существует множество алгоритмов распределения памяти – основная цель у них одна – минимизация размеров и количества пустых участков памяти.

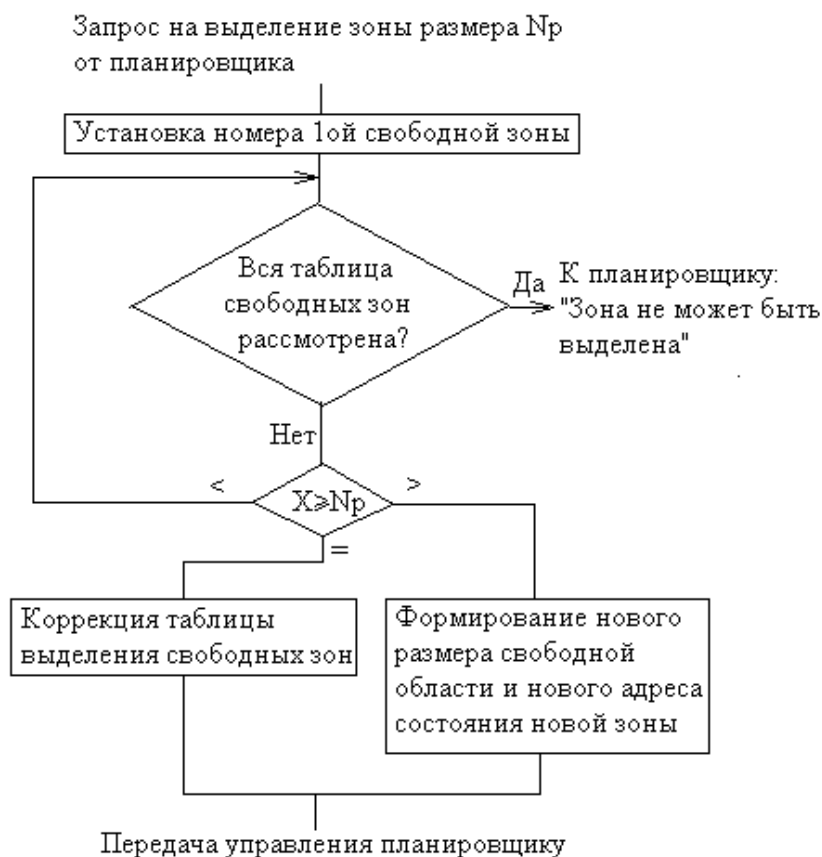
### 31. Система распределения оперативной памяти. Алгоритмы, основанные на выделении непрерывной единственной зоны.

1)ОП разбивается до начала работы ОС, причём на фиксированное число зон. Зоны могут быть разными по размеру, причём каждый запрос может выполняться только в пределах конкретной зоны.

2)Второй вариант размеры зон могут меняться при работе, например, зона может отводиться при долгосрочном планировании каждый раз. Могут формироваться новые зоны из свободных участков (дыр). Но минимальный размер дыры при этом ограничен. Основное условие – нельзя выделить зону, размер которой больше размера ОП.

3)Алгоритмы оптимального размещения.

### 32. Система распределения оперативной памяти. Алгоритм оптимального размещения.



Для запроса выделяется свободный участок минимально возможного размера. Идея алгоритма: минимизировать объём свободного пространства, который остаётся после каждого распределения.

$$V_z^2 = V_z^1 - V_x^1 \longrightarrow V_z^3 = V_z^2 - V_x^2 \quad \text{и}$$

т.д., где  $V_z$  – размер свободного участка и  $V_x$  – требуемый размер участка.

Каждый раз остаток памяти  $V$  сравнивается с некоторой пороговой величиной и вырабатывается решение об эффективности такого решения. Центральное место занимает способ учёта пространства зон.

Один из способов: все дыры в зоне объединяются в список определённого вида. Элемент списка – сама дыра, в начальных и конечных адресах которой строится структура данных, которая описывает характеристики дыры и ссылки на другие элементы списка.

Списки дыр могут быть упорядоченными или неупорядоченными. Чаще всего они упорядочены в порядке возрастания начальных адресов дыр. Преимущество: легко обнаружить смежные дыры, которые можно объединить в одну. Списки могут быть

двунаправленными, что позволяет сократить среднее время поиска.

Поиск на практике осуществляется по принципу «первая подходящая дыра». Причём, поиск всегда начинается сначала списка. Существует другой принцип, который называется «самая подходящая дыра».

### 33. Управление оперативной памятью. Свопинг.

Эффективность использования ОП можно увеличить за счёт **внешней памяти**. При нехватке ОП ненужные в данный момент разделы могут копироваться на диск. Это называется **откачкой**. Обратный процесс копирования называется **подкачкой**. Совокупность данных процессов называется **свопингом**. Для организации свопинга требуется **планировщик памяти**. Сложность в том, что один раздел может использоваться несколькими процессами. Свопинг позволяет заново распределить память для процесса, не запуская его с самого начала.

**Потребность в перераспределении возникает по следующим причинам, например:**

- 1-Появляется возможность выполнить больше малоактивных процессов, чем может одновременно разместиться в ОП.
- 2-Позволяет освободить память, занимающую процессом, который требует вмешательства пользователя.
- 3-Позволяет более эффективно использовать другие ресурсы, кроме ОП, например, с более высоким приоритетом.
- 4-В многопользовательских системах, когда используется один и тот же код. В свопинге могут участвовать только области данных пользователя.

### 34. Управление оперативной памятью. Использование оверлеев.

Программа пользователя разбивается на главный (корневой сегмент) и один или более не основных сегментов, фиксированной длины (их называют оверлеями).

Корневой сегмент резидентно находится в памяти, а оверлеи находятся на диске, и при необходимости подгружаются в память. Область памяти программы пользователя зарезервированная под оверлеи называется оверлейной группой. Внутри группы оверлей фиксированной длины, но различные группы могут иметь разные длины.

**Оверлейные системы.**

**По принципу организации и способу загрузки оверлейные системы классифицируются:**

**1-Автоматическая система** (деление программы на оверлеи решает ОС).

**2-Полуавтоматическая** (когда пользователь сам может решать, как должна быть разделена программа, а ОС, как загружать оверлеи).

**3-Программная** (пользователь не только определяет, как разделить программу, но и выдаёт запрос на загрузку оверлеев).

Программа может быть разделена на оверлеи в виде дерева, где корневой сегмент представляет собой нулевой уровень дерева. Оверлей, вызываемый из корневого сегмента, принадлежит первому уровню дерева и т.д.

### 35. Управление оперативной памятью. Схема механизма физической адресации.

**Схема механизма физической адресации.**

Форма задания адреса полностью определяется механизмом доступа к элементам хранения. Механизм механической адресации – это простейший способ доступа, реализованный аппаратно.

**Адрес** – это число, которое однозначно определяет номер требуемого элемента хранения. Размер адресного пространства равен объёму конкретного вида памяти.

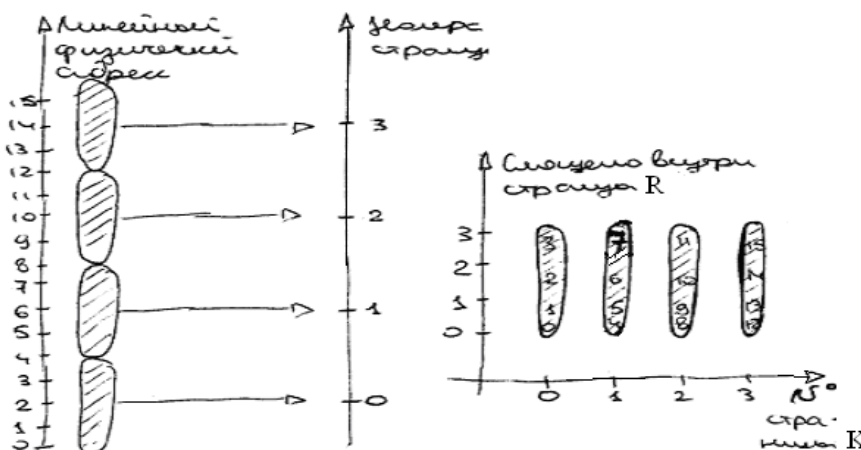
Данный механизм служит основой для более сложного доступа, который реализуется, как правило, в программно-аппаратной форме. В этом случае адрес может задаваться уже в форме, удобной для пользователя, и отличаться от физических адресов конкретного вида памяти. Такие адреса называются виртуальными, а пользователи имеют доступ к программному слою физической адресации.

### 36. Организация виртуальной оперативной памяти. Схема структурирования фиксированными страницами.

Различают два класса схем структуризации адресного пространства:

1-Схемы страничной структуризации.

2-Схемы сегментной структуризации.



**Схема структурирования фиксированными страницами.** Если размер адресного пространства кратен 2, то и размер страницы также выбирают кратным 2. Это позволяет упростить механизм преобразования адресов. В каждую страницу входят одинаковое число адресов  $L$  (размер страницы). Для перехода из страничного в исходное одномерное адресное пространство используется выражение  $A = K \cdot L + R$  для вычисления адреса. Адрес в страничном виде задан парой  $(K, R)$ . Чтобы перейти в непрерывный адрес нужно выполнить два действия: умножение и сложение.

Можно обойтись одним действием, если воспользоваться двоичным представлением номера страницы и смещения. Из двух двоичных чисел с помощью конкатенации получают третье число.

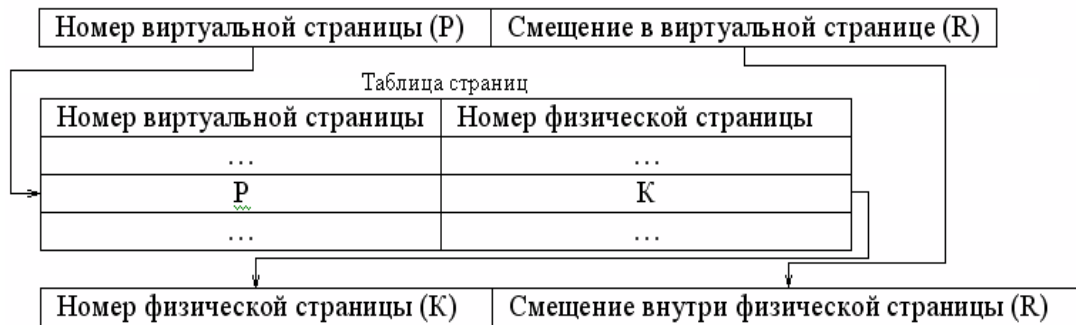
**Механизм работы страницы:** во время загрузки процесса часть его виртуальных страниц помещается в ОП, а остальные на диск. Причём, смежные виртуальные страницы не обязательно располагаются в смежных физических страницах. ОС создаёт информационную структуру для каждого процесса, которая называется таблица страниц. В этой таблице устанавливается соответствие между номерами виртуальных и физических страниц (только для страниц, загруженных в ОП) или делается отметка о том, что виртуальная страница выгружена на диск.

В таблице страниц содержится следующая управляющая информация:

1. Признак модификации страницы.
2. Признак запрета на выгрузку.
3. Признак обращения к странице (используются для подсчёта числа обращений за определённый период времени).

Время преобразования виртуального адреса в физический в основном определяется временем доступа к таблице страниц.

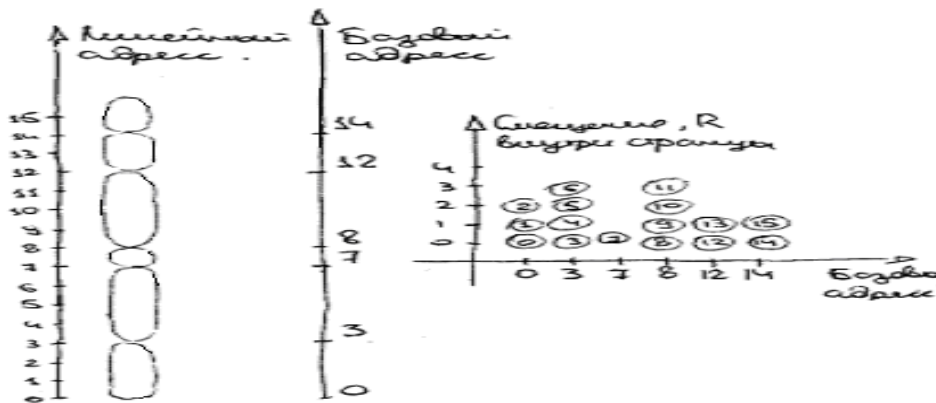
Страничная организация может быть реализована в упрощённом варианте (без выгрузки страниц на диск). В этом случае фрагментация уменьшается, так как программа может загружаться в несмежные области, и не происходит выхода за пределы физической памяти.



Механизм преобразования виртуального адреса в физический при страничной адресации. При обращении к ОП выполняются следующие действия: 1. Зная

адрес таблицы страниц (обычно хранится в регистре адреса таблицы страниц), зная номер виртуальной страницы (старший разряд адреса) и длину записи в таблице (системная константа), находят нужную запись. 2. Считывается номер физической страницы. 3. К номеру физической страницы присоединяется смещение (младшие разряды).

### 37. Организация виртуальной ОП. Схема структурирования переменными страницами.



Максимальный размер страницы  $L=4$ . Вместо номеров страниц - их базовые адреса. Здесь отсутствует взаимоднозначное соответствие между структурированными и непрерывными адресами. Для получения значения непрерывного адреса ( $A_i, R$ ) необходимо использовать выражение  $A=A_i+R$ . По значению непрерывного адреса нельзя сказать, к какой странице

он принадлежит.

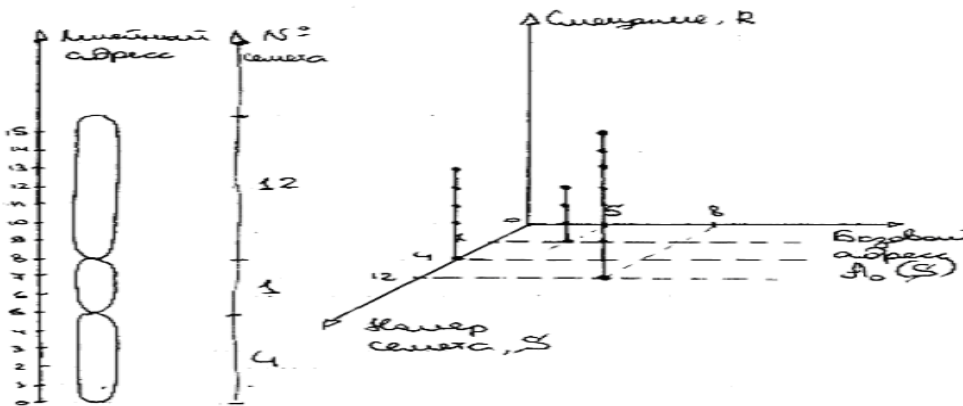


Механизм преобразования виртуального адреса в физический при страничной адресации. При обращении к ОП выполняются следующие действия: 1. Зная адрес таблицы страниц (обычно хранится в регистре адреса таблицы страниц), зная номер виртуальной страницы (старший разряд адреса) и длину записи в таблице (системная константа), находят нужную запись. 2. Считывается номер физической страницы. 3. К номеру физической страницы присоединяется смещение (младшие разряды).



### 38. Организация виртуальной оперативной памяти. Схема сегментной структуризации.

**Особенности:** сегменты формируются различных размеров. Причём их номера не упорядочены и могут быть произвольными целыми числами. Каждому сегменту ставится в соответствии его базовый адрес. В трёхмерном представлении добавляют смещение внутри сегмента. Но на практике задают не три координаты, а две (S, R), тогда для перехода к непрерывному адресу необходимо выполнить две операции: 1. Сегменту с номером S присваивают базовый адрес  $AS_0$ , а затем применяется принцип



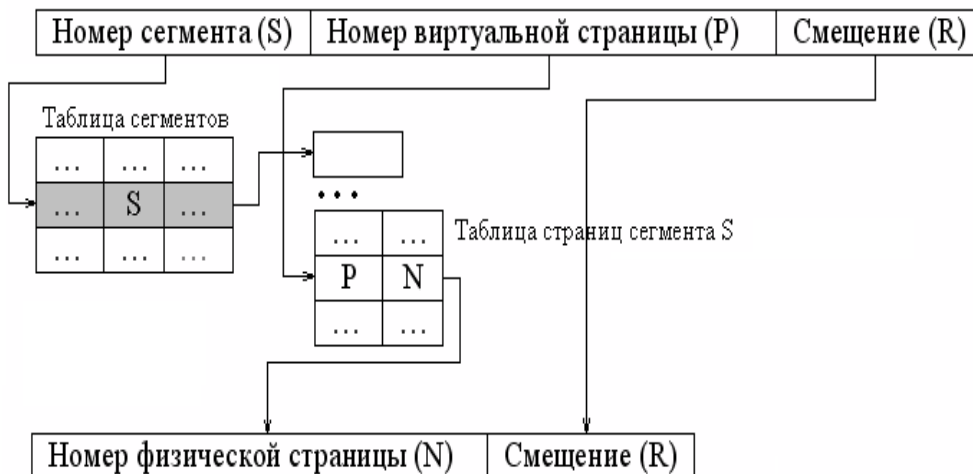
принцип база+смещение  $A = AS_0 + R$ .

**Механизм работы:** отдельный сегмент может представлять собой подпрограмму, массив данных и др. То есть всё, что определяется программистом. Иногда сегментация программы выполняется компилятором по умолчанию. При загрузке процесса часть сегментов помещается в ОП, при этом может использоваться алгоритм оптимального размещения, оставшаяся часть размещается на диске. Сегменты могут занимать несмежные участки, во время загрузки создаётся таблица сегментов процесса, в которой для каждого сегмента указываются: 1. Начальный физический адрес сегмента в ОП. 2. Размер сегмента. 3. Правила доступа. 4. Признак модификации. 5. Признак обращения к данному сегменту за последний квант времени.

Если виртуальное адресное пространство нескольких процессов выполняют один и тот же сегмент, то в таблице сегментов этих процессов делаются ссылки на один и тот же участок ОП, в котором находится сегмент в единственном экземпляре. Недостатком данного метода является фрагментация на уровне сегментов и более медленное преобразование адреса по сравнению со страничной организацией.

### 39. Организация виртуальной оперативной памяти. Схема сегментно-страничной структуризации.

Шаги:



1. Исходное пространство структурируют фиксированные страницы.

2. Сегмент рассматривается как некоторая непрерывная последовательность номеров страниц. Размер сегмента - количество страниц.

3. Каждый сегмент имеет свой уникальный номер S.

4. В пределах данного сегмента происходит перенумерация страниц, начиная с 0 и по возрастанию.

5. Сегменту назначается базовый адрес  $AS_0$ . В итоге адрес указывается с помощью четырех координат (S,  $AS_0$ , R', R). S – номер сегмента,  $AS_0$  – базовый адрес сегмента, R' – номер страницы в пределах сегмента, R – смещение в пределах страницы.  $A_{R6} = AS_0 + R' \cdot L$ . Далее, если размер страницы был кратен 2, то к базовому адресу страницы применяют операцию конкатенации (присоединяют значение смещения).

Основные цели страничной сегментной организации: страничная организация ориентирована в первую очередь на удовлетворение нужд системы. Позволяет улучшить использование ОП с уменьшением объёма пересылок между рабочей и активной средами. Сегментная организация ориентирована на пользователя, на использование сложных многомодульных программ в мультипрограммной системе.

**Механизм преобразования виртуального адреса в физический при сегментно-страничной организации.** Для каждого сегмента создаётся своя таблица страниц. Адрес таблицы записывается в специальный регистр процессора, когда процесс становится активным.



#### 40. Задачи управления виртуальной памятью.

1-Размещение. В адресном пространстве ОП выбираются страницы и сегменты, на которые будут отображаться некоторые страницы и сегменты виртуального адресного пространства. Сложность в том, что размер виртуального существенно больше линейного адресного пространства ОП. Главная задача уменьшить фрагментацию при размещении.

2-Перемещение. Например, из архивной среды хранения необходимо перенести информацию какой-либо виртуальной страницы и отобразить страницу ОП.

3-Преобразование. Необходимо найти абсолютный адрес в рабочей среде хранения его виртуального адреса в соответствии с функцией преобразования.

4-Замещение. Необходимо выбрать среди страниц адресного пространства кандидата на перераспределение.

#### 41. Файловая система. Задача файловой системы. Функции файловой системы.

**Файл** – это последовательная целостная поименованная совокупность на внешнем носителе, на которого наложена структура.

**Файловая система** – это часть ОС, которая обеспечивает выполнение операций над файлами.

**Традиционная(реляционная) задача файловой системы** это:

- Скрытие от пользователей реального расположения информации на физическом уровне.
- Обеспечение независимости программ от конкретной конфигурации ЭВМ.

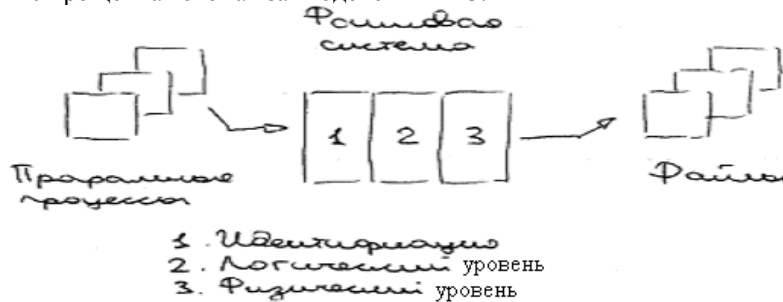
**Функции файловой системы:**

1-Реализует метод организации данных. 2-Выполняет перевод логических характеристик в физические.

3-Защищает пользователя от случаев сбоев оборудования. 4-Позволяет многим пользователям разделять одно устройство. 5-Обеспечивает возможность восстанавливать файлы. 6-Защищать от несанкционированного доступа и т.д.

#### 42. Схема взаимодействия файловой системы. Уровни ФС.

Упрощенная схема взаимодействия ФС:



На уровне идентификации модули выполняют следующие действия:

1. По символьному имени файла определяется его уникальное имя (в тех ОС, где один файл может иметь несколько символьных имён. В MS-DOS одно символьное имя и оно же уникально).
2. По уникальному имени определяются атрибуты файла.
3. Сравниваются полномочия пользователя

или процесса с правами доступа к файлу.

**На логическом уровне:** определяют координаты логической записи в файле. Алгоритм работы зависит от конкретной логической модели организации файла.

**На физическом уровне:** определяется номер конкретного физического блока, который содержит требуемую логическую запись.

**Метод доступа к файлу** – это способ адресации к составным его элементам на основе логической структуры файлов.

#### 43. Характеристики файлов. Типы доступа к файлу.

**1-Имя файла.** В старых ОС MS-DOS 6.22 и ниже используется формат <8.3> и максимальная длина пути 80 символов. В современных ОС WINDOWS 95 и выше используются длинные имена до 255 символов и длина пути до 260 символов.

**2-Расширение файла.** ОС должна распознавать стандартный набор расширений.

**3-Атрибуты файлов.** Специфицируют тип файла, защиту и способ буферизации (пароль для доступа, владелец файла, создатель, признаки только для чтения, скрытый, системный, архивный, временный, текущий размер и др.).

**4-Тип файла.** Может быть:

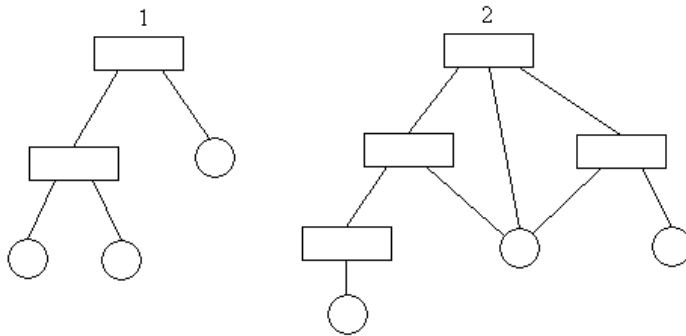
- *Сегментированный* (обеспечивает структуру файла с произвольным доступом и может иметь неограниченный размер).
- *Непрерывный* (обеспечивает один непрерывный блок и используется для быстрого непосредственного доступа).
- *Последовательный* (обеспечивает последовательную организацию данных, и файл может расти неограниченно).

**По другой классификации файлы бывают:** 1-Обычные (текстовые, двоичные). 2-Специальные (например, для операции ввода-вывода. Блоко-ориентированные, байто-ориентированные). 3-Файлы каталоги (справочники, они содержат список файлов и их характеристики).

**По типу доступа классифицируются:**

1-На чтение. 2-На обновление (модификацию имеющихся записей). 3-На запись (модификацию старых и добавление новых). 4-На удаление. 5-На изменение атрибутов и т.д.

#### 44. Логическая организация файловой системы.



Выделяют две основные иерархические модели:

1. Дерево (MS-DOS).
2. Сеть (UNIX).

Пользователь работает с ФС на уровне логических записей. **Логическая запись** – это наименьший элемент данных, который доступен пользователю для выполнения различных операций.

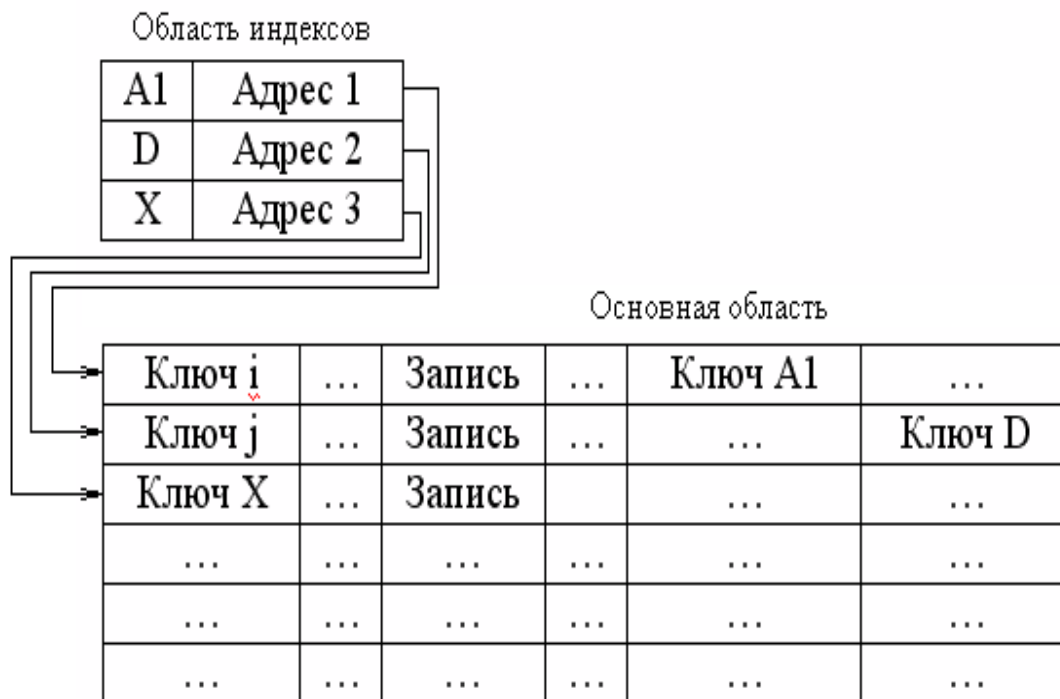
#### 45. Логическая организация файла. Файлы с последовательной структурой.

Пользователь работает на уровне логических записей. Логическая запись – наименьший элемент данных, доступный пользователю для выполнения разных операций.

**Файлы с последовательной структурой.** Файл рассматривается как одномерный массив составных элементов, называемых записями. Длина логических записей может быть как постоянной, так и переменной. Каждая логическая запись характеризуется своим порядковым номером в составе файла. Доступ к файлу последовательный (то есть, после обработки  $i$ -ой записи доступна следующая  $i+1$ -я запись). Для организации такого доступа достаточно иметь один указатель на текущую запись.

#### 46. Логическая организация файла. Файлы с индексно-последовательной структурой.

Пользователь работает на уровне логических записей. Логическая запись – наименьший элемент данных, доступный пользователю для выполнения разных операций.



**Файлы с индексно-последовательной структурой.** Существует ряд методов, основанных на идентификации записей файла по некоторому ключу (индивидуальному отличительному признаку). Структура файла усложняется, но сокращается число обращений к диску. Кроме данных дополнительно вводят служебную учётную информацию. Дорожки на

нескольких поверхностях HDD образуют цилиндры. Данная структура строится так, что поиск элемента файла проводится сначала в прямом, потом – в последовательном порядке. Все записи упорядочиваются по значению ключей, далее выделяют группы записей, ключи которых расположены подряд в файле и могут храниться в пределах одной дорожки на диске. Для более быстрого поиска таких групп, строят специальную структуру - индекс (например, индекс дорожки). Каждый элемент индекса описывает отдельную группу записей. Индекс может содержать максимальный ключ в группе и ссылку на начальную запись в группе. По индексу находят начало первой записи требуемой группы, а затем внутри группы последовательным анализом ключей находят требуемую запись.

Индексно-последовательная структура организации файла учитывает аппаратную реализацию. На жестком диске выходим на дорожку (прямой доступ) и добираемся до записи (последовательный доступ). Для реализации такого вида доступа необходимо, чтобы информация была упорядочена по ключам, отсюда появляется **недостаток**: Проблема расширения файла во время работы с ним. Для решения проблемы вводят специальную область переполнения, куда заносят записи, динамически вводимые файлы. А затем, из основной области устанавливают ссылки на требуемые элементы области переполнения.

## **47. Логическая организация файла. Библиотечная структура файлов.**

### **3- Библиотечная структура файлов. Имеется два уровня:**

- *Учётный.*
- *Информационный.*

Файл составляется совокупностями последовательных наборов данных, где каждый набор имеет собственное имя в составе данного файла. Такие наборы называют разделами. Расположение разделов не упорядочено и записываются в порядке поступления. Распределение раздела фиксируется в каталоге учётного уровня, а сами элементы каталога расположены в алфавитном порядке.

## **48. Физическая структура файла. Способы размещения информации. Непрерывное размещение. Достоинства и недостатки.**

Расположение информации на конкретном носителе чаще всего значительно отличается от логической упорядоченности. Преобразование логической в физическую структуру осуществляется на основе информации сосредоточенной в каталогах файлах и специальных описателях (дескрипторах). На практике для распределения внешней памяти используются те же алгоритмы, что и для ОП. Отличие лишь в способе реализации (например, организация свободных участков в упорядоченный список явно не подходит из-за большого количества обращений к диску, поэтому указатели на элементы списка, как правило, объединяют в таблицу и загружают её в память, а затем отыскивают нужный указатель). Внешняя память разбивается на блоки фиксированного размера. Каждый блок имеет свой уникальный порядковый номер. **Блок** – наименьшая единица данных, которая участвует в обмене между устройствами ВП (HDD) и ОП. В итоге файл состоит из таких физических блоков.

### **Способы размещения блоков:**

**1. Непрерывное размещение.** Файл состоит из последовательности блоков диска, которые образуют единый, сплошной участок. **Достоинства:** адрес файла определяется номером начального блока, простота реализации. **Недостатки:** заранее неизвестна длина файла, неэффективное использование дискового пространства из-за большой дефрагментации.

### **2. Связный список блоков.**

В начале каждого блока содержится указатель (ссылка) на следующий блок. **Достоинства:** адрес файла задаётся номером первого блока, практически отсутствует фрагментация, нет проблем изменения файла. **Недостатки:** сложность реализации доступа к произвольной записи, затрата памяти на служебную информацию.

## **49. Физическая структура файла. Способы размещения информации. Связный список индексов. Достоинства и недостатки.**

Расположение информации на конкретном носителе чаще всего значительно отличается от логической упорядоченности. Преобразование логической в физическую структуру осуществляется на основе информации сосредоточенной в каталогах файлах и специальных описателях (дескрипторах). На практике для распределения внешней памяти используются те же алгоритмы, что и для ОП. Отличие лишь в способе реализации (например, организация свободных участков в упорядоченный список явно не подходит из-за большого количества обращений к диску, поэтому указатели на элементы списка, как правило, объединяют в таблицу и загружают её в память, а затем отыскивают нужный указатель). Внешняя память разбивается на блоки фиксированного размера. Каждый блок имеет свой уникальный порядковый номер. **Блок** – наименьшая единица данных, которая участвует в обмене между устройствами ВП (HDD) и ОП. В итоге файл состоит из таких физических блоков.

**Связный список индексов.** С каждым блоком связывается индекс. Индексы располагается в отдельной области диска. Например, в MS-DOS такая область называется таблицей размещения файлов FAT (File Allocation Table). Это позволяет отслеживать состояние различных участков дискового пространства. После выхода WINDOWS 95 широкое распространение получила 32-х битная VFAT (Virtual File Allocation Table) виртуальная таблица размещения файлов. В ней разрешены длинные имена файлов. Далее появилась FAT32 (в Windows 95 OSR 2), она отличается от VFAT только количественными параметрами (меньший размер кластеров, больший размер дисков, отсутствие ограничений на число файлов в корневом каталоге и др.). Для FAT32 нет отдельных драйверов. Здесь **сохраняются достоинства** предыдущего подхода, устраняется недостаток (то есть, чтобы осуществить доступ к произвольному месту не нужно просматривать все блоки, а достаточно прочитать блок индексов).

## **50. Физическая структура файла. Способы размещения информации. Перечень номеров блоков. Достоинства и недостатки.**

Расположение информации на конкретном носителе чаще всего значительно отличается от логической упорядоченности. Преобразование логической в физическую структуру осуществляется на основе информации сосредоточенной в каталогах файлах и специальных описателях (дескрипторах). На практике для распределения внешней памяти используются те же алгоритмы, что и для ОП. Отличие лишь в способе реализации (например, организация свободных участков в упорядоченный список явно не подходит из-за большого количества обращений к диску, поэтому указатели на элементы списка, как правило, объединяют в таблицу и загружают её в память, а затем отыскивают нужный указатель). Внешняя память разбивается на блоки фиксированного размера. Каждый блок имеет свой уникальный порядковый номер. **Блок** – наименьшая единица данных, которая участвует в обмене между устройствами ВП (HDD) и ОП. В итоге файл состоит из таких физических блоков.

**Перечень номеров блоков.** Номера блоков, занимаемых файлами, просто перечисляются. **Достоинства:** снижаются проблемы динамического размера файла, практически отсутствует фрагментация. **Недостатки:** усложняются алгоритмы поиска, увеличивается время доступа к информации.

Каждый файл в системе имеет дескриптор, в составе которого хранится список, содержащий 13 номеров блоков на диске. В этой схеме используется как прямая адресация, так и косвенная адресация. Первые 10 элементов списка непосредственно указывают на 10 блоков файла, если блоков не достаточно, то используют следующие 3 элемента списка. 11 элемент для одноуровневой адресации в нём указан номер блока, хранящий список из 128 номеров блоков, которые могут принадлежать файлу. Если требуется объём файла более чем  $10+128$  блоков, то переходят на следующий уровень. В итоге можно адресоваться к  $10+128+128^2+128^3$  блоков в составе первого файла.

## 51. Файловая система. Права доступа к файлу. Основные подходы к определению прав доступа.

Определение права доступа означает определение для пользователя дозволенные операции над файлами. В разных ОС определён свой список операций доступа. Можно выделить следующие операции: создание, уничтожение, открытие, закрытие, поиск, чтение, запись, получение атрибутов, установка атрибутов, чтение каталога и др.

Права могут быть заданы матрице прав доступа.

	Файл 1	Файл 2	...
Пользователь 1	Читать	Выполнять	...
Пользователь 2	Читать, писать	Создать, читать	...
...	...	...	...

Пользователи могут быть разделены на отдельные категории. При этом подход назначения прав может быть различными для каждой категории (например в UNIX все пользователи подразделяются на три категории: 1. Владелец файла 2. Член группы. 3. Все

остальные). Выделяют два подхода к определению прав доступа:

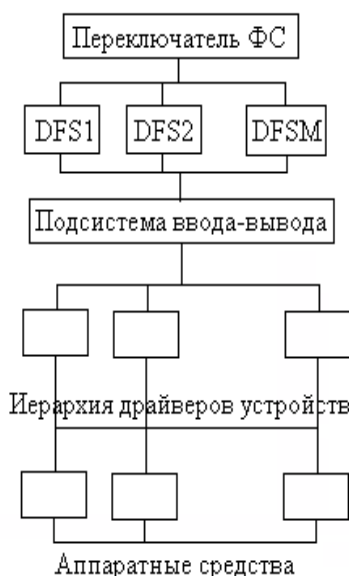
- Избирательный доступ (для каждого пользователя и каждого файла сам владелец может определять допустимые операции).
- Мандатный подход (система устанавливает права доступа по отношению к каждому разделу ресурсов в зависимости от того, к какой группе принадлежит пользователь).

## 52. Файловая система. Кэширование диска. Механизм кэширования диска. Как меняется размер кэша в Windows 95.

В некоторых файловых системах при работе с внешними устройствами используется подсистема буферизации, которая работает по принципу кэш-памяти. Запрос к внешнему устройству, в котором адресация осуществляется блоками, может быть перехвачена подсистемой буферизации. Такая система представляет собой буферный пул (однородных динамически распределяемых блоков ОП одинаковой длины). И комплекс программ управляющих этим пулом. Каждый буфер пула имеет размер равный одному блоку файла.

**Механизм работы:** При поступлении запроса на чтение некоторого блока подсистема буферизации сначала просматривает свой буферный пул, в случае обнаружения нужного блока, копирует его в буфер, запрашивающего процесса без обращения к внешнему устройству. Если нужный блок не обнаружен, то он считывается и одновременно передается процессу и записывается в буфер подсистемы. При отсутствии свободного буфера на диск вытесняется самая редкая используемая информация. В WINDOWS 95 размер кэша диктуется текущей ситуацией. При интенсивной работе сети ОС увеличивается размер кэша. И/или запуске большого числа приложений автоматически уменьшает размер.

## 53. Архитектура современной файловой системы. Многоуровневая фс.



Современные ОС дают возможность работать с несколькими ФС. Система имеет многоуровневую структуру. Приложение обращается к такой ФС только через переключатель (в Windows 95 – устанавливаемый диспетчер ФС – IFS (Installable File System manager)). Переключатель преобразует запросы в формат следующего уровня. Каждый DFS поддерживает определённую организацию файловой системы и позволяет сразу нескольким приложениям выполнять операции с файлами.

**Подсистема ввода-вывода** отвечает за загрузку, инициализацию и управление всеми модулями низших уровней файловой системы (драйверами портов). Данная подсистема постоянно присутствует в памяти и организует совместную работу иерархии драйверов устройств. Каждый уровень драйверов устройств, представляет определённый тип (драйвер HDD, драйвер, перехватывающий запросы к блочным устройствам, драйверы портов, управляющие конкретными адаптерами и т.д.)

EFS – представляет собой надстройку над NTFS, которая позволяет шифровать данные FAT. Для небольших дисков и простых структур каталогов максимальный объём до 2 Гб. По организации таблица размещается в начале тома. В целях защиты на томе хранится две копии таблицы. Таблица размещения файлов и каталог должны размещаться по строго фиксированным адресам. ОС использует таблицу для определения кластеров, которые занимают необходимые по существу таблицы похожа на оглавление книги.

## **54. Файловая система FAT32. Основные составляющие и характеристики. Организация доступа. Файловые системы NTFS и CDFS.**

ФС FAT16 используется для небольших дисков и простых структур каталогов (максимальный объем до 2 Гб). По организации таблица размещается в начале тома. В целях защиты на томе хранятся две копии таблицы. Таблица размещения файлов и каталог должны размещаться по строго фиксированным адресам. ОС использует таблицу для определения кластеров, которые занимает нужный файл. По существу таблица похожа на оглавление книги.

FAT 32 – это усовершенствованная версия FAT16, разработанная для WINDOWS. Предусматривает ряд специальных областей на диске, выделенных в процессе его форматирования: 1. Головная запись загрузки. 2. Таблица разбиения диска. 3. Запись загрузки. 4. Таблица размещения файлов. 5. Корневой каталог.

На физическом уровне пространство диска разбивают на области по 512 байт или более (их называют секторами). Блоки, из которых состоит файл, называют кластерами. Они состоят из целого числа (кратного степени 2) секторов. Размер кластера можно определить, поделив объем диска на 64 Кб и округлив результат до ближайшего целого числа кратного степени 2. Каждый кластер содержит номер следующего кластера занятого файла и так по цепочке. FAT представляет собой БД, связывающую кластеры дискового пространства с файлами. Каждый каталог также представляет собой БД.

Механизм доступа к БД каталога реализуется следующим образом: после того как пришел запрос на чтение, ОС просматривает запись каталога для него, с целью получить первый кластер этого файла, затем система обращается к элементу FAT для данного кластера, чтобы найти следующий кластер по цепочке и т.д., пока не будет найден последний кластер.

В FS FAT 32 номера элементов и номера секторов 32-х разрядные, то есть, максимальная емкость диска 2 Тб.

**Преимущества FAT 32 (по сравнению с FAT16):** более эффективное использование дискового пространства, более надежная и быстрая загрузка программ.

**NTFS** имеет более эффективные алгоритмы при работе с большими дисками, более высокую отказоустойчивость (есть возможность самовосстановления). Каждый файл имеет более богатый набор свойств. При разработке NTFS основной целью было обеспечение большой скорости выполнения стандартных функций над файлами. Позволяет назначать права доступа к отдельным файлам. Также используются кластеры в качестве базовой единицы дискового пространства. Размер кластера зависит от размера раздела.

Формирование раздела для использования NTFS приводит к созданию нескольких системных файлов и главной таблицы файлов MFT (Master File Table). MFT содержит информацию обо всех файлах и папках, имеющихся в разделе NTFS. NTFS – это объектно-ориентированная файловая система, которая обрабатывает все файлы, как объекты с атрибутами. Каждый занятый сектор в разделе NTFS принадлежит какому-нибудь файлу. Частью файла является информация с описанием самой FS.

**Отличия NTFS от предыдущих ФС:** 1. Система безопасности NT позволяет устанавливать различные права доступа к файлам и папкам для пользователей и групп пользователей. 2. Быстрое восстановление тома в случае сбоя. 3. Гибкие опции форматирования позволяют более эффективно использовать дисковое пространство. 4. Тома могут расширяться. 5. Возможность создания зеркальных томов (WINDOWS NT Server).

**Отличия NTFS 5 от NTFS 4:** 1. Защита отдельных файлов при помощи шифрования. 2. Расширение томов без перезагрузки. 3. Имеются возможности по отслеживанию распределенных ссылок, что позволяет сохранить ярлыки при перемещении файлов. 4. Использование квотирования диска. Можно ввести квоты на дисковое пространство, доступное для работы каждому пользователю (в предыдущих версиях любой пользователей имел все пространство диска). Квотирование выполняется по каждому тому.

В NTFS5 имеется оснастка MMC (Microsoft Management Console), работающая с томами FAT16, FAT32 и NTFS.

**Особенности дефрагментации в NTFS5:** при дефрагментации свободное пространство не объединяется в одну непрерывную область, а располагается в нескольких областях. Это сокращает время дефрагментации и практически не уменьшает производительность.

С помощью оснастки можно управлять и создавать следующие динамические тома:

1. Простой том (пространство на одном физическом диске).
2. Составной том (связанное пространство на нескольких дисках).
3. Чередующийся том (имеет несколько областей на разных дисках и при записи информация расщепляется и пишется параллельно на каждый из дисков тома).
4. Зеркальный том (это отказоустойчивая система, обеспечивает избыточность данных, создавая две копии одного тома).

5. RAID-5 (Redundant Array of Inexpensive Disk) нужен для избыточности информации (подсчета контрольных сумм на каждом диске и т.д.).

**CDFS** – это виртуальная ФС для CD-ROM аналогичная VFAT. Драйвер CDFS загружается динамически, если обнаружен привод CD-ROM. Поддерживает возможность мультисансової работы, при условии, если это поддерживает сам привод и драйверы. Использует стандарт ISO-9660. Исходная версия ISO не поддерживает права доступа и символьные ссылки. Имена файлов – до 32 символов, не различает прописные и строчные буквы. Ограничения на ISO были сняты в расширенном варианте стандарта RockRidge (с ним работают UNIX системы, но в Windows он не поддерживается). Для Windows и Linux была разработана альтернатива – Joliet, поддерживающая длинные имена файлов в формате Unicode.

## **55. Требования к ОС. Частотный принцип. Принцип модульности. Виды модулей по хар-ру использования.**

Независимо от назначения и не зависимости от их использования ОС, которые положены в основу их разработки:

**1-Частотный принцип.** Он основан на выделении действий в алгоритмах программ и данных в обработке массивов по частоте использования. Для действий, которые часто встречаются при работе ОС, обеспечивается условие их быстрого выполнения (такие программные тексты постоянно находятся в ОП и активно поддерживаются специальными средствами, как правило, часто операции стараются сделать более короткими). К данным, которые часто используются, обеспечивают более быстрый доступ. Частотный принцип наиболее важен в случае многоуровневого планирования.

**На долгосрочный уровень:**

- *Редкие и длинные операции управления при этом минимальным объектом управления является непосредственно программы без детализации особенностей их исполнения.*

**На краткосрочный уровень:**

- *Выносятся часто используемые и короткие операции отдельных программ.*

**2-Принцип модульности.** ОС должна состоять из законченных функциональных элементов (модулей), которые имеют средства сопряжения с подобными элементами или элементами более высокого уровня данной или другой ОС. Разделение системы на модули определяется использованием методов проектирования ОС (нисходящий, восходящий принцип). Например, модули могут быть отдельно транслируемыми программными единицами. Определённый уровень ОС может иметь свою систему модулей, образуя в результате обобщенный модуль. На более высоком уровне этот обобщенный модуль является одним из базовых модулей. Такое иерархическое упорядочивание модулей упрощает разработку и уменьшает число проектных ошибок.

**По характеру использования модули бывают:**

- *Однократными* (могут испортить сами себя и не восстанавливаться в исходное состояние).
- *Многократными* (они не портят себя и могут восстанавливаться).

Важное значение при построении ОС имеют модули, которые можно параллельно использовать, такие модули называются **реентерабельными**.

## **56. Требования к ОС. Принцип функциональной избирательности. Принцип генерируемости.**

**3-Принцип функциональной избирательности.**

Является логическим продолжением двух предыдущих принципов. В ОС выделяется некоторые части модулей, которые должны быть постоянно под рукой, чтобы обеспечить эффективную работу, называемым ядром. При формировании состава ядра, требуется выполнить два противоречивых требования:

1) В состав ядра должны войти наиболее часто используемые модули (функция). 2) Ядро не должно занимать много памяти. В ядро входят модули по управлению системой прерываний, средства по переводу программных состояния готовности и обратно, средства по распределению в ОП и CPU.

**4-Принцип генерируемости.** Должны быть разработаны специальные системные программы, которые при генерации ОС обеспечивают возможность настраивать ее, исходя из конкретной конфигурации машины и круга решаемых проблем.

## **57. Требования к ОС. Принцип функциональной избыточности. Принцип «по умолчанию».**

**5-Принцип функциональной избыточности.** Этот принцип учитывает возможность проведения одной и той же работы различными средствами (программными средствами). Например, ОС может обладать функциональной избыточностью при организации пакетной мультипрограммной обработки и допускать три альтернативных конфигурации:

- 1-Обеспечение мультипрограммирования с фиксированным числом задач.
- 2-С переменным числом задач.
- 3-На основе использования виртуальной памяти.

Реализация нескольких методов доступа к данным. Данный принцип обеспечивает гибкость и эффективность ОС.

**6-Принцип «по умолчанию».** Применяется при настройке ОС, как на стадии генерации, так и при эксплуатации. Принцип основан на хранении в системе базовых описаний структур процессов, модулей, конфигурации оборудования и данных, прогнозируемый объем ОП, времени счета и др. Эту информацию ОС использует в качестве заданной в случае, если пользователь сознательно или не сознательно не конкретизировал ее. В целом в результате этого приема сокращается число параметров, устанавливаемых пользователем.



## **58. Требования к ОС. Принципы перемещаемости и переносимости.**

### **7-Принцип перемещаемости и переносимости.**

Предусматривает построение модулей, исполнение которых не зависит от места расположения в ОП. ОС должна позволять динамически размещать командные сегменты и сегменты данных в различных областях памяти, не зависимо от процедуры компиляции. Действие перемещения, при котором с логическими адресами связываются конкретные физические адреса, могут осуществляться в разное время. Например, во время компоновки, когда отдельные модули связываются друг с другом.

2-Во время загрузки готового модуля в память.

3-Во время выполнения, когда реальные распределения адресов осуществляются аппаратно и др.

### **Принцип переносимости:**

Должен предусматриваться относительно легкий способ перемещения всей системы с различными типами процессоров или с разными аппаратными платформами. В некоторых случаях это может быть достаточно сложной задачей. Например, ОС с 32 битными адресами может быть легко перенесена на 16 битные адреса, а наоборот с большими трудностями.

## **59. Требования к ОС. Принцип совместимости. Принцип независимости программ от внешних устройств. Принцип открытой и наращиваемой системы. Принцип надежности и защиты.**

**8-Принцип совместимости.** Должны быть предусмотрены средства для выполнения прикладных программ, написанных для других ОС или для более ранних версий той же самой ОС. Выделяют два случая совместимости:

1-На уровне исполнения модуля можно запускать на различных ОС. Здесь необходимо иметь совместимость на уровне команд процессора на уровне системных вызовов и библиотечных, если они являются динамически связываемых.

2-На уровне исходных текстов приложений можно перекомпилировать исходные тексты в новый исполняемый модуль.

**9-Принцип независимости программ от внешних устройств.** Позволяет одинаково осуществлять операции управления внешними устройствами, не зависимо от конкретных физических характеристик.

**10-Принцип открытой и наращиваемой ОС.** Быть открытой означает быть доступной для анализа пользователя. Наращиваемой означает модифицируемой использующей не только использование генерации, но и добавлять новые модули и совершенствовать действующие, не нарушая целостности системы.

**11-Принцип надёжности и защиты.** ОС должна быть также надёжной, как и аппаратура, на которой она работает. ОС должна диагностировать ошибки, восстанавливаться после большинства сбоев или ошибок, защищать пользователей от их собственных ошибок.

## **60. Защита от несанкционированного доступа. Уровни несанкционированного доступа. Специальные средства защиты. Лицензирование средств защиты.**

Эффективной считается такая защита, стоимость взлома которой соизмерима с ценностью добываемой при этом информации. По степени сложности применяемых технических средств можно выделить **три уровня несанкционированного доступа:**

1. Низкий (вход в систему и получение в ней прав привилегированного пользователя).
2. Средний (прослушивание каналов передачи данных).
3. Высокий (сканирование излучения).

ОС предоставляют некоторую защиту от несанкционированного доступа, но для реализации более высокого уровня защиты необходимо использовать специальные средства шифрования и защиты. Такие средства нуждаются в обязательном государственном лицензировании на их создание, установку и эксплуатацию, а также реализацию и предоставление услуг в области криптографии. Государство ограничивает доступ на рынок средств безопасности зарубежных компаний, что обеспечивает широкое поле деятельности для российских разработчиков. Например, если компания не имеет лицензии ФАПСИ или ГТК, то разработка, производство, эксплуатация и/или реализация шифровальных средств, а также предоставление услуг в области криптографии для неё запрещены.

Вероятность несанкционированного входа в систему возрастает при её перегрузках. Например, при перегрузке ОС в результате массового подключения. Хакеры иногда создают сходные ситуации, направляя поток сообщений, который ОС не в состоянии корректно обработать, в результате создается открытый канал для доступа.

### **От администраторов требуются следующие действия по обеспечению безопасности:**

1. Своевременное обновление программного обеспечения.
2. Соблюдение всех правил конфигурирования программного обеспечения.
3. Проведение административных мероприятий.

Информацию можно перехватить, прослушивая канал связи. ФАПСИ выделяет 3 класса коммуникаций:

**1 Класс** – локальные сети, расположенные в «зоне безопасности» (территории с ограниченным доступом, экранированным оборудованием и линиями).

**2 Класс** – каналы вне «зоны безопасности», защищенные организационно-механическими мерами. Например, оптоволоконный кабель (в нем перехват затруднен).

**3 Класс** – незащищенные каналы связи общего пользования

## **61. Защита от несанкционированного доступа. Классы операционных систем по отношению к степени защиты.**

Программное обеспечение по степени защиты делится на следующие классы:

**Класс D** – защита отсутствует и пользователь имеет неограниченный доступ ко всем ресурсам (MS-DOS).

**Класс C** – наиболее популярный подкласс C2, общая характеристика:

Доступ с паролем и с именем. При работе с БД класса C пользователь, получив доступ к какой-нибудь таблице, получает доступ и ко всем имеющимся в ней данным. К этому классу относится большинство ОС. Например, UNIX, WINDOWS NT, 3.5, 2000 и др. Они сертифицированы C2.

**Рассмотрим основные требования данного класса:**

- 1-Владелец ресурсов должен иметь возможность контроля доступа к нему.
- 2-ОС должна защищать находящиеся в памяти и принадлежащей одному процессу данные от случайного использования их другими процессами. Например, WINDOWS NT Server защищает участок памяти так, что его содержимое не может быть прочитано даже после того, как процесс освободил его, а при удалении файла с диска запрещен доступ к его данным, даже если пространство удаленного файла выделено новому файлу.
- 3-Система должна иметь возможность уникальным образом идентифицировать каждого пользователя с целью отслеживания всей его деятельности.
- 4-Администраторы должны иметь возможность аудита всех событий связанных с защитой, но такими правами должен обладать ограниченный круг администраторов.
- 5-Система должна защищать себя от несанкционированной модификации работающей системы.

**Дополнительные уточняющие требования:**

- 1-Возможность контроля администратором, затем какие и кем используются ресурсы.
- 2-Возможность включение пользователей в группы и возможность установки времени работы и др.
- 3-Возможность аудита таких событий, как попытка регистрации, попытка получения доступа к файлам, принтерам и другим устройствам.
- 4-Блокировка учетных записей при неверной регистрации.
- 5-Установление срока жизни и правил использования паролей.

Необходимо отметить, что стандарты защиты при удаленном доступе еще не определены, реально это решает администратор. К этим вопросам относят: дозвон, сетевая маршрутизация, фильтрация сетевых потоков или сервисов, сетевая регистрация или аутентификация, передача файлов, электронная почта, связь с INTERNET и др.

**Ключевыми элементами подсистемы защиты WINDOWS NT являются:**

Распределитель локальной безопасности LSA (Local Security Authority).

**Обязанности:** 1-Предоставление пользователям доступа в систему.

2-Создание маркеров доступов в процессе регистрации.

3-Управление интерактивным процессом аутентификации пользователя.

4-Управление локальной политики защиты.

5-Контроль политики аудита.

6-Запись сообщений аудита, посылаемых справочным монитором защиты в журнал защиты.

Менеджер защиты учетных записей SAAM (Security Account Authority Manager). Он работает с БД защиты учетных записей и в сети может быть несколько таких БД.

Справочный монитор защиты SRM (Security Reference Monitor). Он нужен для усиления политики авторизации доступа и аудита и обеспечивает защиту объектов или ресурсов от несанкционированного доступа или модификации.

**Класс B** – БД этого класса позволяет использовать дифференцированный доступ к данным для различных пользователей даже внутри одной таблицы. Улучшенный с точки зрения безопасности стандартные ОС производят многие фирмы. Например: DEC, Hewlett-Packard, Santa Cruz Operation, Sun.

**Класс A** – наиболее защищенные ОС рекомендуется только при построении сетевой защиты от внешнего мира. Создают брандмауэры. Вероятность несанкционированного входа в систему возрастает при её перегрузках. Например, при перегрузке ОС в результате массового подключения к ней пользователей. Хакеры иногда создают сходные ситуации, направляя поток сообщений, который ОС не в состоянии корректно обработать, в результате создается открытый канал для доступа. **От администраторов при этом требуется следующее:**

- 1-Своевременное обновление программного обеспечения. Как правило, при выпуске новых версий становится общедоступная информация об ошибках предыдущей, в том числе и система защиты.
- 2-Соблюдение всех правил конфигурирования программного обеспечения.
- 3-Проведение административных мероприятий.

Защита данных от перехвата:

**ФАПСИ разделяет коммуникации на три канала:**

**1 Класс** – локальные сети, расположенные в «зоне безопасности». «Зона безопасности» (территории с ограниченным доступом и экранированным оборудованием и линиями).

**2 Класс** – каналы вне «зоны безопасности» защищенные организационно механическими мерами. Например, оптоволоконный кабель, в нем перехват затруднен.

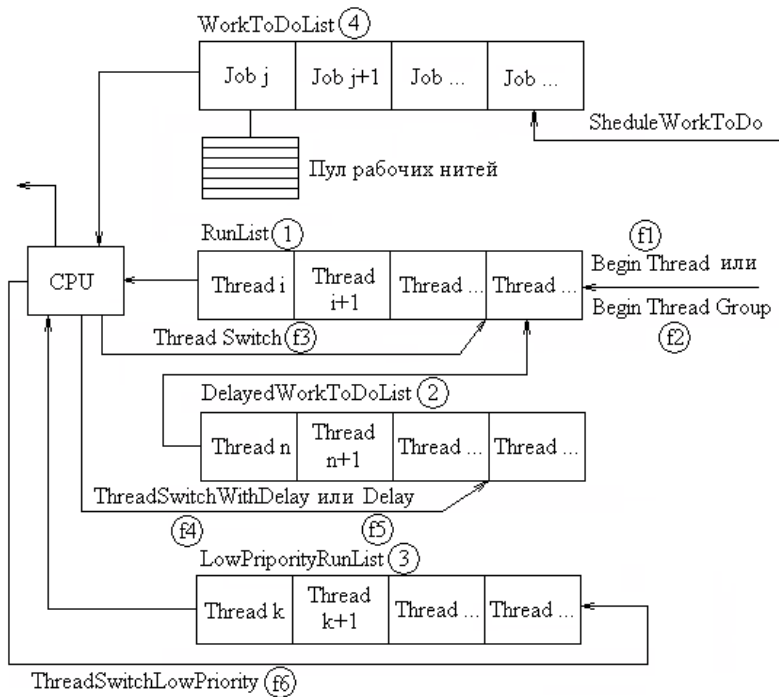
**3 Класс** – незащищенные каналы связи общего пользования.

**Брандмауэры** – каждый брандмауэр имеет собственную философию защиты. Общим является то, что они делят окружающий мир на два лагеря:

- *Внутренний* (доверенный пользователям).
- *Внешний* (всех остальных).



## 62. Система очередей планирования NetWare.



При создании нити с помощью функций f1 или f2 она попадает в конец очереди 1. Очередь 1 (**Run List**) содержит готовые к выполнению нити. Планировщик выбирает для выполнения стоящую первой нить и запускает ее на выполнение.

Нить, завершившая свою очередную итерацию, помещается в конец одной из очередей в зависимости от того, какой вызов передачи управления она использовала: в конец очереди 1, если вызвала f3; в конец очереди 2 при вызове f4 или f5; в конец очереди 3 при вызове f6. Если нить завершила свою работу, то она выполняет главную функцию return () и уничтожается.

Нити, находящиеся в очереди 3, запускаются на выполнение только в том случае, когда очередь 1 пуста. Обычно в эту очередь назначаются нити, которые выполняют несрочную фоновую работу. Очередь 4 является в системе самой приоритетной и необходима для выполнения очень срочных работ. Планировщик

разрешает выполниться подряд только определенному количеству нитей из очереди 4, а затем запускает нить из очереди 1. Очередь 4 и связанная с ней функция появились в версии 4.0 и значительно повышают производительность NLM (Novell Load Module) приложений. Исполняемые модули этой фирмы и имеют более высокий приоритет.

Рассмотренный механизм организации многопоточной работы сочетается со средствами синхронизации нитей.

## 63. Синхронизация и взаимодействие процессов. Эффект «гонок». Критическая секция. Взаимное исключение. Способы обеспечения взаимного исключения.

Взаимодействие между процессами может осуществляться с помощью обмена сообщениями или при помощи разделяемых совместно используемых переменных (участков памяти).

**Событие** – объект синхронизации, который используется для информирования потоков о том, что произошло некоторое событие. В ОС допускается одновременное возникновение нескольких событий.

Синхронизация подразумевает сигнализацию между процессами по определенному протоколу.

**Протокол** – набор правил и соглашений, не зависящий от времени и называемый событиями. В ОС допускается одновременное возникновение нескольких событий, каждому событию присваивают идентификатор, который называют флагом события. Этот флаг определяет возможность синхронизации. Количество флагов задается при проектировании. С каждым событием связано статическое слово и числовые значения. Если статус меньше 0, то это означает, что событие прекращено из-за ошибки и статическое слово содержит системный код ошибки. Если статус равен 0, то событие продолжается. Если статус больше 0, то это означает удачное завершение.

Обмен сообщениями между процессами имеет сложную структуру, поэтому такая связь между процессами создает новые проблемы, касающиеся адресации процессов структуры планировщиков процесса и др. Асинхронность выполнения процессов усложняет взаимодействие, так как предполагается, что один процесс выполняется со скоростью, не зависящей от других процессов. Принимающий сообщение процесс может быть не готов к приему, чтобы обеспечить взаимодействие используют временный буфер. Обмен между процессами может быть разделен на два класса:

1 класс – называется разделяемые (совместно используемые) переменные: события и семафоры.

**Семафор** – счётчик числа доступных ресурсов (число сообщений в очереди, притом предельное значение семафора соответствует максимуму этой очереди). Это неотрицательная целая переменная, для которой определены две операции: P и V.

**Операция P** – уменьшение семафора на 1 (если это возможно). Если P=0, то процесс, вызвавший операцию, ждет, пока уменьшение не станет возможным.

**Операция V** – семафор увеличивается на 1.

2 класс – сообщения.

Используется в распределённых системах, когда прочесу имеют собственную ОП и использовать распределённую память невозможно.

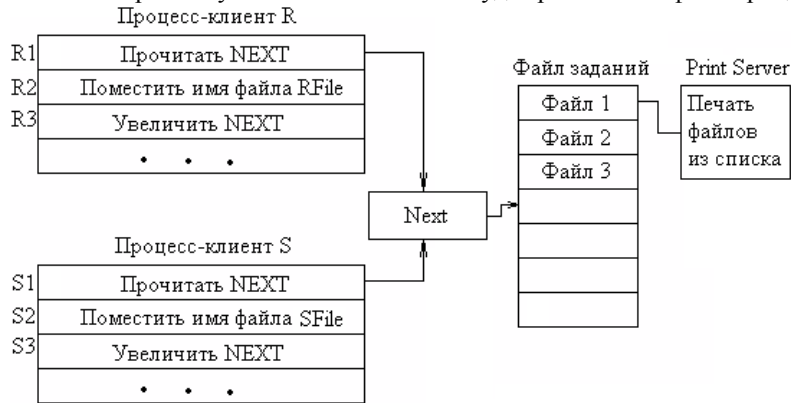
Обмен информации между процессами имеет 2 ограничения со стороны ресурсов:

1. Объем сообщений не должен превышать емкости отведенного буфера.
2. Принимающий ресурс не может обрабатывать сообщение быстрее, чем они создаются передающим процессом.

Необходимость в синхронизации возникает в программе печати файлов (Print Server). Эта программа печатает по очереди все файлы, имена которых другие программы в порядке поступления записывают в специальный

общедоступный файл заказов. Особая переменная NEXT также доступна всем процессам-клиентам и содержит номер первой свободной для записи позиции файла заказов. Процессы-клиенты читают эту переменную, записывают в свободную позицию имя своего файла и увеличивают значение NEXT на 1.

**Эффект «гонок».** Пусть процесс R решил распечатать свой файл и прочитал NEXT=4, но поместить имя файла не успел, так как закончился квант времени. Очередной процесс S, желающий распечатать файл, прочитал то же самое значение переменной NEXT и поместил в 4 позицию имя своего файла и нарастил значение переменной на 1. После передачи управления процессу R он размещает в позицию 4 (это значение он прочитал раньше) свой файл и увеличивает переменную NEXT. В итоге не будет распечатан файл процесса S.



**Критическая секция** – это часть программы, в которой осуществляется доступ к разделенным данным.

**Взаимное исключение** – это исключение эффекта «гонок», обеспечивающее, чтобы в каждый момент в критической секции находилось не более одного процесса. Простейший способ его обеспечения – позволить процессу, находящемуся в критической секции, запрещать все прерывания. **Недостаток** – возможность крайне длительно долгой загрузки системы процессом. Другой

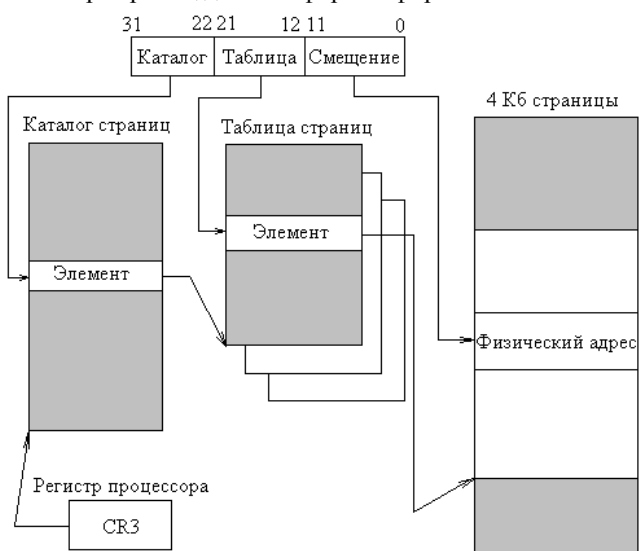
способ заключается в использовании блокирующих переменных. С каждым разделенным ресурсом связывается двоичная переменная (1 – ресурс свободен, 0 – ресурс занят). Однако операция проверки и установки блокирующей переменной должна быть неделимой, т.е. необходимо запретить прерывание на протяжении всей этой операции. Недостаток – бесполезно тратится процессорное время при опросе блокирующей переменной.

## 64. Организация виртуальной памяти в Windows NT. Схема преобразования адреса для платформы Intel. Элемент PTE.

Windows NT построена в соответствии с классическими принципами. ВП имеет страничную организацию.

В общем виде схема такова: линейный адрес разбивается на несколько частей (старшая часть адреса содержит номер элемента в корневой таблице. Этот элемент содержит адрес в таблице следующего уровня. Следующая часть линейного адреса содержит номер элемента уже в этой таблице и т.д. до последней страницы, которая содержит номер физической страницы. Самая младшая часть адреса является номером байта в этой странице).

Размер страниц для платформы фирмы Intel составляет 4 Кб, а для платформ DEC Alpha – 8 Кб.



А схема страничного преобразования выглядит так: линейный 32-х разрядный адрес занимает три части:

1. Старшие 10 разрядов адреса определяют номер одного из 1024 элементов в каталоге страниц, адрес которого находится в регистре процессора CR3. Этот элемент содержит физический адрес таблицы страниц.
2. Следующие 10 разрядов линейного адреса определяют номер элемента таблицы. Элемент в свою очередь содержит физический адрес страницы ВП.
3. Младших 12 разрядов линейного адреса достаточно, чтобы определить внутри 4КБ страницы точный физический номер, адресуемой ячейки.

Рассмотрим отдельный элемент таблицы страницы, которой называют PTE (Page Table Element).

Старшие 5 бит определяют тип страницы с точки зрения допустимых операций. WIN 32 API поддерживает 3 допустимых значения этого поля: PAGE\_NOACCESS, PAGE\_READONLY, PAGE\_READWRITE. Следующие 20 бит определяют базовый адрес страницы. Следующие 4

бита описывают используемый файл подкачки. Комбинация этих битов определяют один из 16 возможных в системе файлов подкачки. Последующие 3 бита определяют состояние страницы в системе. Старший из них называют T (Transition). Он определяет следующую страницу, как переходную. Следующий – бит D (Dirty) отмечает страницу, в которой была произведена запись. Информация об изменениях в страницах необходима для того, чтобы принимать решение о состоянии страницы в файле по указанию при её вытеснении. То есть если страница не изменялась в памяти после загрузки, то её можно просто стереть, так как копия осталась в файле подкачки. Последний – P (Present) определяет, присутствует страница в ОП или в файле подкачки. Для ускорения страничного преобразования в процессоре имеется специальная кэш-память, которая называется TLB (Translation Look-a-side Buffer). Каждый процесс в Windows NT имеет свой отдельный каталог страниц и свое собственное независимое адресное пространство, что дает дополнительные возможности при защите процессов.

## 65. Структура файла в ОС Unix. Структура дескриптора файла.

Усложняются алгоритмы распределения, поиска, то есть, увеличивается время доступа к информации. Пример в (UNIX) реализован вариант, который позволяет обеспечить фиксированную длину адреса, независимо от размера файла.

Каждый файл в системе имеет дескриптор, в составе которого хранится список, содержащий 13 номеров блоков на диске. В этой схеме используется как прямая адресация, так и косвенная адресация. Первые 10 элементов списка непосредственно указывают на 10 блоков файла, если блоков не достаточно, то используют следующие 3 элемента списка. 11 элемент для одноуровневой адресации в нём указан номер блока, хранящий список из 128 номеров блоков, которые могут принадлежать файлу. Если требуется объём файла более чем  $10+128$  блоков, то переходят на следующий уровень. В итоге можно адресоваться к  $10+128+128^2+128^3$  блоков в составе первого файла.

## 66. Сравнительная характеристика FAT16, FAT32, NTFS4, NTFS5.

ФС FAT16 используется для небольших дисков и простых структур каталогов (максимальный объём до 2 Гб). По организации таблица размещается в начале тома. В целях защиты на томе хранится две копии таблицы. Таблица размещения файлов и каталог должны размещаться по строго фиксированным адресам. ОС использует таблицу для определения кластеров, которые занимает нужный файл. По существу таблица похожа на оглавление книги.

FAT 32 – это усовершенствованная версия FAT16, разработанная для WINDOWS. Предусматривает ряд специальных областей на диске, выделенных в процессе его форматирования: 1. Головная запись загрузки. 2. Таблица разбиения диска. 3. Запись загрузки. 4. Таблица размещения файлов. 5. Корневой каталог.

Блоки, из которых состоит файл, называют кластерами. Размер кластера в FAT32 меньше, что позволяет более экономно использовать пространство диска. В FS FAT 32 номера элементов и номера секторов 32-х разрядные, то есть, максимальная ёмкость диска 2 Тб. **Преимущества FAT 32 (по сравнению с FAT16):** более эффективное использование дискового пространства, более надёжная и быстрая загрузка программ.

NTFS имеет более эффективные алгоритмы при работе с большими дисками, более высокую отказоустойчивость (есть возможность самовосстановления). Каждый файл имеет более богатый набор свойств. При разработке NTFS основной целью было обеспечение большой скорости выполнения стандартных функций над файлами. Позволяет назначать права доступа к отдельным файлам. Также используются кластеры в качестве базовой единицы дискового пространства. Размер кластера зависит от размера раздела, но он меньше, чем для ОС FAT32 (для одинаковых объёмов раздела).

Формирование раздела для использования NTFS приводит к созданию нескольких системных файлов и главной таблицы файлов MFT (Master File Table). MFT содержит информацию обо всех файлах и папках, имеющихся в разделе NTFS. NTFS – это объектно-ориентированная файловая система, которая обрабатывает все файлы, как объекты с атрибутами. Каждый занятый сектор в разделе NTFS принадлежит какому-нибудь файлу. Частью файла является информация с описанием самой FS.

**Отличия NTFS от предыдущих ФС:** 1. Система безопасности NT позволяет устанавливать различные права доступа к файлам и папкам для пользователей и групп пользователей. 2. Быстрое восстановление тома в случае сбоя. 3. Гибкие опции форматирования позволяют более эффективно использовать дисковое пространство. 4. Тома могут расширяться. 5. Возможность создания зеркальных томов (WINDOWS NT Server).

**Отличия NTFS 5 от NTFS 4:** 1. Защита отдельных файлов при помощи шифрования. 2. Расширение томов без перезагрузки. 3. Имеются возможности по отслеживанию распределённых ссылок, что позволяет сохранить ярлыки при перемещении файлов. 4. Использование квотирования диска. Можно ввести квоты на дисковое пространство, доступное для работы каждому пользователю (в предыдущих версиях любой пользователей имел все пространство диска). Квотирование выполняется по каждому тому.

## 67. Файловая система HPFS. Основные характеристики.

Разработана в 1989 г. совместно с корпорациями IBM и Microsoft. Используется в OS/2. Многие идеи в HPFS получили развитие в NTFS. В HPFS размер кластера всегда равен размеру сектора, не зависимо от физического размера раздела диска. Первые 16 секторов раздела соответствуют загрузочный блок, содержащий метку диска и программу начальной загрузки. В секторе 16 располагается суперблок. Он содержит общую информацию о ФС: размер раздела, указатель на корневой каталог, счетчик элементов каталогов, номер версии HPFS, указатель на список испорченных блоков на диске, таблицу дефектных секторов и список доступных секторов.



Сектор 17 называется запасной блок (Spare Block) и содержит указатель на список секторов, которые можно использовать, счетчик доступных секторов для «горячего» исправления ошибок, указатель на резерв свободных блоков (они нужны для управления деревьями каталогов), информацию о языковых наборах символов (она нужна для пересылки файлов, составленные на различных языках). Оставшееся пространство делится на полосы, размером 8 Мб.

Каждая полоса содержит таблицу, в которой хранится информация, о занятых и свободных кластерах. Размер таблицы составляет 2Кб. Чтобы максимально увеличить протяженность непрерывного пространства размещения файлов таблицы располагаются в начале и в конце полос.

Особенность HPFS в физическом расположении каталогов на диске. При форматировании раздела для каталогов заранее резервируется необходимое пространство в полосе, расположенной на середине диска. Это позволяет добиться, чтобы магнитные головки никогда не проходили более половины ширины диска (уменьшение времени доступа). Если системе HPFS требуется больше пространства, то она может его выделить в любой свободной области диска. Каждый элемент в каталоге HPFS описывается специальной структурой (файловым дескриптором), занимающей 1 сектор и содержащей указатель на начало файла, первые 15 символов имени файла (максимально до 255), время последней записи и последнего доступа, журнал с информацией о предыдущих обращениях к файлу, структуру распределения секторов, первые 300 байт расширенных атрибутов файла (до 64 Кб).

Файловые дескрипторы хранятся в смежных с представляемыми ими файлами секторах. Когда файл открывается, в кэш автоматически считывается 4 сектора (файловый дескриптор и 3 первых секторов файла).

В HPFS файл делится на фрагменты, состоящие из несколько подряд идущих секторов. Каждый фрагмент определяется указателем на первый сектор и количеством секторов. Файловый дескриптор может хранить до 8 таких пар сектор-длина. Если файлу требуется больше пространства, то HPFS изменяет структуру таким образом, что файловый дескриптор становится корнем сбалансированного дерева секторов размещения.

Каталог в HPFS – это файл переменного размера, содержащий имя файла, длину имени, время создания, указатель на файловый дескриптор и т.д. Первое поле в каждой записи в каталоге содержит его длину, последнее поле – пустое и служит для указания конца каталога. Все записи в каталоге упорядочены по имени файла. Пространство под каталог выделяется блоками из 4 секторов. Если весь каталог не помещается в такой блок, то выделяется новый блок из 4 секторов (как и в NTFS).

## 68. Файловая система Exr2fs.

Используется в ОС Linux. Minix – первая ФС этого класса. Minix имела низкую производительность, объем не более 64 Мб, имя файла не более 14 символов, была эффективна на гибких дисках НМД.

После форматирования диск разбивается на сектора:

<table><tr><td>Boot сектор</td><td>Группа блоков 0</td><td>Группа блоков 1</td><td>...</td><td>Группа блоков N</td></tr></table>						Boot сектор	Группа блоков 0	Группа блоков 1	...	Группа блоков N
Boot сектор	Группа блоков 0	Группа блоков 1	...	Группа блоков N						
Суперблок	Файл дескриптора	Битовая карта блоков	Битовая карта индексных дескрипторов	Таблица индексных дескрипторов	Данные					

Суперблок описывает информацию самой ФС (размер блока, число блоков в группе, количество свободных индексных дескрипторов, количество свободных блоков и т.д.). Он повышает надёжность ОС. Файловый дескриптор содержит таблицу дескрипторов групп (каждый её элемент описывает свою группу блоков и содержит указатели на блоки (битовая карта свободных блоков и битовая карта индексных дескрипторов)), таблицу индексных дескрипторов и свободных блоков в группе.

Битовая карта используется для выделения блоков файла. Битовая карта индексных дескрипторов позволяет отслеживать занятые и свободные индексные дескрипторы. Файлы представляются индексными дескрипторами, где каждый дескриптор содержит тип файла, права доступа, владельца, ячейки времени изменения, размер и указатели на блоки данных. Каталог в данной системе – файл, содержащий массив специальных структур. В каждой структуре имеется 3 поля: индексный дескриптор, длина имени файла и имя файла. Данная система работает на разделах до 2 Тб. Размер блока устанавливается при создании ФС и может быть 1,2,4 Кб.

Особенности:

1. Можно установить атрибуты на каталог, и новые файлы этого каталога наследуют их.
2. Есть возможность создавать быстрые символические ссылки, не содержащие блоков на диске. Вся информация хранится в индексном дескрипторе (объём её ограничен).
3. Поддерживает новые объекты ядра, которые используются для межсетевого взаимодействия.
4. в данной ФС используется упреждающее выделение блоков файла (до 8 ближайших блоков).

## 69. Свойства распределённых FS.

**Основные вопросы, решаемые распределёнными ФС:**

1. Вопрос обеспечения пространства имен.
  - Каждый клиент использует один и тот же путь для доступа к определенному файлу.
  - У каждого клиента свое пространство имен, реализуется это путем монтирования разделяемых поддеревьев произвольного каталога в иерархии файлов.
2. Вопрос определения вектора состояний.
  - Сервер обеспечивает хранение информации об операциях клиента между запросами. Такая информация используется для корректного выполнения следующих запросов, например, запоминать, какие файлы клиент открыл, а также смещение внутри файла и другую информацию. **Достоинство:** серверы работают быстрее. **Недостаток:** нужны дополнительные ресурсы.
  - Сервер без сохранения состояний. Он более простой в разработке и реализации, но имеет меньшую производительность.
3. Семантика разделения файлов – несколько клиентов обращаются к 1му файлу. Требование ОС: изменение, сделанное одним пользователем должно быть видно другим.

Подходы:

1. Сессионная семантика – подход «открыть-закрыть», то есть только после закрытия файла одним процессом,

другие процессы могут видеть изменения. Особенность: последний закрытый вариант считается окончательным.

2. Прохождение определённого интервала времени, после которого изменения попадут к другим пользователям.

3. Результаты каждой операции **немедленно** становятся видны другим (UNIX).

4. Транзакции. Используется принцип «всё или ничего». Один процесс открывает файлы-объекты и начинает транзакцию с другими процессами, то есть, они тоже могут выполнять операции. Инициатор может объявить, что он завершает работу и если с ним все согласны, то результат фиксируется.

5. Неизменяемые файлы. Всегда доступны 2 операции: создать и прочитать. Файл, например, модифицируется, а затем старый заменяется целиком.

4. Вопросы, связанные с методами удалённого доступа. Один из вариантов – DFS (централизованный доступ к ресурсам на основании единого иерархического пространства имён)

**Свойства распределённых ФС.** В основе распределений ФС лежит модель клиент-сервер. В данном случае под клиентом подразумевается машина, которая обращается к некоторому файлу, а под сервером понимается машина, хранящая файлы и обеспечивающая доступ к ним. Распределённые ФС имеют ряд важных свойств:

1. Сетевая прозрачность. Клиенты должны иметь возможность обращаться к удалённым файлам, пользуясь теми же самыми операциями, что и для доступа к локальным данным.

2. Прозрачность размещения. Имя файла не должно определять его местоположение в сети.

3. Независимость размещения. Имя файла не должно меняться при изменении его физического местоположения.

4. Мобильность пользователя. Пользователи должны иметь возможность обращаться к разделяемым файлам из любого узла сети.

5. Устойчивость к сбоям. Система должна продолжать функционировать при неисправности отдельного компонента. Однако это может приводить к уменьшению производительности или к исключению доступа к некоторой части ФС.

6. Масштабируемость. Система должна обладать возможностью масштабирования в случае увеличения нагрузки. Кроме этого должна существовать возможность постепенного наращивания системы, путем добавления отдельных компонентов.

7. Мобильность файлов. Должна быть обеспечена возможность перемещения файлов из одного места в другое в пределах системы.

## 70. Микроядерные ОС. Примеры. Проблемы проектирования. Микроядерные ОС. Принципы организации, функции и особенности микроядра.

**Микроядро** – это минимальная стержневая часть ОС, служащая основой модульных и переносимых расширений.

**Проблемы проектирования микроядерных ОС (не решены вопросы):**

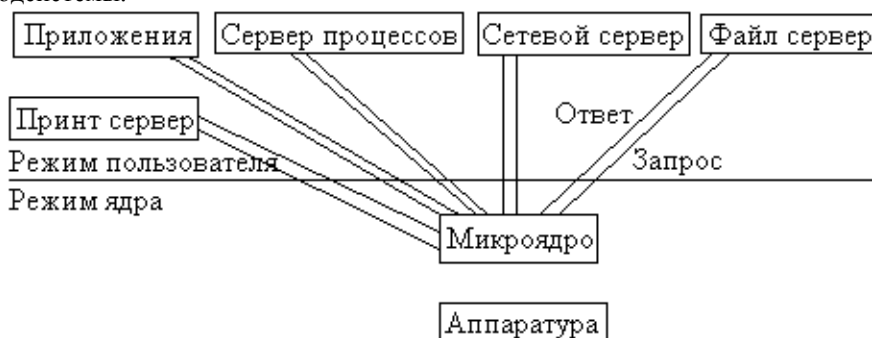
1. Как следует организовать различные службы ОС по отношению к микроядру.

2. Как проектировать драйверы устройств большой эффективности и как сохранить максимальную независимость от аппаратуры.

3. Где следует выполнять операции, напрямую не относящиеся к ядру.

4. Стоит ли сохранять программы, имеющихся подсистем или лучше отбросить всё и начать с нуля.

**Функции микроядра.** Микроядро реализует базовые функции, на которые опираются другие системные службы и приложения. Такие важные компоненты, как FS, система управления окнами и службы безопасности становятся внешними модулями. Они взаимодействуют с ядром и друг с другом. WINDOWS NT отчасти можно считать микроядерной, так как используется присущая микроядерному подходу модульность для создания общей структуры WINDOWS NT. А каждая ОС (MS-DOS, WINDOWS 3.1, OS/2 и др.) эмулируется в виде отдельного модуля или подсистемы.



В микроядерной архитектуре вертикальное распределение функций заменяется на горизонтальное, то есть компоненты взаимодействуют и используют средства микроядра для обмена сообщениями. Микроядро проверяет законность сообщений, пересылает их между компонентами и обеспечивает доступ к аппаратуре. Такой подход позволяет использовать микроядерный ОС в

распределённых средах. Микроядру безразлично, поступило ли сообщение от локального или удалённого процессора. Однако пересылка сообщений производится медленней обычных вызовов функций. Одна из задач – задача оптимизации пересылок сообщений.

**Обычно в микроядре включают:**

1. Машиннозависимые программы, включая поддержку мультипроцессорной работы.

2. Некоторые функции управления процессами.

3. Обработку прерываний.

4. Поддержку пересылок сообщений.

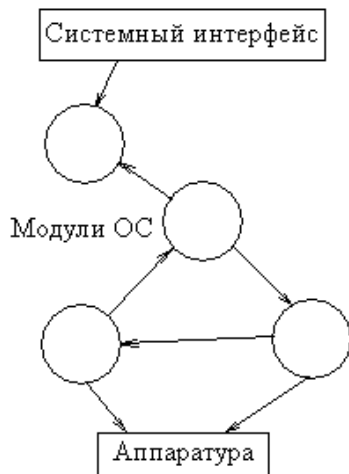
Часто в микроядро включается функция планирования процессов, но может быть вариант, когда планировщик

размещен вне ядра, а микроядро используется только для непосредственного управления процессом.

**Переносимость** – в микроядре изолированная вся машинно-зависимая часть ОС, поэтому перенос системы на новую машину требует меньших изменений, так как они логически сгруппированы.

**Расширяемость** – такая ОС позволяет добавлять новые функции на основе ограниченного набора интерфейса микроядра. Микроядро обладает небольшим набором API, например, QNS имеет 14 системных вызовов. OSF (Open Software Foundation) – около 200 системных вызовов. Такое маленькое количество системных вызовов увеличивает шансы получить качественные программы. Основным принципом организации микроядерной ОС является включение в состав ядра только тех функций, которые необходимо выполнять в режиме супервизора и защиты памяти.

## 71. Структуры монолитной, структурированной, микроядерной ОС и их особенности.

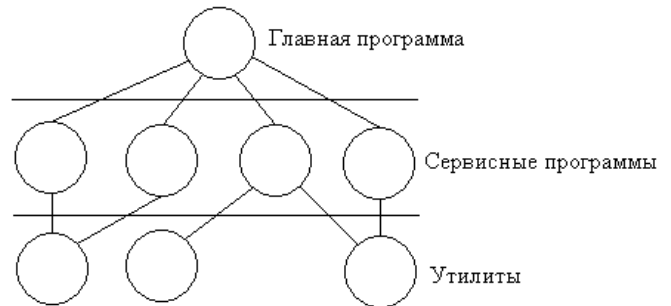


### 1. Монолитные ОС.

ОС в виде набора процедур, где каждая процедура может вызвать любую другую. Отсутствует правило вертикального управления. Для построения монолитной системы необходимо скомпилировать все отдельные процедуры и связать их в единый объектный файл.

### 2. Структурированная, монолитная ОС.

Монолитная ОС может быть структурирована. При обращении к системе, вызываемые параметры помещаются в строго отведенные места (в регистр или стек), а затем выполняются специальная команда прерывания (вызов



ядра). Эта команда переключает машину из пользовательского режима в режим ядра. Затем проверяются параметры вызова, индексируется таблица ссылок на процедуры и вызывается соответствующая процедура. В этой модели для каждого системного вызова имеется одна сервисная процедура. Главная программа вызывает требуемые сервисные процедуры, реализующие системные вызовы, а утилиты обслуживают сервисные процедуры (одна может обслуживать несколько сервисных процедур).

### 3. Многоуровневые иерархические системы.

Является обобщением предыдущего подхода. В 1968 г. Предложена простую пакетная ОС с шестью уровнями:

0 - отвечает за распределение времени. Процедура, переключающая процедуры по прерыванию или окончанию кванта времени.

1 - выполняет функции виртуальной ОП.

2 - связь между консолью оператора и процессами.

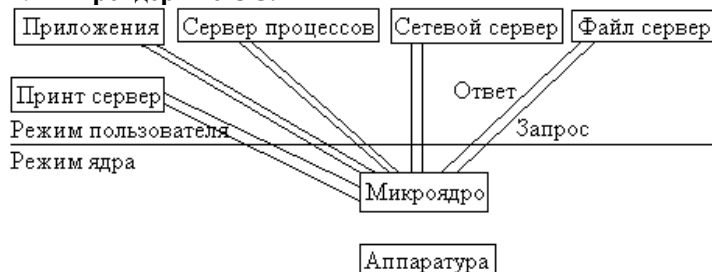
3 - управление устройствами ввода/вывода и буферизация потоков.

4 - пользовательские программы.

5 - процесс системного оператора.

Каждый из уровней может взаимодействовать только с непосредственно примыкающим к нему уровнем. Вся ОС связана иерархией уровней, а это затруднило развитие и модификацию системы.

### 4. Микроядерные ОС.



В микроядерной архитектуре вертикальное распределение функций заменяется на горизонтальное, то есть компоненты взаимодействуют и используют средства микроядра для обмена сообщениями. Микроядро проверяет законность сообщений, пересылает их между компонентами и обеспечивает доступ к аппаратуре. Такой подход позволяет использовать микроядерный ОС в распределенных средах. Микроядру безразлично, поступило ли сообщение

от локального или удаленного процессора. Однако пересылка сообщений производится медленней обычных вызовов функций. Одна из задач – задача оптимизации пересылок сообщений.

**Обычно в микроядре включают:** 1. Машиннозависимые программы, включая поддержку мультипроцессорной работы. 2. Некоторые функции управления процессами. 3. Обработку прерываний. 4. Поддержку пересылок сообщений.

Часто в микроядро включается функция планирования процессов, но может быть вариант, когда планировщик размещен вне ядра, а микроядро используется только для непосредственного управления процессом.



## 72. ОС карманных компьютеров.

### Особенности в сравнении с ОС общего назначения:

1. Графический интерфейс пользователя относительно небольшой и упрощенный из-за небольшого размера экрана.
2. У ПК загрузка десятки секунд, а у КПК - несколько секунд, так как пользователь включает его намного чаще.
3. Должна быть система взаимодействия карманной ОС с персональной.
4. Отсутствие традиционной клавиатуры и мыши, поэтому подсистема ввода-вывода отличается.
5. Контроль источника питания на уровне ОС.
6. Большая энергоемкость памяти, накладывает ограничения на подсистему управления памятью.
7. FS сильно отличается от классических FS.
8. Аппаратная поддержка преобразования виртуальных адресов в физические процессором не предоставляются, но есть поддержка ускоренного вычисления физического адреса по специальным алгоритмам.
9. По переключению потоков необходимо быстро сохранять приостановленный процесс и восстанавливать **контекст** процесса, который становится активным.
10. Поддержка прерываний обеспечивается контролером прерываний. Например, платформа Palm поддерживает 18 уровней прерываний и 7 уровней приоритетов. Приоритеты прерываний определяются уровнем приоритета устройства, вызвавшего его.

### Примеры:

1. Palm OS. Это микроядерная ОСРВ. Имеет многозадачное ядро. Функции ядра: планировщик процессов, поддержка прерываний, синхронизация процессов, контроль над использованием ресурсов. Использует вытесняющий и невытесняющий (по умолчанию) алгоритмы планирования. Для синхронизации использует события, сообщения, буферы.
2. WINDOWS CE – является модульной и в отличие от Palm на ее базе можно создавать системы под конкретные цели и задачи. Поддерживает стандартные коммуникационные протоколы Ethernet и большое число мониторов, различные типы процессоров. Это полноценная 32-х-разрядная объектно-ориентированная, многозадачная ОС с фиксированной очередью приоритетов. Применяется вытесняющий алгоритм. Имеет возможности ОСРВ, 500 наиболее распространенных функций Win32API.