# *Операционные системы*

1. Логическая организация файла. Файлы с последовательной и библиотечной структурой.
2. Типы и свойства операционных систем.
3. Архитектура современной файловой системы. Многоуровневая файловая система.
4. Организация виртуальной оперативной памяти. Схема структурирования переменными страницами.
5. Организация виртуальной оперативной памяти. Схема сегментно-страничной структуризации.
6. Задачи управления виртуальной памятью.
7. События. Семафоры. Сообщения. Их основное назначение.
8. Структура файла в ОС Unix. Структура дескриптора файла.
9. Система распределения оперативной памяти. Алгоритм оптимального размещения.
10. Синхронизация и взаимодействие процессов. Эффект «гонок». Критическая секция. Взаимное исключение. Способы обеспечения взаимного исключения.
11. Эволюция операционных систем.
12. Процесс. Основное понятие. Дескриптор процесса. Виды групп информации дескриптора.
13. Физическая структура файла. Способы размещения информации: непрерывное размещение и перечень номеров блоков. Достоинства и недостатки.
14. Файловая система. Кэширование диска. Механизм кэширования диска.
15. Логическая организация файла. Файлы с индексно-последовательной структурой.
16. Многоочередные дисциплины обслуживания процессов. Простая и приоритетная дисциплины.
17. Вытесняющие алгоритмы планирования процессов.
18. Организация виртуальной оперативной памяти. Схема структурирования фиксированными страницами. Механизм работы.
19. Структура операционной системы на примере Windows NT. Назначение основных модулей.
20. Система распределения оперативной памяти. Цели распределения. Основные решаемые задачи. Распределение памяти в двухуровневой ОС.
21. Алгоритм циклического планирования процессов.
22. Организация процессов в операционной системе UNIX.
23. Алгоритм приоритетного планирования процессов. Статическое и динамическое приоритетное планирование.
24. Управление оперативной памятью. Свопинг.
25. Файловая система. Права доступа к файлу. Основные подходы к определению прав доступа.
26. Синхронизация процессов. Мьютексы.
27. Ресурсы. Классификация ресурсов. Категории ресурсов.
28. Логическая организация файловой системы.
29. Управление оперативной памятью. Схема механизма физической адресации.
30. Организация виртуальной памяти в Windows NT. Схема преобразования адреса для платформы Intel. Элемент PTE.
31. Основные функции операционных систем.
32. Структура операционной системы на примере Windows NT. Назначение основных модулей.
33. Характеристики файлов. Типы доступа к файлу.
34. Граф существования процесса. Основные состояния процесса. Условия перехода из одного состояния в другое.
35. Физическая структура файла. Способы размещения информации. Связный список индексов. Достоинства и недостатки.
36. Динамические тома в Windows NTFS5.
37. Виды памяти. Основные функции управления оперативной памятью.
38. Сравнительная характеристика FAT16, FAT32, NTFS4,NTFS5.

39. Управление оперативной памятью. Использование оверлеев.
40. Организация виртуальной оперативной памяти. Схема сегментной структуризации.
41. Система очередей планирования NetWare.
42. Процессы в Windows NT. Процесс как объект на высоком уровне абстракции. Атрибуты и сервисы процесса-объекта.
43. Основные понятия: Операционная система. Процесс. Поток. Многозадачность. Многопоточность.
44. Структура операционной системы на примере MS-DOS. Назначение основных модулей.
45. Файловая система. Задача файловой системы. Функции файловой системы.
46. Классические дисциплины обслуживания очереди на исполнение процесса.
47.Система распределения оперативной памяти. Алгоритмы, основанные на выделении непрерывной единственной зоны.
48. Алгоритм адаптивно-рефлективного планирования процессов.
49. Планирование процессов. Планировщик. Двухуровневая система управления процессами. Типы планировщиков.
50. Структуры монолитной, структурированной, микроядерной ОС и их особенности.
51. Схема взаимодействия файловой системы. Уровни файловой системы.
52. Планирование процессов. Алгоритм лотерейного планирования.

# *1. Логическая организация файла. Файлы с последовательной и библиотечной структурой.*

Файл - поименованная целостная совокупность данных на внешнем носителе.

Файловая система - часть ОС, которая обеспечивает выполнение операций над файлами.

Программист имеет дело с логической организацией файла, представляя файл в виде определенным образом организованных логических записей.

Логическая запись - это наименьший элемент данных, которым может оперировать программист при обмене с внешним устройством. Даже если физический обмен с устройством осуществляется большими единицами, операционная система обеспечивает программисту доступ к отдельной логической записи.

На рисунке показаны несколько схем логической организации файла. Записи могут быть фиксированной длины или переменной длины. Записи могут быть расположены в файле последовательно (последовательная организация) или в более сложном порядке, с использованием так называемых индексных таблиц, позволяющих обеспечить быстрый доступ к отдельной логической записи (индексно-последовательная организация). Для идентификации записи может быть использовано специальное поле записи, называемое ключом.

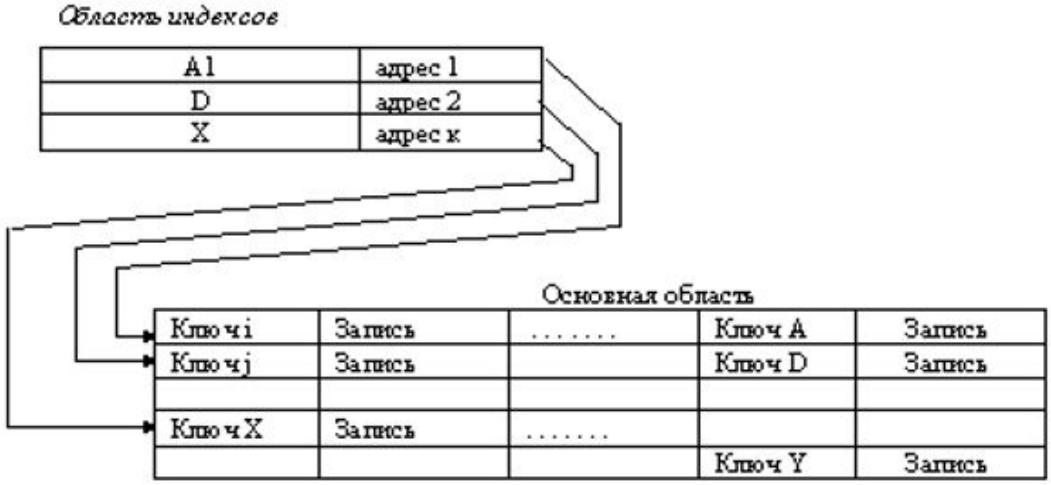

| Индекс | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Адрес | 21 | 201 | 315 | 661 | 670 | 715 |

*Индекс ≡ ключ*

*Способы логической организации файлов*

Структуры:

1. *Файлы с последовательной структурой*
   - Файл - одномерный массив составных элементов называемых записью.
   - Длина записей может быть постоянной и переменной, имеет порядковый номер в составе файла.
   - Доступ последовательный(после обработки i-й записи доступна только i+1)
   - Для организации такого доступа необходимо иметь 1 указатель.
2. *Файлы с индексно-последовательной структурой*
   - Существует ряд методов организации, основанных на идентификации файлов по ключу(индивидуальному отличительному признаку).
   - Кроме обычных данных используют учетную информацию(метаданные).




- Поиск данных идет сначала в прямом, а потом в последовательном порядке.
- Записи упорядочиваются по значению ключей.
- Выделяют группы записи ключей, которые располагаются подряд в файле и могут храниться в пределах одной дорожки на диске.
- Для доступа к таким дорожкам существует *Индекс*.
- Недостаток: проблема расширения файла из-за требования упорядоченности по ключам. Решение: вводится область переполнения.

*3. Библиотечная структура файла*

- Уровни: учетный, информационный.
- Файл - совокупность последовательных наборов данных, где каждый набор имеет собственное имя в начале файла.
- Расположение разделов не упорядоченно, они записываются в порядке поступления.
- Расположение разделов хранят в каталоге учетного файла. Элементы каталога расположены в алфавитном порядке.

## *2. Типы и свойства операционных систем.*

*ОС* - организованная совокупность программ, предоставляющая пользователю набор средств, позволяющих упростить программирование, отладку и сопровождение программ; предоставляет собой интерфейс между пользователем и аппаратной частью машины.

*Типы ОС:*

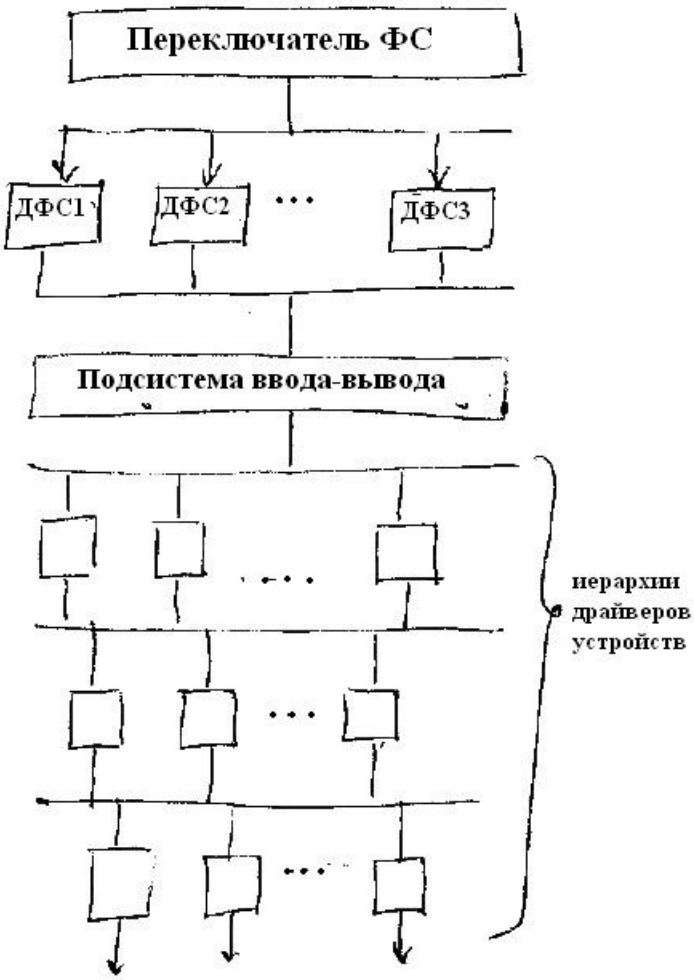
1. ОС машин ЭВМ *общего назначения*. Предназначены для широкого круга пользователей. Обладают высокой универсальностью.
2. ОС *реального времени* (ОСРВ). ОС такого типа предоставляют необходимые для работы функции системе реального времени, которая работает на конкретном оборудовании. Например, RTLinux (микроядерная операционная система жёсткого реального времени, которая выполняет Linux как полностью вытесняемый процесс.) Основные особенности ОСРВ: возможность управления датчиками, широкий спектр этих средств, используются упрощенные алгоритмы обработки, высокая надёжность.
3. ОС *мобильных или портативных устройств*. Нищеброды могут расскать про Андроид, миллионеры про iOS.
4. ОС *специального назначения* (бортовые, т.е. для танков, самолётов. Можно мега-сильно подлизнуть, сказав, что Пугач говорил на лекции про своего студента, который писал ОС для танка).

*Свойства ОС:*

1. *Надёжность* - ОС должна быть надёжна, как и аппаратура, на которой она работает. ОС должна обладать средствами диагностики и восстановления после различного вида сбоев.
2. *Защита* - ОС должна обеспечить защиту от взаимного влияния задач пользователя друг на друга.
3. *Предсказуемость* - реакция ОС на запросы пользователя должна быть предсказуемой и не должна варьироваться.
4. *Удобство* - ну тип должна быть удобна!!!
5. *Эффективность* - ОС должна эффективно распределять и использовать ресурсы. Если начнёт докапываться, что значит эффективно, то просто начни лечить чё-нибудь очевидное.(норм примеры эффективного размещения: размещение процессов в оперативной памяти так, чтобы оставалось меньше неиспользуемых дырок, размещение файлов на диске так, чтобы оставалось меньше неиспользуемой памяти)
6. *Гибкость* - ОС должна предоставлять пользователю возможность настройки различных параметров. Совсем круто, если эта настройка еще и гибкая.
7. *Расширяемость* - ОС должна обеспечивать подключение новых модулей (так написано в лекции, что это конкретно значит - я хз. Я бы начал лечить про то, что ОС должна позволять устанавливать различные программы, расширяющие функционал системы.)
8. *Прозрачность(ясность)* - открытость ОС (пользователь должен иметь возможность знать об ОС всё, что захочет).

## *3. Архитектура современной файловой системы. Многоуровневая файловая система.*

**Фа́йловая систе́ма** — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов и (каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла.
Архитектура современной файловой системы такова:




Приложение обращается к такой файловой системе только через переключатель. В WINDOWS 95 такой переключатель называется **устанавливаемым диспетчером файловой системы IFS** (Installable File System manager).

Переключатель преобразует запросы в формат следующего уровня конкретной файловой системы. Каждый диспетчер файловой системы поддерживает определённую организацию файловой системы и позволяет сразу нескольким приложениям выполнять операции с файлами.

ДФС (драйвер файловой системы) – расширение сетевого сервиса, позволяющее объединить в единый логический том сетевые ресурсы, причем расположенных в разделах с различными ФС.

**Подсистема ввода-вывода** отвечает за загрузку, инициализацию и управление всеми модулями низких уровней файловой системы. 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 блокам, драйверы портов и т.д.)

*Многоуровневая файловая система* - это когда тип есть вложения, например, папки в папку. То есть можно что-нибудь про это полечить.

## *4. Организация виртуальной оперативной памяти. Схема структурирования переменными страницами.*

Схема структуризации страниц переменного размера


Схема структуризации переменными страницами.

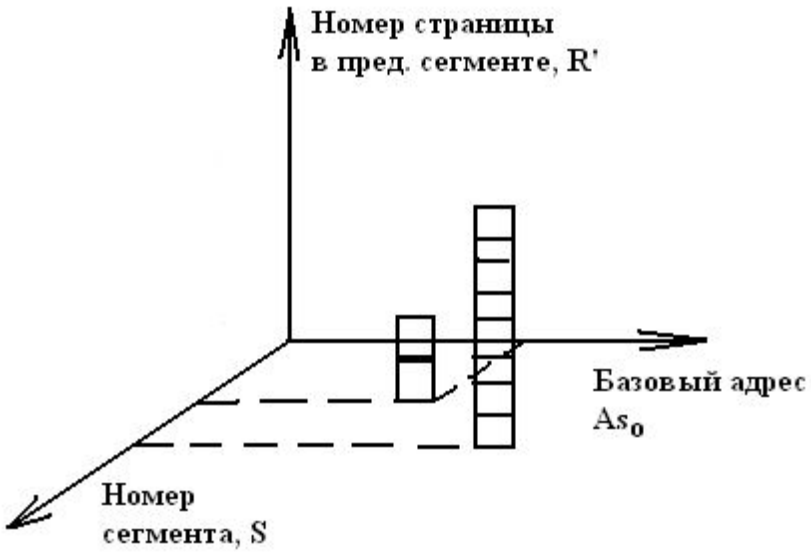
**Короче чтобы не тратить дохера времени на байду написанную здесь, сначала читаем 18 вопрос про страничный метод организации, понимаем его, а потом смотрим, что тут разница лишь в том, что при обращении к памяти используем не номер страницы, а базовый адрес страницы, так как размер страниц разный. Собственно и вся разница.**

Максимальный размер страницы L=4 вместо номеров страниц их базовые адреса страниц. Здесь отсутствует взаимооднозначное соответствие между структурированным и непрерывным адресами. Для получения значения непрерывного адреса (Ai, R) необходимо использовать выражение A=Ai+R. По значению непрерывного адреса нельзя сказать к какой странице он принадлежит.

## *5. Организация виртуальной оперативной памяти. Схема сегментно-страничной структуризации.*

**Этапы:**

1. Исходное пространство структурируют исходными страницами.
2. Сегмент рассматривается как уже 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 в возрастающем.
5. Сегменту назначается базовый адрес As0. В итоге адрес указывается с помощью четырех координат (S, As0, R', R).




Базовый адрес в составе сегмента определяется $A_{R'} = A_{S_0} + R'L$. L – номер страницы.

Далее, если размер страницы был кратен 2, то к базовому адресу страницы применяют операцию конкатенацию (присоединяют значение смещения).

**Механизм преобразования виртуального адреса в физический.**

Для каждого сегмента создается своя таблица страниц. Адрес таблицы загружается в специальный регистр процессора в тот момент когда процесс становится активным.

**Основные цели страничной и сегментной организации.**

Страничная организация ориентирована в первую очередь на удовлетворение нужд системы. Она позволяет улучшить использование оперативной памяти за счет уменьшения объемов пересылок между рабочей и архивной средами.

Сегментная организация в первую очередь ориентирована на пользователя и на исполнение сложных многомодульных программ в мультипрограммном режиме.

## *6. 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.

## *7. События. Семафоры. Сообщения. Их основное назначение.*

При решении общих задач процессы должны иметь возможность обмениваться данными. Передаваемая последовательность данных называется сообщением.

Синхронизация - сигнализация между процессами по определенному протоколу(набор правил и соглашений). Такая операция не зависит от времени, её принято называть событием.

~~Событие~~ Сообщение - объект синхронизации, который используется для информирования потоков о том, что произошло событие.

Каждое событие представляет собой структуру данных, которая содержит код, обозначающий тип события: движение мыши, нажатие кнопки и т.д., а также поля, различные для различных типов событий: для "мышиных" событий это текущие координаты мыши и битовая маска, обозначающая состояние кнопок (нажата/отпущена) и тд.

В ОС возможно возникновение нескольких событий. Любому событию присваивают идентификатор - флаг события, который определяет возможность синхронизации.

С любым событием связано статусное слово, числовое значение. Если статус < 0, то событие прекращено из-за ошибки. При этом статусное слово содержит код ошибки. Если статус > 0 - успешное завершение, = 0 - событие еще продолжается.

*Процесс* - совокупность последовательных действий, необходимых для достижения какого-либо результата в вычислительной системе.

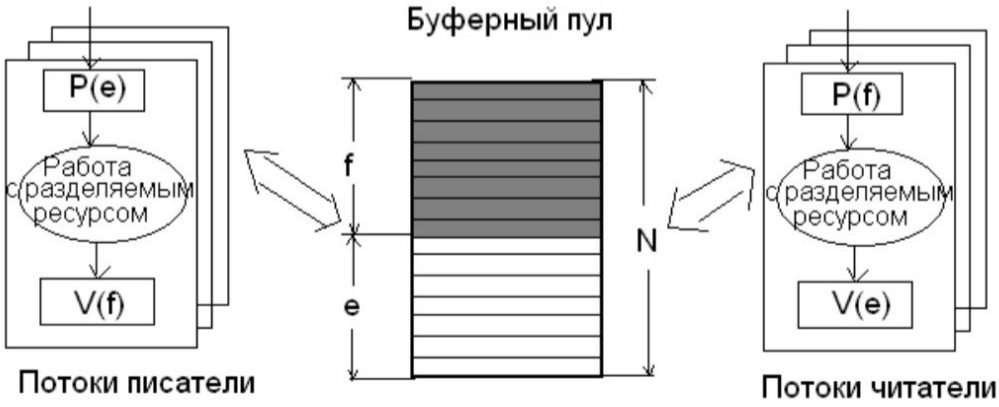
При решении общих задач процессы должны иметь возможность обмениваться данными. Такую последовательность называют сообщениями.

Обмен между двумя процессами может быть разделен на 2 класса:

- **1 класс - Разделяемые переменные**(напр. семафоры и события).

Семафор - счетчик числа доступных ресурсов.(Представляет собой неотрицательные целые переменные, используемые для синхронизации процессов).

ПРИМЕР ДЛЯ ПОНИМАНИЯ:




Для семафоров определены 2 операции: p и v(см. рисунок). Пусть переменная S - семафор, тогда p(s) уменьшает s на единицу, v(s) - увеличивает. Если уменьшить на 1 невозможно, то поток, вызывающий операцию p, ждет пока это станет возможно. Когда семафор может принимать только значения 0 или 1 - он становится блокирующей переменной(двоичным семафором). Размер буферного пула - N, семафор f - число заполненных буферов, e - пустых. В исходном состоянии e = N. ВЫВОДЫ: использование двоичное переменной позволяет организовать доступ к ресурсу только одному потоку, а семафор синхронизирует работу нескольких потоков.
ПРИМЕР ЗАКОНЧЕН.

Мьютексы похожи на семафоры, используются 2 операции: захват и освобождение мьютекса.
Отличие от семафоров: мьютекс может быть захвачен не более чем одним потоком управления.(подробнее 26 вопрос).

- **2 класс - сообщения** использующиеся в распределенных системах, когда процессы имеют собственную ОП, а использовать разделяемую память невозможно.

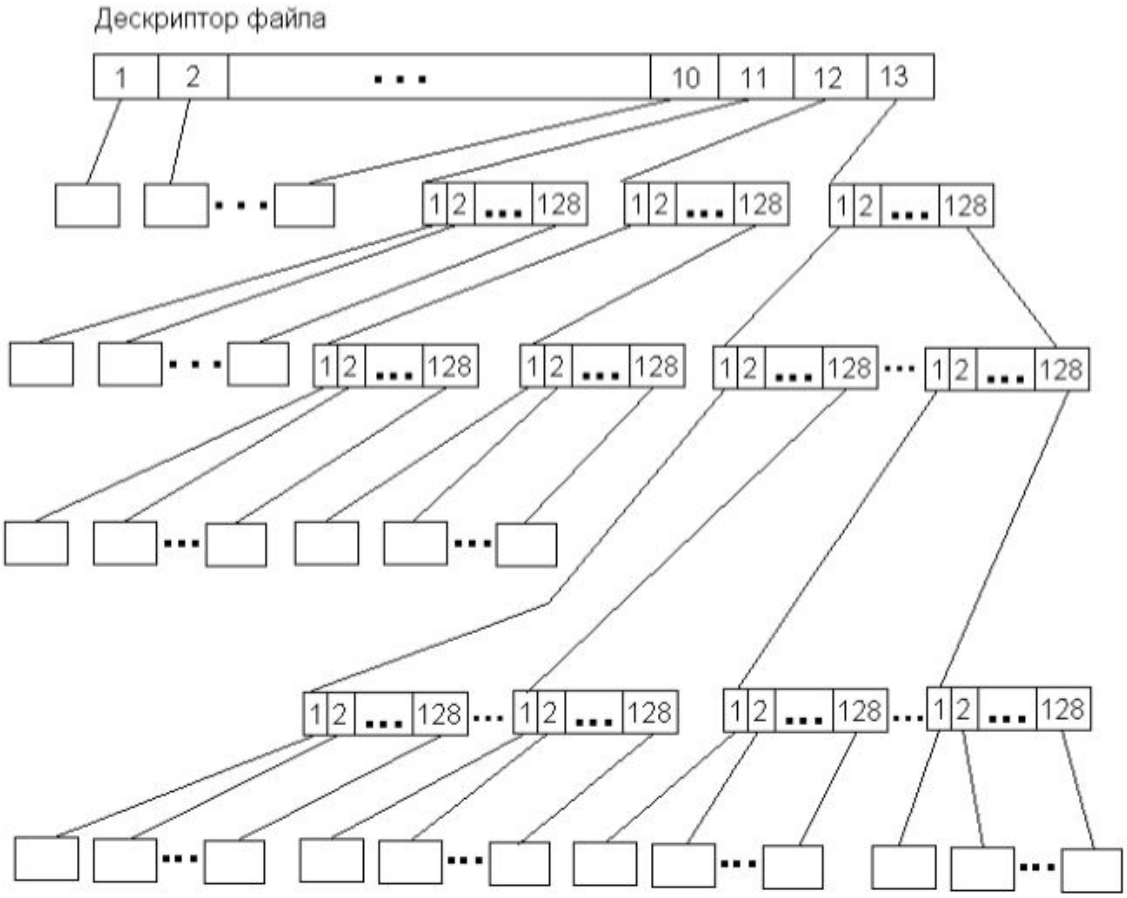
При обмене сообщениями возможны 3 подхода:

1. процесс может обмениваться сообщениями только с процессом, имеющим общего прародителя.
2. с любыми процессами
3. комбинированно

## *8. 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.




## *9. Система распределения оперативной памяти. Алгоритм оптимального размещения.*

При распределении ставят 2 цели:

1. Если памяти недостаточно для нормальной работы, то цель - эффективная загрузка оперативной памяти.
2. Если памяти достаточно, то цель - повысить эффективность доступа к данным.

В случае дефицита памяти, используется подход, при котором пользователь работает с оперативной памятью не на физическом, а на более высоком - логическом или виртуальном уровне.

Если ОС двухуровневая, то на верхнем уровне память распределятся статически, а на нижнем - динамически.

На любом из уровней решается 3 взаимосвязанные задачи:

1. Выделение памяти
2. Учёт памяти
3. Возврат памяти

Ядро ОС обычно размещается в начальных адресах памяти.

Цель алгоритма распределения ОП - минимизация размера и количества пустых участков памяти. Примечание: независимо от алгоритма, всегда есть потери из-за фрагментации.

Алгоритм оптимального размещения.

Для запроса выделяется участок минимально возможного размера. Идея алгоритма: минимизировать объем остающегося после любого распределения свободного пространства.

(Vz)^2 = (Vz)^1 - (Vx)^1 => (Vz)^3 = (Vz)^2 - (Vx)^2, что и требовалось доказать. Это образец невиданной математики от Пугачева, хз как и откуда это следует.

*P.S. Это не какое либо супермататематическое преобразование, тут лишь тривиально описывается то, что после каждого выделения памяти отщипываем кусочек от предыдущего свободного участка. То есть степени тут обозначают первое распределение, второе и тд. (с) Артур*

Vz - размер свободного участка ( остаток )

Vx - требуемый размер участка на определенном шаге

Каждый раз остаток памяти V сравнивается с некоторой пороговой величиной и принимается решение производить такое распределение или нет. 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.

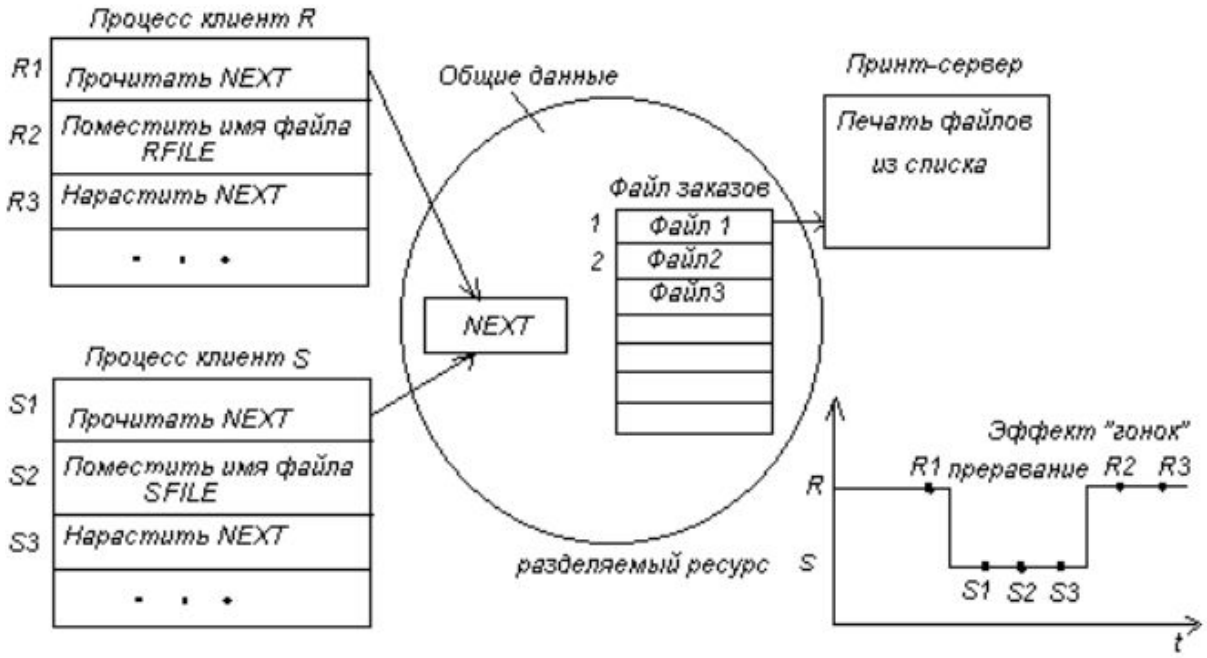
Списки ~~твоих~~ ~~бывших~~ дыр могут быть упорядоченными или неупорядоченными. Чаще всего они упорядочены в порядке возрастания начальных адресов дыр. Преимущества легко обнаружить смежные дыры, которые можно объединить в одну. Списки могут быть двунаправленными, что позволяет сократить среднее время поиска.

Поиск на практике осуществляется по принципу первая подходящая дыра. Я так девушек ищу обычно. Причём, поиск всегда начинается с начала списка. Существует другой принцип, который называется самая подходящая дыра.

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

1. *Синхронизация и взаимодействие процессов. Эффект “гонок”. Критическая секция*

Многие процессы находятся во взаимосвязи, поэтому необходимо обеспечить их взаимодействие. Необходимость в синхронизации возникает напрямую в программе печати файлов ( принт-сервер ).




Действие процессов связаны с критической секцией; здесь страдает процесс S. Видим, что сначала процесс R читает имя файла из переменной NEXT. Распечатать файл он не успевает так как прерывается процессом S. Процесс S считывает опять переменную NEXT, получает тот же номер позиции, что и R, и помещает в эту позицию имя своего файла. Потом управление возвращается к процессу R и, так как он хранит тот же номер позиции, куда было записано имя файла процесса S, то он запишет имя своего файла поверх того файла. В итоге имя файла, который должен был быть напечатан процессом S, было утеряно.

Критическая секция - часть программы, в которой осуществляется доступ к разделенным данным (ресурсам).

*2. Взаимное исключение. Способы обеспечения взаимного исключения*

Взаимное исключение - исключение эффекта “гонок”. Обеспечивает, чтобы в каждый момент в критической секции находилось не более одного процесса.

Простейший способ обеспечить взаимное исключение - позволить процессу, находящемуся в критической секции запрещать все прерывания. Но доверять управление самому пользовательскому процессу опасно. Особая переменная *NEXT* доступна всем процессам и она содержит номер первой свободной позиции файла заказов для записи имени файла на распечатку.

Другой способ обеспечить взаимное исключение - использование блокирующих переменных. Операции проверки и установки блокирующей переменной должны быть неделимыми.

## *11. Эволюция операционных систем.*

1. *ОС с пакетной обработкой*

В данном типе ОС есть оператор, который формирует пакет, который состоит из совокупности отдельных программ и данных пользователя. Также есть программа, которая считывает пакеты, запускает их на выполнение, отслеживает аварийные ситуации, фиксирует время. Вот такую программу как раз и можно назвать операционной системой, так как она фактически исполняет её основные функции.

2. *ОС с мультипрограммной обработкой*

Такие ОС обеспечивают эффективное использование ресурсов несколькими пользователями, т.е. в оперативной памяти содержится несколько пользовательских программ, процессорное время разделяется между программами. Одновременно могут работать процессор, ОП, каналы (мультиплексные, селекторные) и внешние устройства.

3. *ОС с разделением времени*

В таких системах преследуется следующая основная цель: обслужить каждого пользователя и обеспечить допустимое время реакции на его запрос. Пользователь имеет непосредственный доступ к ЭВМ. Вводится детерминизм (схема по мультиплексированию центральной памяти процессора среди программ, готовых на выполнение), то есть каждой программе отводится свой интервал времени.

4. *Многопоточные и многопроцессорные системы*
5. *Микроядерные ОС.* В микроядре изолирована вся машинно-зависимая часть ОС.

## *12. Процесс. Основное понятие. Дескриптор процесса. Виды групп информации дескриптора.*

*Процесс* - совокупность последовательных действий, необходимых для достижения какого-либо результата в вычислительной системе. В современных операционных системах процесс реализуется как динамический объект, который имеет большое количество полей и методов, т.е. атрибутов и сервисов. Он может находиться в различных состояниях, в отношении с другими объектами, т.е. иметь родственные связи, занимать и освобождать необходимые ресурсы. Процессы могут быть последовательными относительно друг друга, параллельными и комбинированными.

*Процесс* - логическая единица работы операционной системы.

Процесс как логическая единица предполагает наличие двух аспектов:

1. Он выполняет операции ( различные действия ).
2. Он сам по себе является носителем данных.

Когда процесс заканчивает свою работу, вызывается деструктор.

Можно выделить 2 части процесса:

1. У процесса есть программа, по которой он работает в активном состоянии.
2. *Дескриптор*, который представляет собой информационную структуру, в которой сосредоточена управляющая информация, необходимая для планирования ( или для планировщика ) и управления процессами.

Очереди процессов организуют через список.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенных в списки.

По функциональному назначению можно выделить несколько групп информации:

1. Информация по идентификации - содержит уникальное имя или номер процесса, нужна для того, чтобы можно было отслеживать и выполнять операции как над поименованными объектами.
2. Информация о ресурсах, которые необходимы процессу для его работы. Конструктор процесса инициализирует его поля или ...
3. Информация о состоянии процесса. Эта информация позволяет определить текущее состояние и возможность перехода в следующее.
4. Информация о родственных связях. Она нужна, чтобы при использовании ресурсов родственника не произошло конфликтов.
5. Информация, которая необходима для учета и планирования процессов. Здесь указываются ссылки на средства синхронизации между процессами, приоритет и др.

Контекст процесса - это информация о процессе, которая необходима ему в активном состоянии ( чтобы он мог работать ).

## *13. Физическая структура файла. Способы размещения информации: непрерывное размещение и перечень номеров блоков. Достоинства и недостатки.*

Расположение информации на конкретном носителе чаще всего значительно отличается от его логической упорядоченности.

Преобразования логической структуры в физическую структуру осуществляется на основе информации сосредоточенной в каталогах файла, и в специальных описателях и дескрипторах.

На практике для управления внешней памятью используются те же алгоритмы, но отличия будут в способе реализации.

Внешняя память разбивается на блоки фиксированного размера.Каждый блок имеет свой уникальный порядковый номер.Каждый блок- минимальная единица данных, которая участвует в обмене между устройством внешней и оперативной памяти.

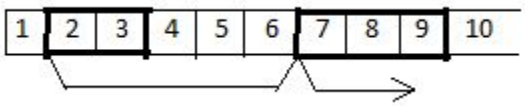
Способы размещения информации:

*1) Непрерывное размещение*

Файл состоит из последовательных блоков


Недостаток: трудно расширить; неэффективное использование дискового пространства.

*2) Связный список блоков*




- адрес файла задается номером первого блока
- фрагментация практически отсутствует
- файл может расширяться

Недостатки:

- доступ медленный
- сложность реализации доступа к произвольной части файла
- больше памяти на служебную информацию

*3)Связный список индексов*

Хранятся в этом способе указатели не в дисковых блоках, а в индексной таблице в памяти, которая называется таблицей отображения файлов (FAT - file allocation table). По-прежнему существенно, что запись в директории содержит только ссылку на первый блок. Далее при помощи таблицы FAT можно локализовать блоки файла независимо от его размера. В тех строках таблицы, которые соответствуют последним блокам файлов, обычно записывается некоторое граничное значение, например EOF.

Главное достоинство данного подхода состоит в том, что по таблице отображения можно судить о физическом соседстве блоков, располагающихся на диске, и при выделении нового блока можно легко найти свободный блок диска, находящийся поблизости от других блоков данного файла. Минусом данной схемы может быть необходимость хранения в памяти этой довольно большой таблицы.

| Номера блоков диска | | |
|---|---|---|
| 1 | | |
| 2 | 10 | |
| 3 | 11 | Начало файла $F_2$ |
| 4 | | |
| 5 | EOF | |
| 6 | 2 | Начало файла $F_1$ |
| 7 | EOF | |
| 8 | | |
| 9 | | |
| 10 | 7 | |
| 11 | 5 | |

**Рис. 12.3.** Метод связного списка с использованием таблицы в оперативной памяти

*4) Перечень номеров блоков*

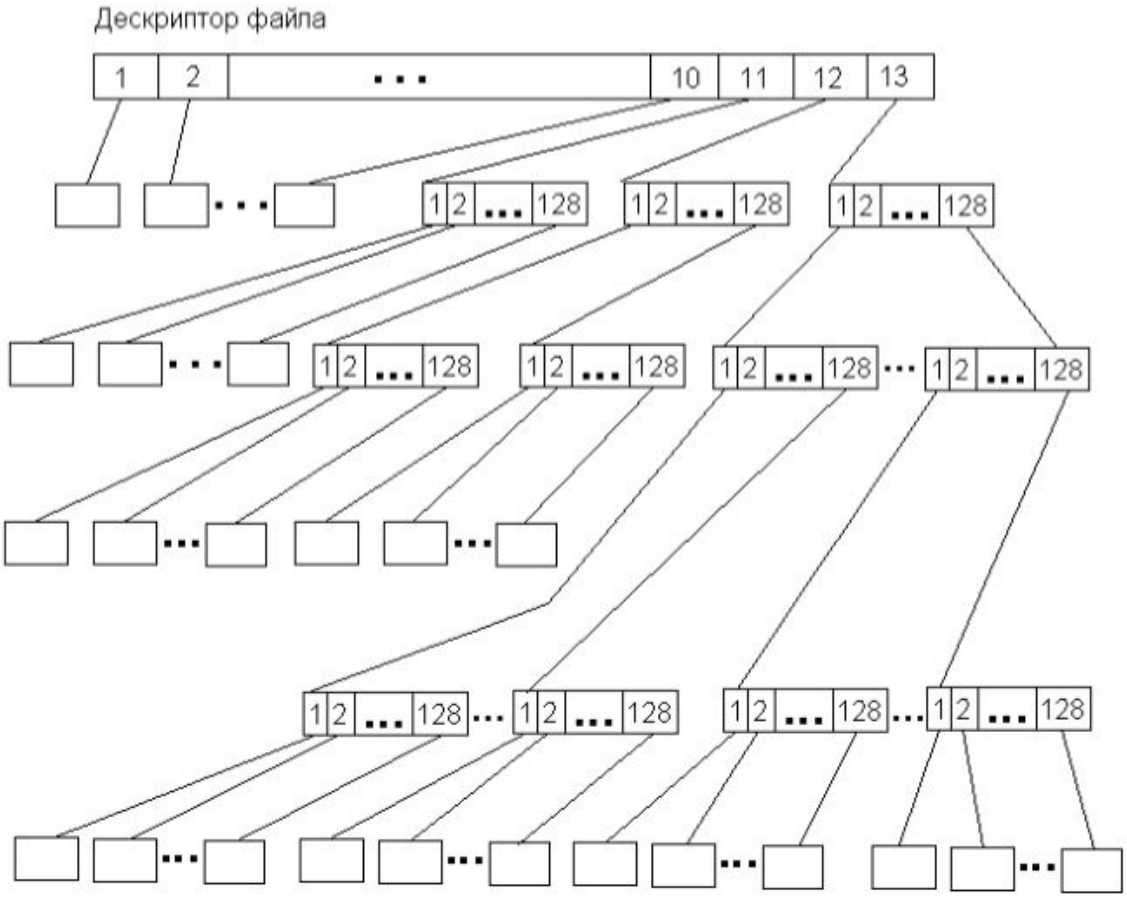
- номера блоков, занимаемых файлами просто перечисляются

Недостатки :

- усложняется алгоритм распределения
- время поиска увеличено

Пример;

В Linux реализован вариант, который позволяет обеспечить фиксированную длину адреса не зависимо от размера файла. Каждый файл в системе имеет дескриптор, содержащий 13 номеров блоков диска.



## *14. Файловая система. Кэширование диска. Механизм кэширования диска.*

Файловая система - часть операционной системы, которая обеспечивает выполнение операций над файлами.

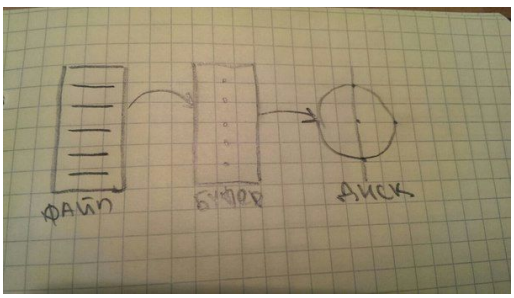
Основные задачи файловой системы:

1. Обеспечение независимости программы от конкретной конфигурации машины.
2. Сокрытие от пользователей реального расположения информации на физическом уровне

Функции файловой системы:

1. Реализация методов организации данных.
2. Перевод логический характеристик в физические
3. Защита пользователя от случайных сбоев
4. Возможность многим пользователям разделять одно устройство
5. Восстановление файлов
6. Защита от несанкционированного доступа.

Механизм кэширования диска - создаём файл, пишем в него информацию, если файл не закрыть то информация из буфера не запишется на диск. Дефрагментация меньше, следовательно, работает быстрее такая штука.




Это уже не из лекций, но, для общего понимания пойдет. КЭШИРОВАНИЕ ДИСКА - Хранение в оперативной памяти наиболее часто используемых секторов диска с целью увеличения скорости обмена данными между диском и оперативной памятью. В оперативной памяти операционной системой выделяется специальная область, в которой организуется буферная память - кэш. В нее помещаются данные, считанные с диска, и сохраняются там до тех пор, пока не будут вытеснены другими данными. Когда данные потребуются повторно, они могут быть считаны не с диска, а из кэша, а значит, быстрее. Кэширование применяется не только при чтении, но и при записи. В этом случае данные могут сначала обновляться в кэше, а затем в "удобное" время копироваться на диск. Таким образом, К. д. не только увеличивает производительность компьютера, но и продлевает срок службы накопителей.

## *15. Логическая организация файла. Файлы с индексно-последовательной структурой.*

Пользователь работает на уровне логических записей - наименьший элемент данных, который доступен пользователю для выполнения операций.

Файлы с индексно- последовательной структурой

- Существует ряд методов организации основанных на идентификации файлов по ключу (индивидуальному отличительному признаку)
- Кроме обычных данных используют учетную информацию ( метаданные)


- Здесь поиск данных идет сначала в прямом, а потом в последовательном порядке.
- Записи упорядочиваются по значению ключей.
- Выделяют группы записи, ключи которых располагаются подряд в файле и могут храниться в пределах одной дорожки на диске.
- Для доступа к таким группам существует индекс(индекс дорожки)

Индекс - может содержать значение максимального ключа в группе и ссылку на начальную запись в группе; по нему находят начало первой записи требуемой группы.

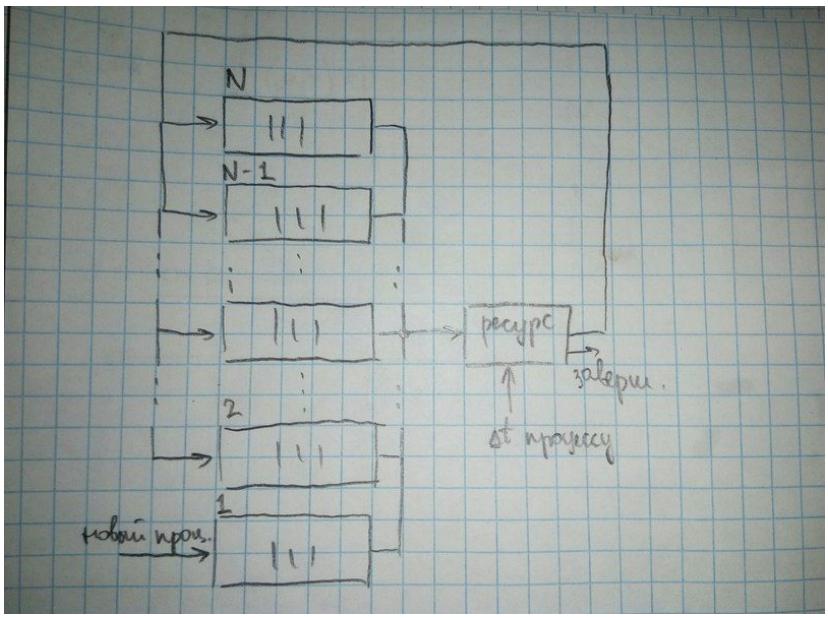
Недостаток структуры: проблема расширения файла из-за требования упорядоченности по ключам

Решение: вводится область переполнения, и из основной области ссылку в область переполнения.

## *16. Многоочередные дисциплины обслуживания процессов. Простая и приоритетная дисциплины.*

Многоочередные дисциплины:

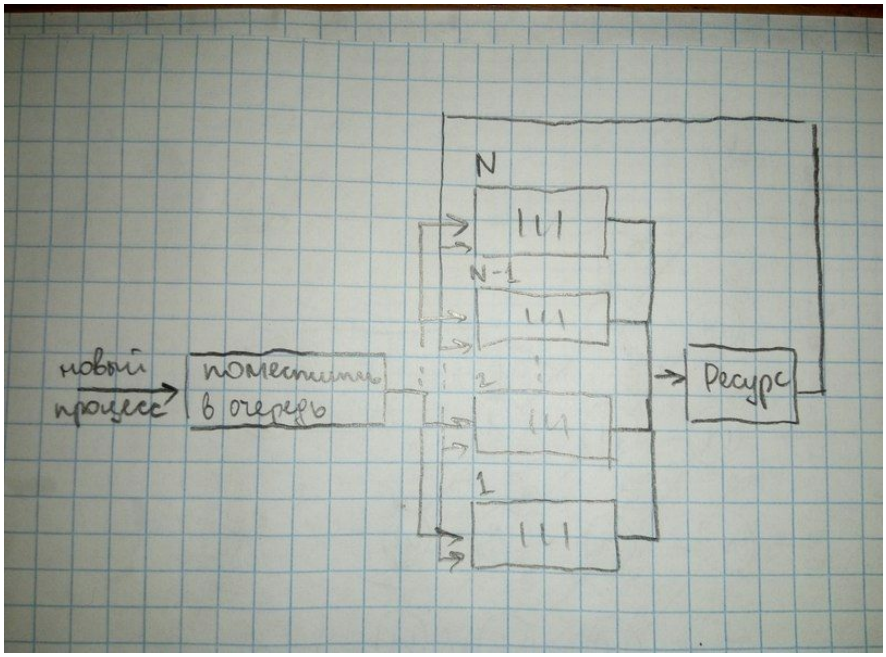
1. Простая многоочередная дисциплина обслуживания




Процесс попадает в конец 1-ой очереди, если ему не хватило кванта, то отправляется во вторую очередь и так далее в (i+1) очередь.

Первый процесс из очереди i попадает на обслуживание, только если все очереди от 1 до (i-1) пустые.

2. Приоритетная многоочередная дисциплина обслуживания




Новые процессы попадают в очередь в соответствии с приоритетами, которые имеются в параметрах процесса или в дескрипторе (мы знаем, что параметры хранятся в дескрипторе).

В большинстве ОС алгоритмы планирования построены как с использованием квантования, так и приоритетов. Например, в основе планирования может лежать квантование, но величина кванта и порядок выбора из очереди определяется приоритетами процессов.

## *17. Вытесняющие алгоритмы планирования процессов.*

Используется стратегия, при которой текущий процесс может быть вытеснен другим процессом.

Например, после обработки прерывания может обрабатываться процесс с более высоким приоритетом, вытесненный процесс должен быть обработан планировщиком.

Стратегия с вытеснением может сочетаться со стратегией без вытеснения, для этого требуются дополнительные флаги. Кроме приоритетов для каждого процесса вводятся дополнительно 2 флага.

Например: U - 1 означает, что процесс может захватить другой процесс, а 0 означает, что не может. V - 1 означает, что процесс может захватиться другим процессом, а 0 означает, что не может.

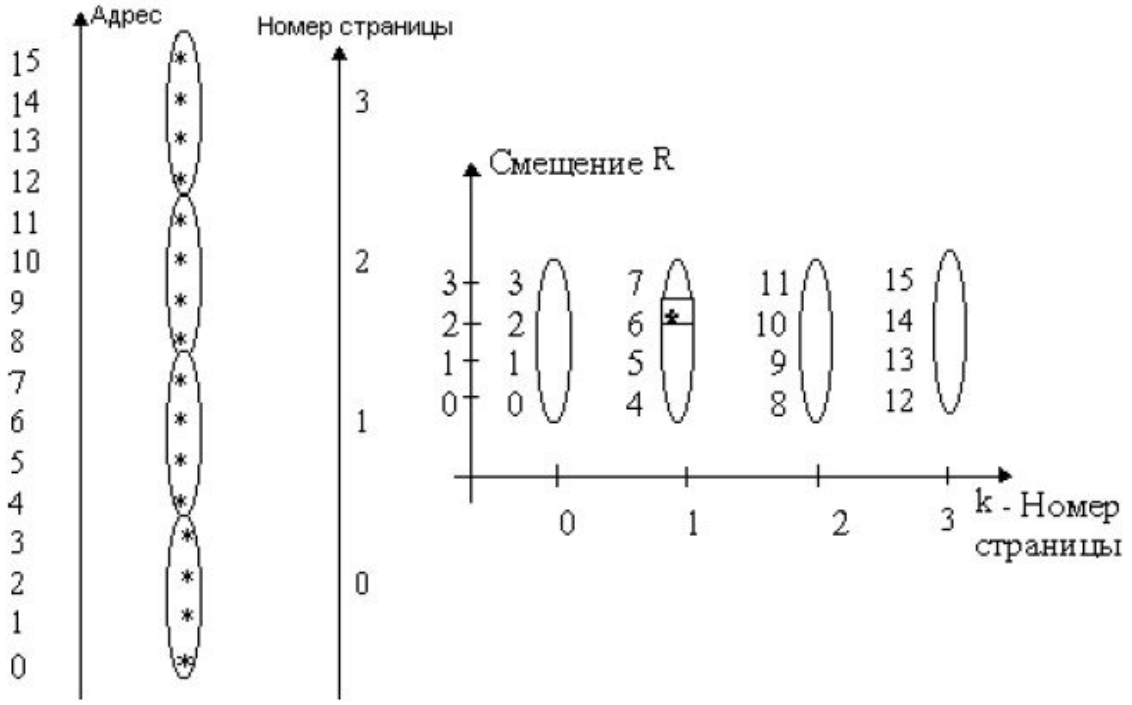
Этот алгоритм применим только к процессам с фиксированными приоритетами.

## *18. Организация виртуальной оперативной памяти. Схема структурирования фиксированными страницами. Механизм работы.*

Различают 2 класса схем структуризации адресного пространства:

1. схема страничной структуризации
2. схема сегментной структуризации

Ниже приведена схема структурирования фиксированными страницами.




Если размер адресного пространства кратен 2, то и размер страницы также выбирают кратным 2. Реально в ОС 512, 1024 и выше. Это позволяет упростить механизм преобразования адресов.

В каждую страницу входят одинаковое число адресов L.

Для перехода из страничного в исходное одномерное адресное пространство используется выражение Аф = K * L + R для вычисления адреса.

Например, от адреса (1,3) мы используем A=(1*2)^2+3=7. Чтобы перейти в непрерывный адрес нужно выполнить два действия: умножение и сложение. Можно обойтись одним действием, если воспользоваться двоичным представлением номера страницы и смещение. Из двух двоичных чисел с помощью конкатенации получают третье число. Например, (01) и (11) получается (0111)->(7).

*Механизм работы страницы.*

Во время загрузки процесса часть его виртуальных страниц помещается в ОП, а остальные на диск. Причём, смещение виртуальной страницы не обязательно располагается в смежных физических страницах. ОС создаёт информационную структуру, которая называется таблица страниц. В этой таблице устанавливается соответствие между номерами виртуальных и физических страниц. (Но это только для страниц, загруженных в ОП). Или может делаться отметка о том, что виртуальная страница выгружена на диск. В таблице страниц содержится *следующая управляющая информация***:**

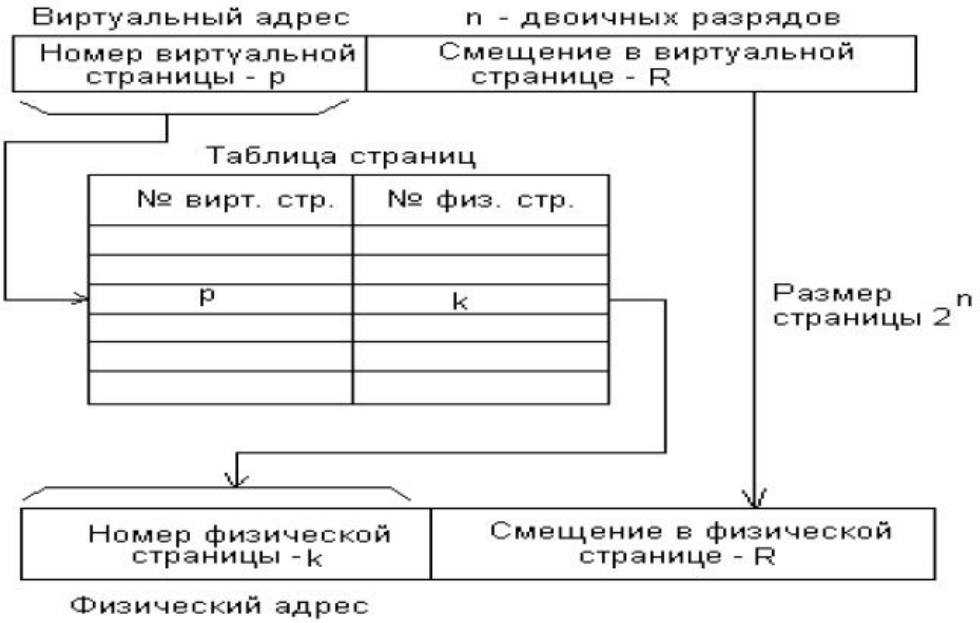
*1-Признак модификации страницы.*

*2-Признак запрета на выгрузку* (выгрузка некоторых страниц может быть запрещена).

*3-Признак обращения к странице* (используются для подсчёта числа обращений за определённый период времени).

*4-Время преобразования виртуального адреса в физический* (в основном определяется временем доступа к страницам).

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. Но при этом расширение за границы физического пространства не происходит.




Механизм преобразования виртуального адреса в физический при страничной организации.

При обращении к ОП происходит следующее действие:

1. Зная начальный адрес таблицы страниц ( он хранится в регистре адреса страниц ), зная номер виртуальной страницы ( старшие разряды ) и зная длину записи в таблице (системная константа ), находим нужную запись
2. Считываем номер физической страницы ( K )
3. К номеру физической страницы присоединяют смещение ( младшие разряды ) с помощью операции конкатенации

## *19. Структура операционной системы на примере Windows NT. Назначение основных модулей.*


Структура Windows NT.

*1-ый уровень (верхний):*

*Защищенные подсистемы или серверы* - каждый из серверов это отдельный процесс, память которого защищена от других процессов с помощью системы виртуальной памяти. Они

предоставляют исполнительной системе ( нижнему уровню ) пользовательские и программные интерфейсы, обеспечивают среды для выполнения приложений различных типов.

Интерфейс прикладных програм API - набор процедур, которые вызываются прикладной программой для осуществления низкоуровневых операций. API реализуется на отдельных серверах ( WIN32 ). Это позволяет устранить конфликты и дублирование в исполнительной системе.

*WIN32* - предоставляет API 32-х разрядной Windows, реализует графический интерфейс, управляет вводом/выводом. Остальные подсистемы имеют свои API, но используют для получения пользовательского ввода/вывода WIN32.

*Подсистема защиты* регистрирует правила контроля доступа на локальном компьютере, то есть ведет БД учётных записей, отвечает за идентификаторы, пароли, привилегии пользователей.

*2-ой уровень:*

*Исполнительная система* - является законченной ОС и выполняет функции низкого уровня, имеет два набора функций: системные сервисы и внутренние процедуры.

*Диспетчер объектов* создает, поддерживает и уничтожает объекты, которые предоставляют системные ресурсы.

*Справочный монитор защиты* оберегает ресурсы, обеспечивает защиту объектов этих ресурсов и аудит во время выполнения (возможность обнаружения и регистрации важных событий, которые относятся к защите).

*Диспетчер процессов* - создает, завершает процессы и потоки.

*Средство локального вызова процедур* (LPC) - передает сообщения между клиентскими и серверными процессами, расположенными на одном и том же компьютере.

*Диспетчер вирутальной памяти* реализует виртуальную память, управляет ей, предоставляет собственное адресное пространство каждому процессу, отвечает за подкачку страниц в памяти.

*Ядро системы* реагирует на прерывания, направляет потоки на выполнение, осуществляет межпроцессорную синхронизацию, скрывает процессорные различия от остальной части ОС.

Ядро имеет следующие компоненты:

1. Пользовательский ввод/вывод
2. Интерфейс графических устройств ( GDI )
3. Ядро ( Kernel ) обеспечивает базовые функции системы.

*Диспетчер ввода/вывода* :

Диспетчер I/O реализует аппаратно независимые средства I/O. *Файловая система* – драйвер, принимающий запросы файлового I/O и транслирующий их запросы, привязанных к конкретному устройству. *Драйверы устройств I/O*– драйверы, напрямую работающие с оборудованием для обеспечения операций I/O с устройств или сети. *Сетевой редиректор* и *сетевой сервер* – драйверы ФС, передающие удалённые запросы I/O на соответствующие сетевые компьютеры. *Диспетчер кэша* – используется для «кэширования» операций I/O (дискового или сетевого).

*Реестр* - центральная информационная БД, которая позволила отказаться от системных файлов config.sys, .ini, .bat.

Реестр осуществляет централизованное хранение следующей информации:

1. профилей всех пользователей
2. данных и программ, и типах документов, создаваемых в этих программах
3. значения свойств для папок и значков программы
4. конфигурацию оборудования
5. данные об используемых портах и др.

*Слой, абстрагированный от оборудования (HAL)* – динамически подключаемая библиотека (DLL). Она изолирует исполнительную систему от особенностей аппаратных платформ разных производителей.

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

При распределении памяти ставят две цели:

1) В машинах, где ОП является дефицитным ресурсом, целью является оптимальная 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.

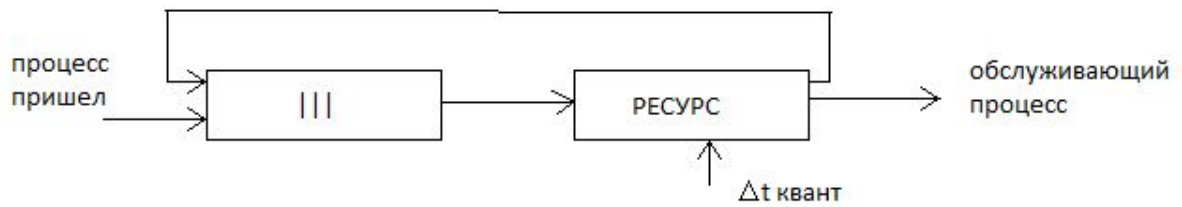
**На каждом уровне решается три взаимосвязанные задачи:**

- *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.

## *21. Алгоритм циклического планирования процессов.*

- в основе лежит FIFO




- процессы выбираются из очереди по порядку
- приоритет определяется линейным положением процесса в очереди

Недостаток: один процесс может занимать ресурс длительное время в ущерб другим

Решение: отводится временной интервал(квант), по окончанию процесс перемещается в конец очереди.

Величины кванта(минимальная) зависит от величины для работы и подготовки процесса.

Данный алгоритм используется в различных вариантах во многих ОС с разделением времени.

А теперь сам алгоритм: каждому процессу в очереди выделяется определенный квант времени процессора. Когда квант времени заканчивается, процесс переводится в конец очереди.

## *22. Организация процессов в операционной системе UNIX.*

Имеется два уровня – верхний и нижний.

Распределение процессора производится по приоритетной схеме. Осуществляется это динамическим изменением приоритета в зависимости от текущих характеристик процесса. Например, на основе оценки соотношения времени t-используемому и t-прогнозируемому.

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.

## *23. Алгоритм приоритетного планирования процессов. Статическое и динамическое приоритетное планирование.*

*Планирование* - управление распределением ресурсов вычислительной системы между различными пользователями и процессами, путём передачи им управления согласно определённой стратегии планирования.

*Приоритет* - число которое характеризует степень привилегированности процесса при использовании ресурсов. Число может быть > 0 , < 0 , дробным.

Каждому процессу присваивается приоритет, который определяет положение процесса по отношению к другим. Процесс с самым низким приоритетом называется холостым, т.к он выполняет пустые инструкции(Сканирование например). Приоритеты разбиваются на группы еще на этапе проектирования. Кол-во групп выбирается так, чтобы во время обработки не происходило скопления процессов в отдельных группах. Границы на число приоритетов могут быть различные.

Выделяют:

**1)Статическое приоритетное планирование** - здесь процессы при создании могут быть разбиты на группы несколькими способами

а)исходя из запросов на ресурсы

б)согласно приоритету программы, которой принадлежит данный процесс

в)приоритет определяется оценкой времени выполнения всей программы

г)приоритет основывается на типе процесса, независимо от ресурсов.

**2)Динамическое приоритетное планирование**

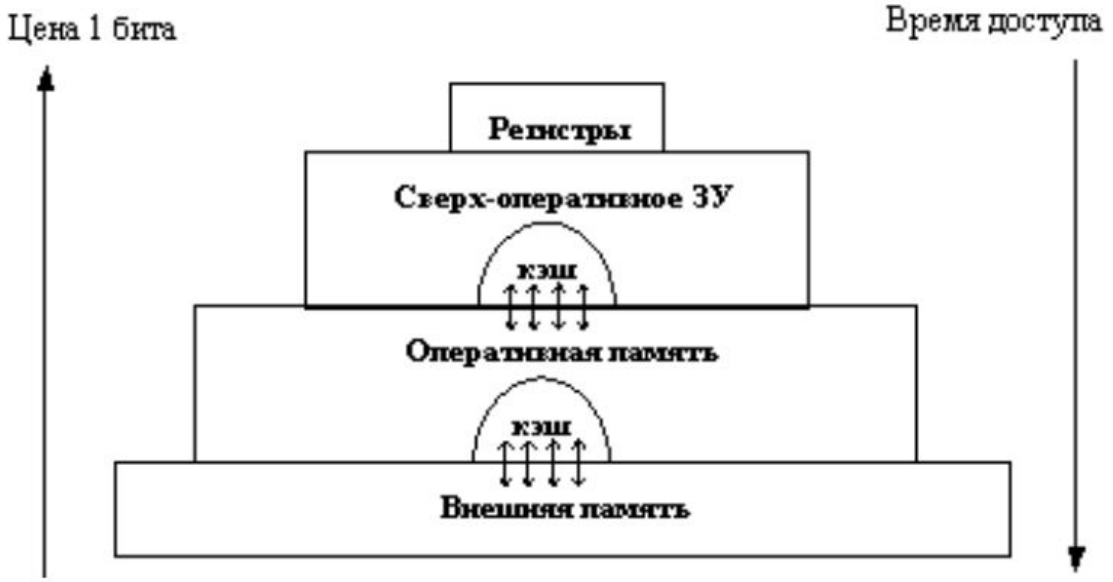
Это когда приоритет процесса изменяется как функция разницы между необходимой услугой и услугой фактически полученной, т.е процесс может перемещаться по приоритетным группам в зависимости от израсходованного времени и ресурсов.

(Другие вида планирования: Циклическое планирование - в основе лежит FIFO.

Адаптивно-рефлективное планирование - контроль над реальным использованием памяти)

Лотерейное планирование(лотерейные билеты на доступ), Вытесняющий алгоритм(текущий может быть вытеснен другими), многоочередные дисциплины,

## *24. Управление оперативной памятью. Свопинг.*




От ОП зависит производительность системы в целом. Организация управления ОП зависит от типа ОС. От кол-ва процессов их активности, а также от общих требований например что любой процесс читает что отведенная ему память начинается с 0 байта. Функции реализующие задачи управления зависят от вида памяти:

Виды:

физическая, логическая,

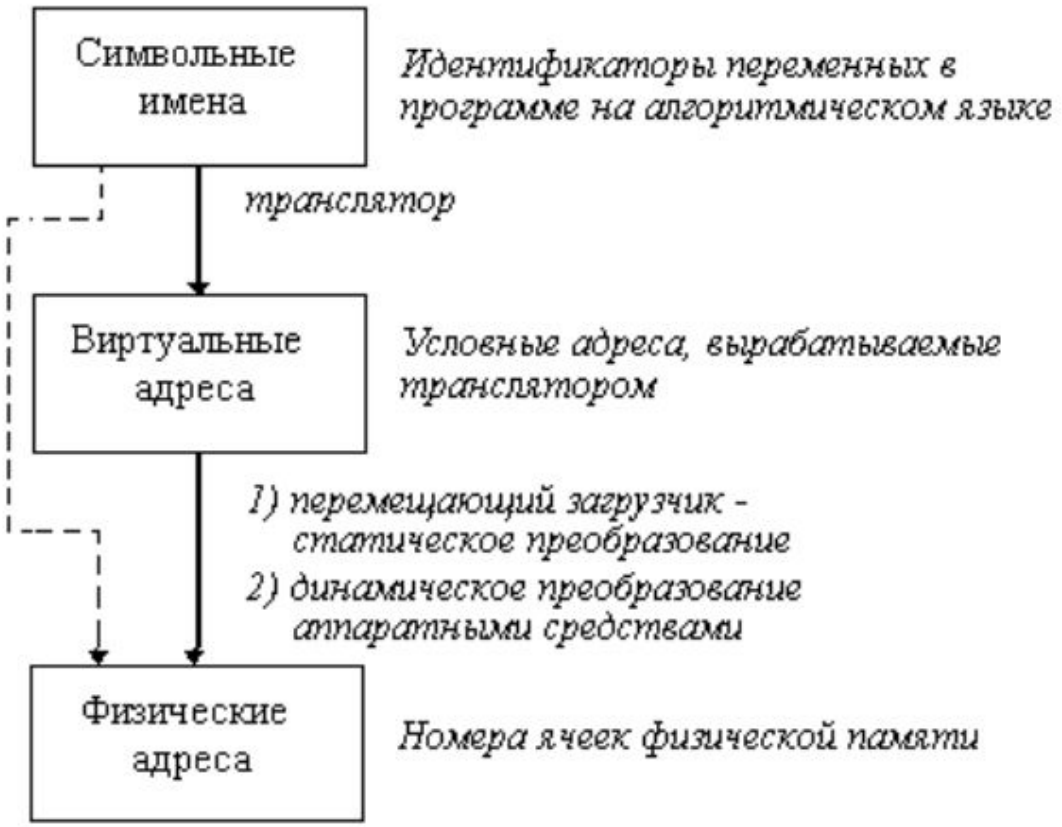



Схема распределения ОП:

При распределении ставят две цели:

1. Если памяти недостаточно для нормальной работы, то цель эффективная загрузка ОП.

2. Если памяти достаточно для нормальной работы, то цель - повысить эффективность доступа к данным.

В случае дефицита памяти памяти, используется подход, при котором пользователь работает с ОП не на физическом уровне, а на более высоком логическом или виртуальном. Если ОС 2-уровневая, то на верхнем уровне память распределяется статически а на нижнем динамически. На любом уровне реализуются 3 взаимосвязанные задачи: выделение, учет и возврат памяти.

Цель алгоритма распределения ОП: минимизация количества пустых участков памяти.(всегда будут потери из-за фрагментации)

Алгоритмы основанные на выделении непрерывной единственной зоны:

1 вариант - ОП разбивают до начала работы ОС на фиксированное число зон, зоны могут быть любые но любой процесс может выполняться только в пределах одной зоны.

2 вариант - Размеры зон могут меняться во время работы, зоны могут формироваться из свободных участков, при этом минимальный размер зон должен быть ограничен.

Основное условие в этих алгоритмах : нельзя выделить зону, размер которой больше ОП.

Задачами ОС при реализации данного метода управления ОП являются:

1)Ведение таблицы свободных и занятых областей в которых указывается начальный адрес и размер участков памяти.

2)При поступлении новой задачи происходит анализ запроса, просмотр таблицы свободных областей, и выбор размера, который достаточен для размещения поступившей задачи.

3)загрузка процесса(задачи) в выделенный ей раздел и корректировка таблиц свободных и занятых участков памяти.

4)После завершения задач идет корректировка таблиц свободных и занятых областей.

(алгоритм оптимального размещения можно почитать)

Для управления ОП необходимо иметь информацию(управляющую или учетную) поэтому центральное место занимает способ учета свободного пространства.

***<u>Алгоритм оптимального размещения.</u>***

$$V_Z^2 = V_Z^1 - V_X^1 \longrightarrow V_Z^3 = V_Z^2 - V_X^2 \longrightarrow и\ т.д.$$

Где $V_z$ - размер свободного участка (остаток);

$V_x$ - требуемый размер участка.

*Свопинг* - Эффективность использования оперативной памяти можно повысить с помощью внешней памяти и наоборот. При нехватке ОП ненужные в данный момент разделы можно копировать на диск. Это называется откачка. Обратный процесс подкачка. А в совокупности эти процессы называются свопинг. Для организации свопинга требуется планировщик памяти. Сложность состоит в том, что один раздел может использоваться несколькими процессами.

Свопинг позволяет:

- заново распределять память для процессов не загруженных сначала.
- выполнять больше малоактивных процессов
- позволяет освободить память, занимаемую процессом, которая требует вмешательства пользователя
- позволяет более эффективно использовать другие ресурсы, кроме ОП, например, при использовании приоритетов.
- в многопользовательских системах когда исполняется один и тот же код в свопинге могут участвовать только данные пользователя.

## *25. Файловая система. Права доступа к файлу.(мне кажется про это надо что-то еще) Основные подходы к определению прав доступа.*

Файл - поименованная целостная совокупность данных на внешнем носителе.

ФС - часть ОС которая обеспечивает выполнение операций над файлами.

Основные задачи ФС -

1) обеспечение независимости программы от конкретной конфигурации машины.
2) сокрытие от пользователей реального расположения информации на физическом уровне.

Функции ФС:

1) реализации методов организации данных
2) Перевод логических характеристик в физические
3) Защита пользователей от случайных сбоев
4) Возможность нескольким пользователям разделять одно устройство
5) восстановление файлов
6) защита от несанкционированного доступа


Уровень идентификации:

- по символьному имени файла определяется его уникальное имя
- по уникальному имени определяют атрибуты файла
- сравниваются полномочия пользователя или процесса с правами доступа файла

Логический уровень:

1) определяются координаты логической записи в файле
2) алгоритм работы зависит от конкретной логической модели

Физический уровень:

Определяется номер физического блока который содержит требуемую логическую запись

Определить права доступа - это значит определить для пользователя дозволенные операции над файлами. В разных ОС определен свой список операций доступа.
основные операции доступа:

- создание
- уничтожение
- открытие
- закрытие
- чтение
- запись
- поиск в файле
- получение атрибутов файла
- установление атрибутов файла

Матрица доступа:


Пользователи могут быть разделены на отдельные группы и подход к назначению прав может быть различным для каждой категории.
В ОС Linux все пользователи подразделяются на 3 типа: владельцы файла, члены группы и все остальные.
Выделяют 2 основных подхода к назначению прав:

1) Избирательный доступ - для любого пользователя и любого файла сам владелец может определять допустимые операции.
2) Мандатный подход - (в зависимости от группы) система устанавливает права пользователя по отношению к любому разделяемому реестру в зависимости от того к какой группе он принадлежит.

## *26. Синхронизация процессов. Мьютексы.*

Синхронизация - сигнализация между процессами по определенному протоколу(набор правил и соглашений) такая операция не зависит от времени, её принято называть событием.

~~Событие~~ Сообщение - объект синхронизации который используется для информирования потоков о том,что произошло событие. В ОС возможно возникновение нескольких событий, любому событию присваивается идентификатор(флаг событий) Он определяет возможность синхронизации.

С любым событием связано статусное слово и числовое значение. Если статус < 0 то событие прекращено из-за ошибки. При этом статусное слово содержит системный код ошибки. Если статус > 0 - удачное завершение. Если статус = 0 - событие еще продолжается.
При решении общих задач процессы должны иметь возможность обмениваться данными. Такую последовательность называют сообщением. Обмен сообщениями имеет сложную структуру и может быть разделен на 2 класса:
*1 класс* - разделяемые переменные (например семафоры и события)

**Семафоры** – это неотрицательные целые переменные, для которых определены две операции:

1. **Операция P** – уменьшение семафора на 1. Если это возможно. Если P=0, то процесс вызвавший операцию ждет, пока уменьшение не станет возможным.
2. **Операция V** – семафор увеличивается на 1, что предотвращает доступ к семафору других процессов. За исключением того, кто эту операцию выполняет. (Разделяемые переменные).

*2 класс* - **сообщение** – обмен информации между процессами имеет 2 ограничения со стороны ресурсов.

Мьютексы похожи на семафоры - исполняются всего 2 операции захват и освобождение мьютекса. Основные отличия мьютекса от семафора : мьютекс может быть захвачен не более чем одним потоком управления, такой поток называют владельцем мьютекса. Освобождение мьютекса может быть произведено только его владельцем, поэтому мьютексы нельзя использовать в функциях обратных прерываний. Если мьютекс захвачен, то при попытке захвата другим потоком , захватывающий поток будет приостановлен и будет находиться в очереди к мьютексу. В этой очереди может находиться несколько потоков. Если потоков с одним приоритетом, то они упорядочиваются по времени нахождения в очереди. В основном мьютексы используются для управления доступом к ресурсам, которые совместно используются разными потоками управления. Операция захвата мьютекса позволяет обеспечить монопольный доступ к ресурсу.

При исполнении средств синхронизации может случиться так, что менее приоритетные процессы будут мешать выполнению более приоритетных, такой феномен называется - инверсией приоритетов. Мьютексы в отличии от семафоров позволяют избежать такой проблемы.

Пример. Пусть имеются 3 потока: высокого, среднего и низкого. Пусть в данный момент выполняется поток, имеющий средний приоритет, а высокоприоритетный ждет освобождения мьютекса. Во избежание инверсии приоритетов повышается приоритет потока-владельца мьютекса.

## *27. 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.

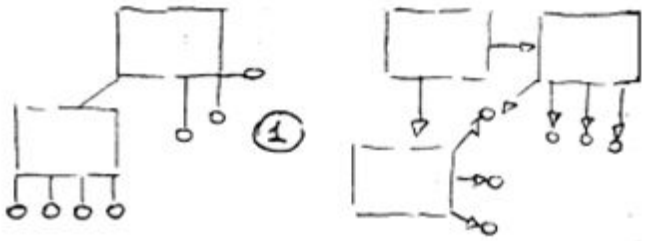
*Категории ресурсов*:

- 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.).

## *28. Логическая организация файловой системы.*

Выделяют две основные модели (обе иерархические):

1. Модель дерево ( MS-DOS ).
2. Модель сеть ( UNIX ).



Тут короче, старые ос - дерево, новые в основном сеть.
Пользователь работает на уровне логических записей.

Логическая запись – это наименьший элемент данных, который доступен пользователю для выполнения различных операций.

## *29. Управление оперативной памятью. Схема механизма физической адресации.*

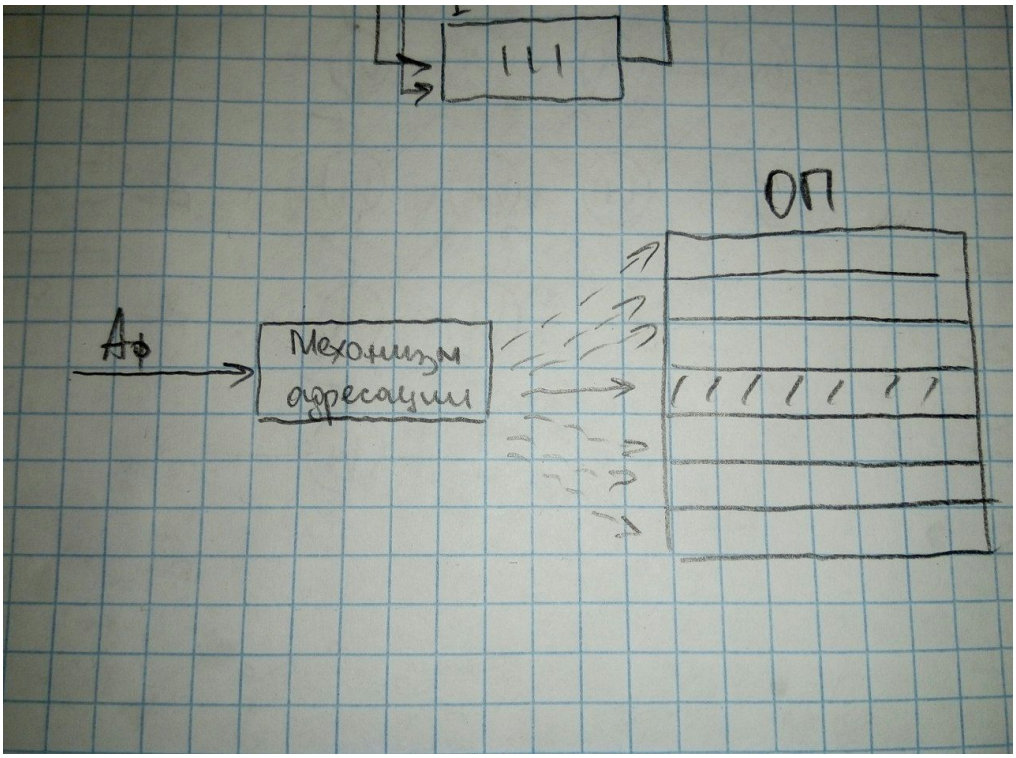



*Схема механизма физической адресации.*

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 объему 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.

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

Windows NT построена в соответствии с классическими принципами:
Виртуальная память имеет страничную организацию.
В общем виде схема описывается следующим образом:

Линейный адрес разбивается на несколько частей:

- Старшая часть адреса содержит номер элемента в корневой таблице.
- Этот элемент содержит адрес таблицы следующего уровня.
- Следующая часть линейного адреса содержит номер элемента уже в этой таблице.
- Самая младшая часть страницы является номером байта в этой физической таблице страниц.

Размер страниц для платформы фирмы Intel составляет 4 Кб, а для платформ DEC Alpha 8 Кб. А схема страничного преобразования выглядит так:

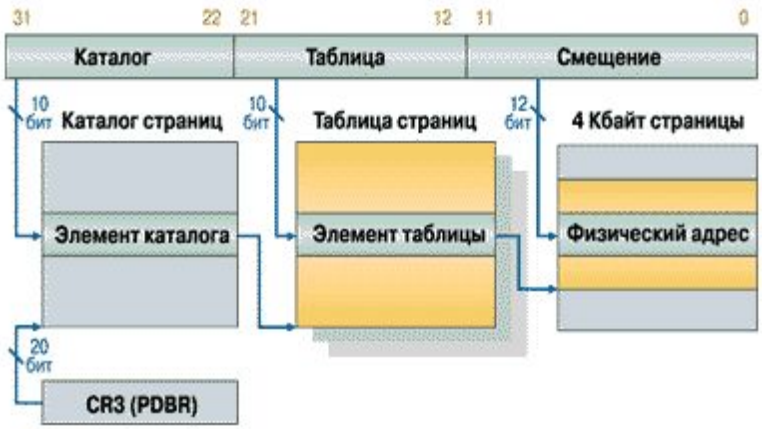



**Рисунок 1.**
Схема страничного преобразования адреса для платформы Intel

Линейный 32-х разрядный адрес занимает три части:

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. Следующие 10 разрядов линейного адреса определяют номер элемента таблицы. Элемент в свою очередь содержит физический адрес страницы ВП. Младших 12 разрядов линейного адреса достаточно, чтобы определить внутри 4КБ страницы точный физический номер, адресуемой ячейки.

Рассмотрим отдельный элемент таблицы страниц, который называют PTE (Page Table Element).


**Рисунок 2.**
Элемент таблицы страниц в Windows NT

Старшие 5 бит определяют тип страницы с точки зрения допустимых операций. WIN 32 API поддерживает 3 допустимых значения этого поля:

1. PAGE NO Access.
2. PAGE READ ONLY.
3. PAGE READ/WRITE.

Следующие 20 бит определяют базовый физический адрес страницы памяти. Если их дополнить 12-ю младшими разрядами(битами) линейного адреса, то они образуют физический

адрес ячейки памяти, к которой производится обращение. Следующие 4 бита (PTE) описывают используемый файл подкачки. Комбинация этих битов определяют один из 16 возможных в системе файлов. Последующие 3 бита определяют состояние страницы в системе.

Старший из них называют T – Transition определяет следующую страницу, как переходную.

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.

P – Present младший бит определяет присутствует страница в ОП или в файле подкачки. Для ускорения страничного преобразования в процессоре имеется специальная кэш-память, которая называется TLB (Translation Lookside Buffer). В ней хранятся наиболее часто используемые элементы каталога и элементы страниц. Каждый процесс в Windows NT имеет свой отдельный каталог страниц и свое собственное независимое адресное пространство, что дает дополнительные возможности при защите процессов.

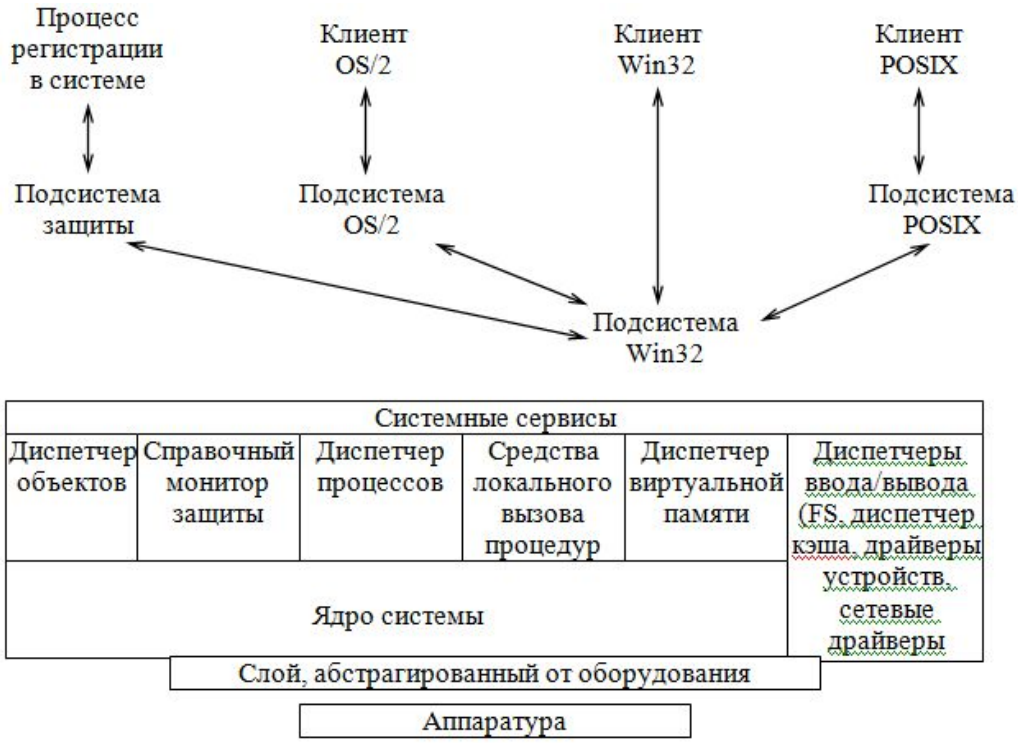
Механизм преобразования виртуального адреса в физический при страничной организации памяти.

Схема преобразования виртуального адреса в физический для сегментно-страничной организации памяти.

## *31. Основные функции операционных систем.*

1. Управление процессами: диспетчеризация, планирование, синхронизация и взаимодействие процессов.
2. Управление ресурсами: организация доступа, управление распределением памяти, процессорного времени с точки зрения пользователя (тип выделять больше ресурсов туда, где пользователю больше всего это нужно. То есть ли он катает в КС-ку, то нужно обеспечить её нормальную работу)
3. Защита и обеспечение безопасности информации
4. Поддержка операционного окружения пользовательских задач

## *32. Структура операционной системы на примере Windows NT. Назначение основных модулей.*




Структура Windows NT состоит из двух частей:

1. **Защищённые подсистемы (серверы)**

**Сервер** – отдельный процесс, память которого защищена от других процессов с помощью системы виртуальной памяти исполнительной системы (ядра). Серверы предоставляют исполнительной системе пользовательский и программный интерфейсы, обеспечивает среду для выполнения приложений различных типов. Сервер подразумевает, что каждая функция подсистемы обеспечивает API-интерфейс (набор процедур, которые вызываются прикладной программой для осуществления низкоуровневых операций, выполняемых ОС). API реализуется на отдельном сервере для Win32, OS/2, POSIX и др. Это позволяет устранить конфликты и дублирования 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. В подсистеме Win32 сохраняется базовая структура 16-тиразрядной Windows, но в NT для повышения производительности были перенесены диспетчер окон, интерфейс и драйверы графических устройств из пользовательского режима в режим ядра. Это позволило избавиться от большого числа сложных участков кода. Подсистема защиты регистрирует правила контроля доступа на локальный компьютер, ведёт базу данных учётных записей пользователя.

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

- **Диспетчер объектов** – создаёт, поддерживает и уничтожает объекты.
- **Справочный монитор защиты** – оберегает ресурсы ОС, обеспечивает защиту объектов и ведёт аудит 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 компьютере.
- **Диспетчер виртуальной памяти** – выделяет и управляет виртуальной памятью и осуществляет подкачку страниц. Каждому процессу предоставляется собственное адресное пространство.

- Ø **Ядро** – реагирует на прерывания, направляет потоки на выполнение, осуществляет межпроцессорную синхронизацию, скрывает различия процессорной и остальной части системы.
- Ø **Диспетчер ввода/вывода** – реализует средства ввода/вывода независимо от типа устройства.
- Ø **Файловая система** – драйверы, принимающие запросы файлового ввода/вывода конкретного устройства.
- Ø **Сетевой редиректор** – драйверы, принимающие запросы ввода/вывода для удалённых файлов, и пересылающие запросы сетевому серверу на другую машину.
- Ø **Драйверы устройств** – низкоуровненвые драйверы, напрямую работающие с оборудованием, в том числе и с сетевым.
- Ø **Диспетчер кэша** – использует средства подкачки страниц диспетчера виртуальной памяти для автоматической записи информациина диск в фоновом режиме (асинхронная запись на диск). Это повышает производительность фалового ввода/вывода.
- Ø **Слой, абстрагированный от оборудования (HAL)** – динамически подключаемая библиотека (DDL). Она изолирует исполнительную систему от особенностей аппаратных платформ разных производителей.

**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.

## *33. 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:

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

**По другой классификации файлы бывают:**

1-Обычные (текстовые, двоичные).

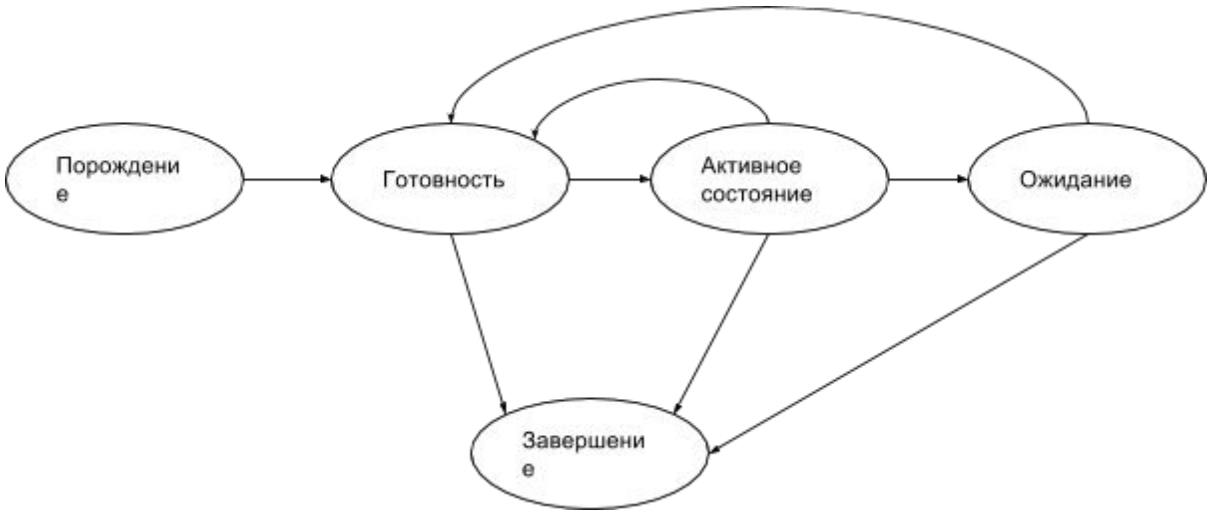
2-Специальные (например, для операции ввода-вывода. Блоко-ориентированные, байто-ориентированные).

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.

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




***Порождение -*** подготавливаются все условия для исполнения (выделяются ресурсы и тд.)

***Готовность -*** все ресурсы предоставлены, кроме процессора. Здесь процесс ждет, когда диспетчер, его выберет.

***Активное состояние -*** процесс обладает всеми ресурсами и непосредственно использует процессор.

***Завершение -*** процесс завершается (нормально или аварийно).

***Ожидание -*** процесс может быть прерван / приостановлен по ряду причин:

1) 1-ый род прерывания (когда он сам виноват). При попытке получить ресурс или отказаться от него; при порождении или уничтожении или других действий по отношению к другим процессам; арифметическое переполнение; обращение к защищенным участкам памяти, которые не относятся к этому процессу (чужие).
2) 2-ой род прерывания (когда он не виноват). Процесс может быть вытеснен более приоритетным процессом или по окончании кванта времени; другие прерывания этого вида обусловлены с проведением синхронизации с взаимодействующими процессами.

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

Внешняя память разбивается на блоки фиксированного размера. Каждый блок имеет свой уникальный порядковый номер. Каждый блок - минимальная единица данных, которые участвуют в обмене между устройством внешней и оперативной памяти.

**Способы размещения.**

1) Непрерывное размещение
   Файл состоит из последовательности блоков


   Недостатки: трудно расширить, неэффективное использование дискового пространства.

2) Связный список блоков

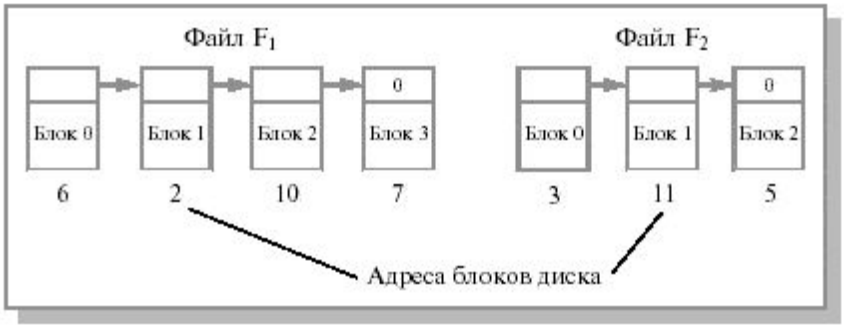



**Рис. 12.2.** Хранение файла в виде связного списка дисковых блоков

Адрес определяется одним числом - номером 1го блока. Каждый блок содержит указатель на следующий блок. Иногда последний блок содержит спец. знак конца файла (EOF).

Достоинство: нет необходимости декларировать размер файла в момент создания, он может расти неограниченно.

Недостатки: наличие дефектного блока приводит к потере информации в оставшейся части файла и потенциально к потере дискового пространства; при прямом доступе к файлу для поиска i-го блока нужно осуществить несколько обращений к диску, последовательно считывая блоки от 1 до i-1, то есть выборка логически смежных записей, которые занимают физически несмежные секторы, может требовать много времени. Здесь мы теряем все преимущества прямого доступа к файлу. Для указателя на следующий блок внутри блока нужно выделить место, что не всегда удобно.

3) Связный список индексов.

Хранятся в этом способе указателя не в дисковых блоках, а в индексной таблице в памяти, которая называется таблицей отображения файлов (FAT - file allocation table).По-прежнему существенно, что запись в директории содержит только ссылку на первый блок. Далее при помощи таблицы FAT можно локализовать блоки файла независимо от его размера. В тех строках таблицы, которые соответствуют последним блокам файлов, обычно записывается некоторое граничное значение, например EOF.

Главное достоинство данного подхода состоит в том, что по таблице отображения можно судить о физическом соседстве блоков, располагающихся на диске, и при выделении нового блока можно легко найти свободный блок диска, находящийся поблизости от других блоков данного файла. Минусом данной схемы может быть необходимость хранения в памяти этой довольно большой таблицы.

| Номера блоков диска | | |
|---|---|---|
| 1 | | |
| 2 | 10 | |
| 3 | 11 | Начало файла $F_2$ |
| 4 | | |
| 5 | EOF | |
| 6 | 2 | Начало файла $F_1$ |
| 7 | EOF | |
| 8 | | |
| 9 | | |
| 10 | 7 | |
| 11 | 5 | |

**Рис. 12.3.** Метод связного списка с использованием таблицы в оперативной памяти

4) Перечень номеров блоков

- номера блоков, занимаемых файлами просто перечисляются

Недостатки :

- усложняется алгоритм распределения
- время поиска увеличено

## *36. Динамические тома в Windows NTFS5.*

*Базовые*, или *основные*, *диски* — это термин, обозначающий дисковые конфигурации, использовавшиеся в системах корпорации Microsoft до появления Windows 2000. После выхода Windows данные технологии прибрели название «базовый (основной)» диск (*basic disk*) для того, чтобы отличить их от новых технологий управления дисками, которые стали называть «*динамическими*» (*dynamic disks*).

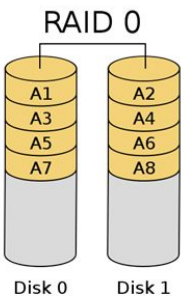
Динамические диски — это технология управления жесткими дисками, позволяющая создавать на базе обычных жестких дисков компьютера более производительные или отказоустойчивые конфигурации.

Технологии создания производительных или отказоустойчивых конфигураций дисков имеют общее название *RAID*.

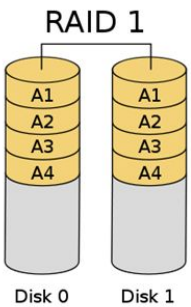
**RAID (англ. redundant array of independent disks) — избыточный массив независимых жёстких дисков**

**Чтоб понять последующий материал, лучше прочитать оригинальную статью на Википедии про RAID.**

Массив из нескольких дисков, управляемых контроллером, взаимосвязанных скоростными каналами и воспринимаемых внешней системой как единое целое. В зависимости от типа используемого массива может обеспечивать различные степени отказоустойчивости и быстродействия. Служит для повышения надёжности хранения данных и/или для повышения скорости чтения/записи информации

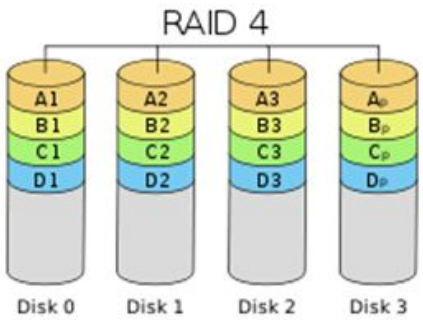



Данные пишутся по очереди на оба диска.
Плюсы: скорость работы
Минусы: нет отказоустойчивости.

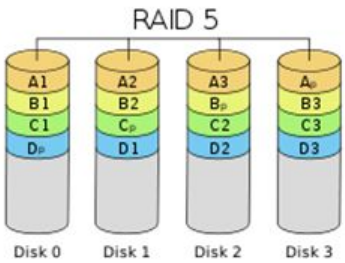



Данные зеркалируются.

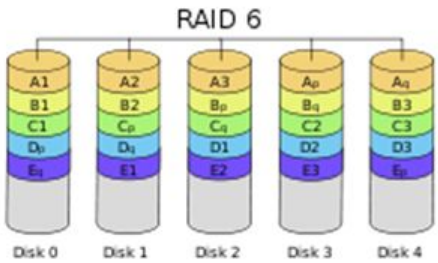
Плюсы: отказоустойчивость
Минусы: скорость работы




Запись происходит поблочно по очереди на 3 диска. Плюс один диск выделен для контрольной суммы. В случае выхода одного из дисков просядет производительность, но данные не потеряются.



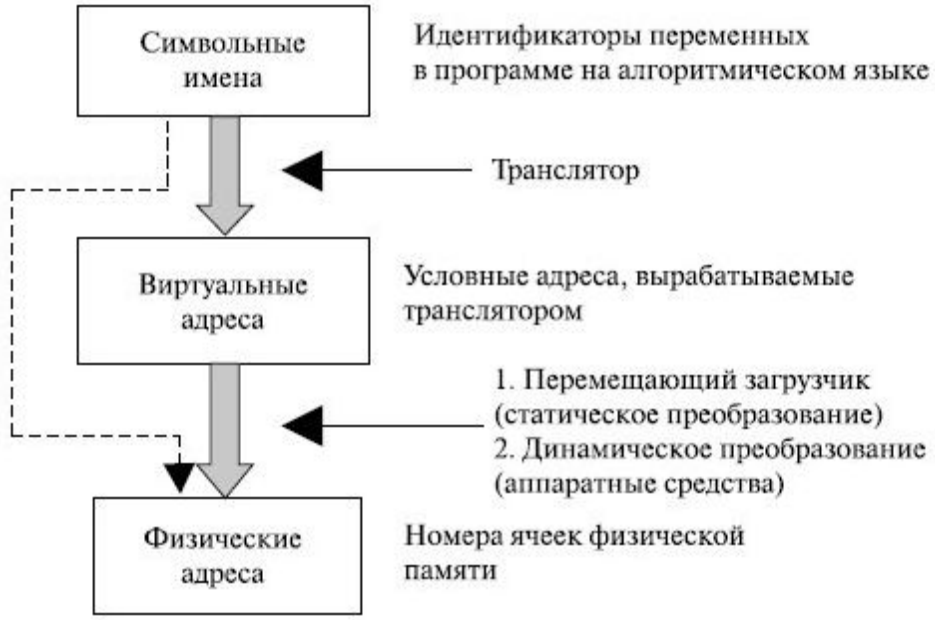

Тоже самое что и Raid 4. Однако контрольная сумма записывается на разные диски. Тем самым уменьшается износ дисков




Тоже самое что и Raid5
Но используются для чётности 2 диска. Отказоустойчивость, соответственно, 2 диска.
Минусы:
- Большее время восстановления
- Медленнее скорость работы

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

По его лекциям. Виды памяти - физическая и логическая(набор ссылок, которые может использовать программа)
Для идентификации переменных используют символьные имена (метки), виртуальные адреса и физические адреса.




**Основные функции ОС про управлению оперативной памятью:**

1) Отображение и перевод логической памяти в физические адреса.

Выделяют следующие функции перевода:

а) абсолютные трансляции - выполняются компилятором или ассемблером при подготовке программы и генерировании ими абсолютных адресов

б) статические трансляция - проводится, если программа составлена в форме выполняемого модуля: отдельные программы собираются вместе и им присваиваются адреса относительно некоторого фиксированного адреса памяти.

в) динамическая трансляция - когда реальные адреса расположения программы в памяти определяются ОС-й.

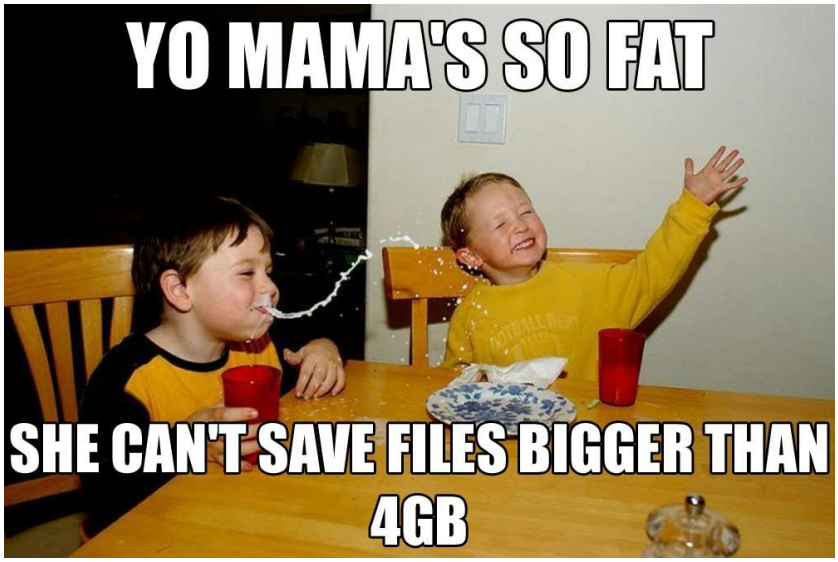
2) Расширение границы логического адресного пространства за границы физического адресного пространства.

Для этого используют оверлейное программирование (разные части программы используют одинаковый набор логических адресов.

Существуют другие методы выхода за границы - свопинг (обмен) и сцепление. При этих методах логическое адресное пространство любой программы соответствует физическому адресному пространству.

3) Разделение между программой пользователя и ОС-ой и распределение между ними.
4) Защита информации (пользователя от ОС и наоборот, друг от друга).

## *38. Сравнительная характеристика FAT16, FAT32, NTFS4,NTFS5.*




FAT16 работала с файлами размером до 2 Гб. Таблица размещается в начале тома, с целью защиты имеются 2 копии FAT. Таблица и корневой каталог (это короче скрытая папка fat на носителе информации) должны располагаться по строго фиксированным адресам, чтобы файлы, необходимые для запуска системы, были размещены корректно. Таблица используется для определения кластеров нужного файла. В FAT16 корневой каталог имеет фиксированный размер.

==========

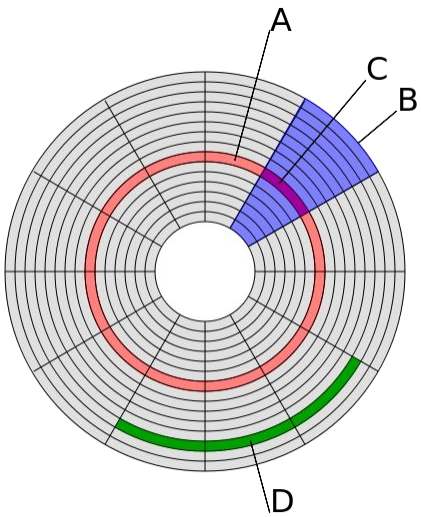
**Небольшое отступление для нубов (таких каким был я).**

Короче Пугач может спросить про такие вещи, как сектор и кластер. Разберемся.

Все знают, что жесткий диск состоит из дорожек. Каждая дорожка пилится на одинаковые куски (обычно 512 байт), каждый из которых называется ***сектор.*** Более крутыми и выпендрежными словами: Сектор - это минимальная пронумерованная область ДИСКА, в которой могут храниться данные.

Идем далее. Чтобы ОС нормально работала на нашем винчестере создается файловая система. Вот она как раз и использует сектора, **НО** по различным причинам файловые системы объединяют сектора в группы, и вот такие группы и называются ***кластерами.*** То есть иными словами: Кластер - это минимальная область ФАЙЛОВОЙ СИСТЕМЫ, которая предназначена для хранения информации и он состоит как из одного, так и нескольких секторов.

Картинка, которая внизу все разъяснит графически. Красным выделена дорожка(A), B - геометрический сектор диска, C - сектор дорожки, а вот кластер выделен зеленым цветом (D). В данном случае размер кластера - 3 сектора, т.е. 3*512 байт = 1536 байт



==========

Каталоги - специальные файлы с 32х битными элементами для каждого файла,содержащегося в каталоге, где хранятся имена и атрибуты.
Для FAT32 максимальный размер файла 4 ГБ, поддерживает диски до 2 ТБ. Преимущества:

- *Поддержка дисков размером до 2 Тбайт.* Следует, правда, отметить, что команда format, включенная в Windows 2000, не позволяет *форматировать* для использования FAT32 тома, размер которых превышает 32 Гбайт. Поэтому при форматировании томов объемом более 32 Гбайт следует использовать файловую систему NTFS. Однако драйвер FASTFAT, имеющийся в составе Windows 2000, позволяет монтировать и поддерживать любые тома FAT32, в том числе и такие, объем которых превышает 32 Гбайт. За исключением упомянутого выше ограничения FAT32 в Windows 2000 работает точно так же, как в Windows 95 OSR2 и Windows 98.
- *Более эффективное расходование дискового пространства.* FAT32 использует более мелкие кластеры (см. табл. 7.1), что позволяет повысить эффективность использования дискового пространства на 10≈15% по сравнению с FAT.
- *Повышенная надежность и более быстрая загрузка программ.* В отличие от FAT 12 и FAT 16, FAT32 обладает возможностью перемещать корневой каталог и использовать резервную копию FAT, если первая копия получила повреждения. Кроме того, загрузочный сектор FAT32 был расширен по сравнению с FAT16 и содержит резервные копии жизненно важных структур данных. Повышенная устойчивость FAT32 обусловлена именно этими факторами.

NTFS размер диска 2^64 байт = 16 эксабайт. Размер файла ~16 эксабайт. (Расписывать про NTFS можно дохера, почитайте доп инфу на википедии, там норм).

Отличие NTFS 4,5 от предыдущих версий:

1) Система безопасности позволяет устанавливать различные права доступа к файлам и папкам для пользователей и групп.
2) Быстрое восстановление томов в случае сбоя за счёт избыточности данных.
3) Гибкие опции форматирования позволяют более эффективно использовать тома дисков.
4) Тома могут расширяться
5) Есть зеркальные тома на сервере (наверн, удаленные резервные копии)

В Windows NTFS5 имеет дополнительные преимущества:

1. Можно задать квоты на использование свободной памяти на диске для определенного пользователя.

2. Размер раздела диска можно увеличить без перезагрузки ОС.

3. Пользователь может передать право владения папкой или файлом другому пользователю (в NTFS-4 этого не было).

4. Точки соединения NTFS (точки перехода)

5. Возможность шифрования файлов. Это исключает доступ к файлу средствами DOS, Unix и т. д.

## *39. Управление оперативной памятью. Использование оверлеев.*

Оверлейное программирование позволяет создавать программы, размер которых больше, чем размер доступной памяти.
Программа разбивается на главный (**корневой**) сегмент и на один или более неосновных сегментов фиксированной длины (**оверлеи**).

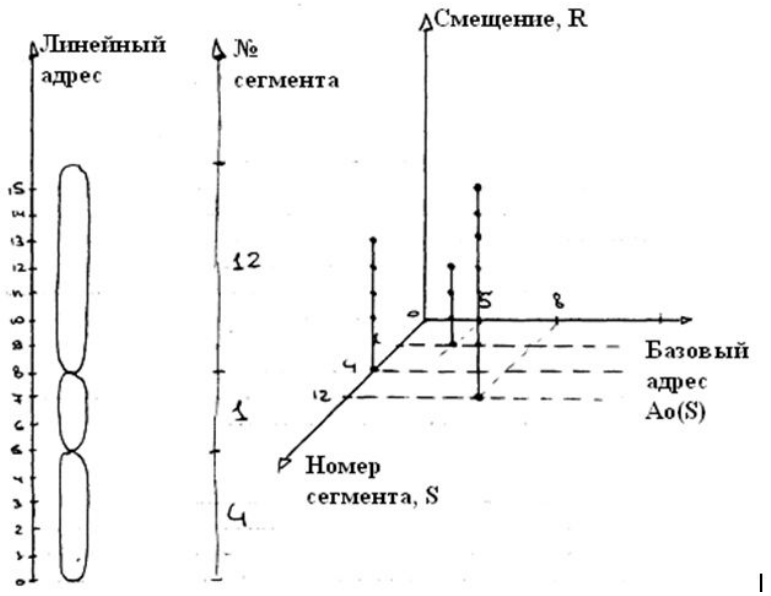
Корневой сегмент резидентно (то есть на протяжении работы всей программы, не выгружаясь обратно) находится в памяти, а оверлеи по мере необходимости подгружаются в память с диска. Область памяти программы пользователя, которая зарезервирована под оверлеи называется оверлейной группой (регионом).

По принципу организации и способу загрузки оверлейные системы делятся на:

1) с автоматической организацией: деление программы на оверлеи производит ОС.
2) с полуавтоматической: пользователь определяет, как должна быть разделена программа, а ОС определяет как загружать оверлеи.
3) с программируемой(?): пользователь не только определяет, как разделять программу, но и определяет опрос на загрузку оверлея.

Программа может быть распределена на оверлеи в виде дерева, где 0-й уровень это корневой уровень. Оверлей вызываемый из 0го уровня - это первый уровень и тд.

## *40. 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. Сегменту с номером S присваивают базовый адрес As0, а затем принцип база+смещение A=As0+R.

**Механизм работы такой:**

Отдельный сегмент может представлять собой подпрограмму, массив данных и др. То есть 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** указывается**:**

**1-Начальный физический адрес.**

**2-Размер сегмента.**

**3-Правила доступа.**

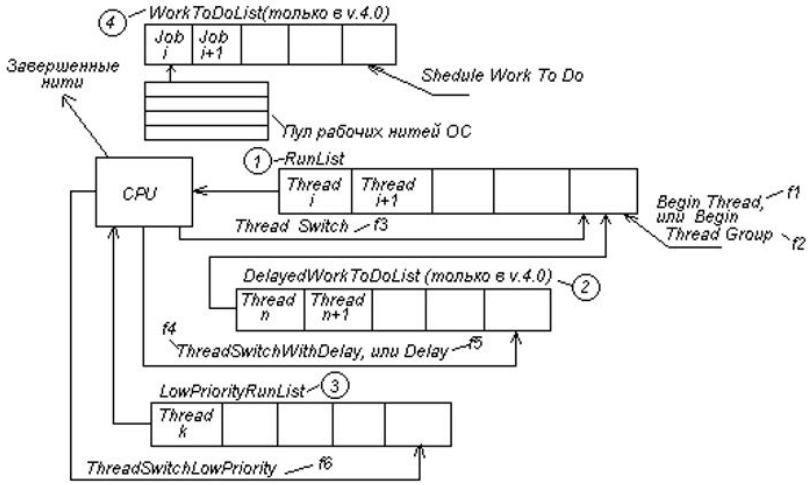
**4-Признак модификации.**

**5-Признак обращения к данному сегменту за последний квант времени и другая информация.**

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. На уровне сегментов более медленное преобразование адреса.

## *41. Система очередей планирования NetWare.*




При создании нити с помощью функции f1 и f2 нить попадает в конец очереди (1) Runlist (содержит готовые к выполнению нити)
Планировщик выбирает первую нить из очереди (1) и запускает на выполнение (приусловии, что (4) пуста).
Если нити требуются еще итерации(перед завершением), то она помещается в конец одной из очередей в зависимости от того, какой вызов управления был использован.
В конец очереди (2) - f4 или f5, (3) - f6…
Нити находятся в очереди (2), после завершения попадают в очередь (1).
Нить из очереди (3) запускается, если очередь (1) пустая. Обычно в очередь (3) попадают нити выполняющие не срочную фоновую работу.
Очередь (4) - самая приоритетная.
Планировщик разрешает выполнять несколько нитей из очереди (4), а затем запускает очередь (1).
Очередь (4) появлялась в версии 4 и это сделано для повышения производительности NLM - приложений (Novell Load Module)

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

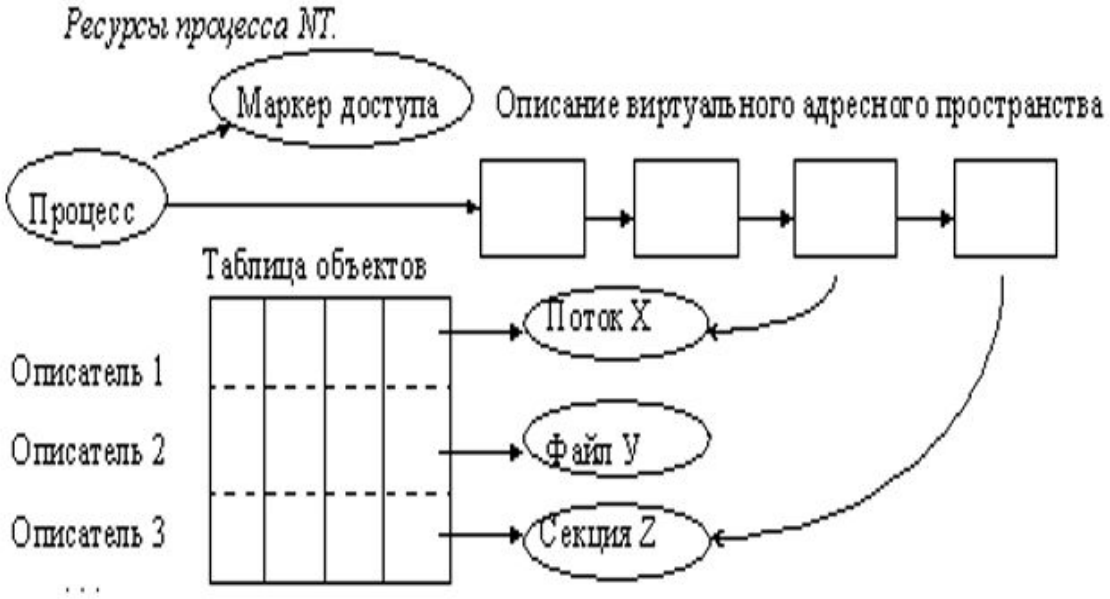
Процесс-один из типов объектов, который связан с другими объектами (с потоком, портом передачи сообщений между процессами, маркерами доступа, событиями, семафорами и др)

Поток-сущность внутри процесса, секция совместно используемой памяти.

Процесс состоит из :

1) исполнительной программы, которая содержит начальный код и данные
2) закрытого адресного пространства (набора адресов виртуальной памяти)
3) системных ресурсов: семафоры, коммуникационных портов и файлов.

Большинство процессов - это процессы пользовательского режима (высшего уровня), а в режиме ядра выполняется 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 ресурса видно, что процесс открыл описателя одного из своих потоков (x), файла Y и секции Z совместно используемой памяти. Стрелки от адресного пространства означают виртуальные адреса занятые стеком потока и объектом секциями.

*Процесс (объект) имеет следующие атрибуты и сервисы*
*Атрибуты (поля объекта)*
1) Идентификатор процесса
2) Маркер доступа
3) Базовый приоритет
4) Процессорное средство по умолчанию
5) Размеры квот на ресурсы памяти
6) Время выполнения
7) Счетчик ввода/вывода
8) Счетчик операций виртуальной памяти
9) Порты исключений-отладки
10) Коды завершений

*Сервисы (методы объекта)*
1) Создать, открыть процесс, запросить информацию о процессе, установить информацию
2) выяснить текущий процесс
3) Завершить процесс
4) Освободить виртуальную память
5) Защитить виртуальную память
6) Чтение, запись в виртуальную память
7) Блокировать, разблокировать виртуальную память
8) Опросить виртуальную память
9) Сбросить виртуальную память

## *43. Основные понятия: Операционная система. Процесс. Поток. Многозадачность. Многопоточность.*

**Операционная система** - организованная совокупность программ, предоставляющая пользователям набор средств, позволяющих упростить программирование, отладку и

управление программами. Представляет собой интерфейс между пользователем и аппаратной частью. Также предназначена для управления ресурсами компьютера и организации взаимодействия с пользователем.

**Процесс** - совокупность последовательных действий, необходимых для достижения какого-либо результата в вычислительной системе.

В современной ОС процесс реализуется как динамический объект, который имеет множество атрибутов и сервисов. Он может находиться в различных состояниях, выполнять действия по отношению в другим процессам - объектам, занимать или освобождать необходимые ему ресурсы. Процесс - логическая единица работы ОС.

**Поток(нить)** - сущность внутри процесса, отображающая одну из возможно многих подзадач процесса.

**Мультипрограммирование(многозадачность)** - способ организации вычислительного процесса, при котором различные потоки совместно используют процессор.

Кооперативная многозадачность - потоками совместно используются не только процессор, но и другие ресурсы (Оперативная память, на пример)

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

(Бонусы из этой же лекции:

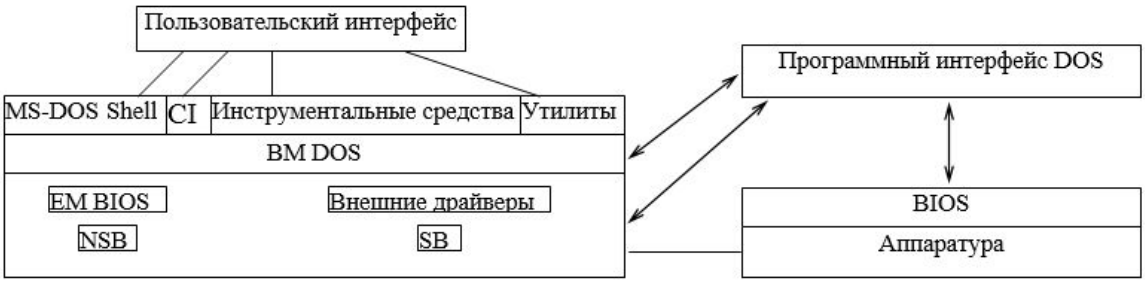
Мультипроцессорная обработка - исполнение одного и того же кода ОС различными процессами

Вытесняющая дисциплина обслуживания - решение о переключении процессора с одного потока на другой принимает ОС

Невытесняющая - поток по собственной инициативе передает управление ОС

Ресурсы - различные средства и возможности, необходимые для организации и поддержки процесса. Могут быть физические и вирутальные, т.е. мнимые (виртуальный ресурс не существует в том виде, в котором он проявляется пользователю. Его построение происходит на базе физического и это некая его модель)

## *44. Структура операционной системы на примере MS-DOS. Назначение основных модулей.*




*MS-DOS* (англ. *Microsoft Disk Operating System*) — дисковая операционная система для компьютеров на базе архитектуры x86.

*BIOS* (Basic Input Output System) - базовая система ввода и вывода.

Функции BIOS:

- автоматическое тестирование основных аппаратных компонентов при включении компьютера, включая ОП и т.д.
- загрузка блока начальной загрузки из ПЗУ в ОП
- обслуживание системных вызовов. Для этого используется механизм прерываний, то есть работа машины приостанавливается одним из сигналов с целью срочной обработки. Каждое прерывание имеет свой уникальный номер и с ним быть связана определенная подпрограмма. Прерывания можно разбить на три группы: аппаратные, логические или процессорные и программные.

В итоге BIOS содержит драйверы стандартных периферийных устройств, тестовые программы аппаратуры и программы начала загрузки.

Таким образом BIOS осуществляет инициализацию векторов прерывания нижнего уровня и считывает в пямять NSB (Non-System Bootstrap - внесистемный загрузчик).

*NSB* - стартовый сектор физического жёсткого диска. Он считается вторичным загрузчиком (после BIOS). NSB считывает в память и запускает SB.

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

*EM BIOS* (Extension Module BIOS) - файл IO.SYS это такой системный файл MS-DOS, осуществляющий определение состояния оборудования, конфигурирование DOS по указаниям в файле CONFIG.SYS, а также инициализацию и переустановку некоторых векторов прерываний нижнего уровня, запускает BM DOS(Basic Module DOS).

*BM DOS (Basic Module)* - реализует основные функции ОС. Его основу составляют обработчики прерываний верхнего уровня. BM DOS также осуществляет считывание в память интерпретатора команд или командный файл CI (COMMAND.COM) и запускает его. Этот файл отвечает за пользовательский интерфейс.

*CI* состоит из резидентного модуля (резидентный - постоянно хранящийся в памяти, невыгружаемый), который включает в себя обработчиков прерываний и код подгрузки транзитного модуля и транзитный модуль, который отвечает за подгрузку и выполнение команд.

*Внешние драйвера* - отдельные файлы для работы с внешними периферийными устройствами.

*Утилиты* - обслуживающие программы, предоставляющая пользователю сервисные услуги (FORMAT.COM: CHKDISK.COM, FDISK.COM)

## *45. Файловая система. Задача файловой системы. Функции файловой системы.*

Файловая система - часть операционной системы, которая обеспечивает выполнение операций над файлами.

Основные задачи файловой системы:

1. Обеспечение независимости программы от конкретной конфигурации машины.
2. Сокрытие от пользователей реального расположения информации на физическом уровне

Функции файловой системы:

1. Реализация методов организации данных.
2. Перевод логических характеристик в физические
3. Защита пользователя от случайных сбоев
4. Возможность многим пользователям разделять одно устройство
5. Восстановление файлов
6. Защита от несанкционированного доступа.

## *46. Классические дисциплины обслуживания очереди на исполнение процесса.*

Есть два классических подхода:

1. FIFO (First In First Out) - дисциплина обслуживания в порядке поступления процессов в очередь на обслуживание. Она обеспечивает минимальную дисперсию времени ожидания. Очередь короч

2. LIFO (Last In First Out) - дисциплина обслуживания, обратная порядку поступления. Является основной для построения стековой памяти. Стэк короч

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.

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

Цель алгоритма распределения - минимизация размера и кол-во пустых участков памяти.
При распределении памяти ставят 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.

Алгоритмы.
1ый вариант: 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.
2ой 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 в этих ал-х – нельзя выделить зону, размер которой больше размера ОП.
Простое непрерывное распределение — это самая простая схема, согласно которой вся память условно может быть разделена на три области:

- область, занимаемая операционной системой;
- область, в которой размещается исполняемая задача;
- незанятая ничем (свободная) область памяти.

Изначально являясь самой первой схемой, схема простого непрерывного распределения памяти продолжает и сегодня быть достаточно распространенной. Эта схема предполагает, что операционная система не поддерживает мультипрограммирование, поэтому не возникает проблемы распределения памяти между несколькими задачами. Программные модули, необходимые для всех программ, располагаются в области самой операционной системы, а вся оставшаяся память может быть предоставлена задаче. Эта область памяти получается непрерывной, что облегчает работу системы программирования. Поскольку в различных однотипных вычислительных комплексах может быть разный состав внешних устройств (и, соответственно, они содержат различное количество драйверов), для системных нужд могут быть отведены отличающиеся объемы оперативной памяти, и получается, что можно не привязывать жестко виртуальные адреса программы к физическому адресному пространству. Эта привязка осуществляется на этапе загрузки задачи в память.

Для того чтобы для задач отвести как можно больший объем памяти, операционная система строится таким образом, чтобы постоянно в оперативной памяти располагалась только самая нужная ее часть. Эту часть операционной системы стали называть *ядром*. Прежде всего, в ядро операционной системы входят основные модули супервизора. Для однопрограммных систем понятие супервизора вырождается в модули, получающие и выполняющие первичную обработку запросов от обрабатывающих и прикладных программ, и в модули подсистемы памяти. Ведь если программа по ходу своего выполнения запрашивает некоторое множество ячеек памяти, то подсистема памяти должна их выделить (если они есть), а после освобождения этой памяти подсистема памяти должна выполнить действия, связанные с возвратом памяти в систему. Остальные модули операционной системы, не относящиеся к ее ядру, могут быть обычными диск-резидентными (или транзитными), то есть загружаться в оперативную память только по необходимости, и после своего выполнения вновь освобождать память.

Такая схема распределения влечет за собой два вида потерь вычислительных ресурсов — потеря процессорного времени, потому что процессор простаивает, пока задача ожидает завершения операций ввода-вывода, и потеря самой оперативной памяти, потому что далеко не каждая программа использует всю память, а режим работы в этом случае однопрограммный. Однако это очень недорогая реализация, которая позволяет отказаться от многих функций операционной системы. В частности, такая сложная проблема, как защита памяти, здесь почти не стоит. Единственное, что желательно защищать — это программные модули и области памяти самой операционной системы.

## *48. Алгоритм адаптивно-рефлективного планирования процессов.*

- контроль над реальным использованием памяти
- к началу планирования для каждого процесса устанавливаются ограничения на использование памяти и виртуального времени процессора.
- ОС следит за рабочей областью каждого процесса в течение всего времени выполнения
- ограничения на память определяются оценкой текущего объема памяти и оценкой времени изменения этого объема ( на основе анализа работы процесса предыдущего кванта времени)
- если памяти достаточно-выделяется временной интервал, но его величина обратно пропорциональна объему памяти, которую заполнил процесс.

Цель подхода – ориентировать систему на процессы с минимальными рабочими областями, то есть дается преимущество небольшим программам с малым временем выполнения.

## *49. Планирование процессов. Планировщик. Двухуровневая система управления процессами. Типы планировщиков.*

*Планирование* - управление распределением ресурсов вычислительной системы между различными пользователями или процессами путём передачи им управления согласно определенной стратегии планирования.

*Диспетчер* отвечает за выбор процесса и передачи ему управления (это называется диспетчеризацией).

При построении системы управления процессами в большинстве современных ОС используется *двухуровневая схема*.

1-ый уровень называется *долгосрочное планирование,* то есть это верхний уровень (планировщик заданий). На верхний уровень выносятся действия редкие в системе, но требующие больших системных затрат. Здесь процесс рассматривается как совокупность состояний по использованию программы на виртуальной машине и для этого уровня состояние порождения означает, что планировщик создает требуемую ВМ. Здесь источник требований на порождение работы является внешним относительно системы, то есть приходит извне (для нижнего уровня наоборот).

*Готовность* для верхнего уровня означает. что для исполнения программы предоставлены все ресурсы ВМ, кроме виртуального процессора.

*Окончание* - освобождение всех ресурсов, которые были затрачены на создание ВМ.

*Краткосрочное планирование (супервизор задач, 2-ой, нижний уровень)*

Здесь моделируется на процессоре деятельность виртуального процессора, а активное состояние определяется исполнением работы на виртуальном процессоре. Заявка на порождение работы такого процессора формируется на верхнем уровне. Доступ любого задания к процессору осуществляется через системные программы планировщика-диспетчера.

*Виды планировщиков или стратегии построения планирования*

Существует 2 стратегии построения планировщика.

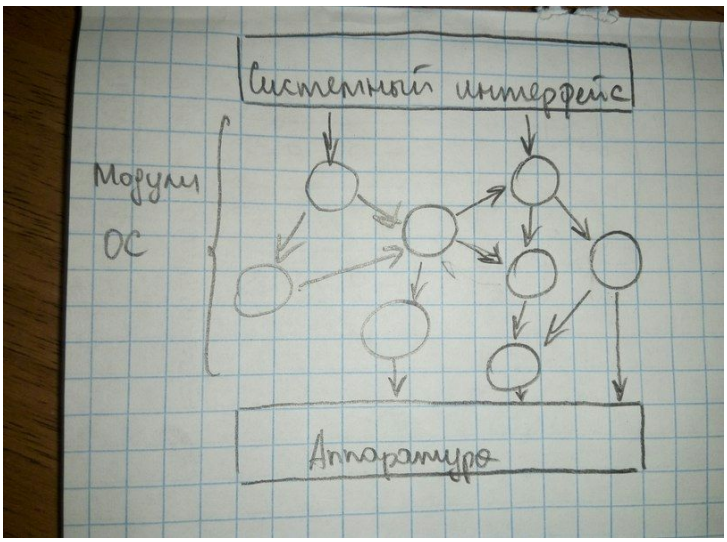
1. *Единый главный планировщик* используется для всех заданий, он встроен в ядро ОС, осуществляет общее системное разделение времени процессора.
2. *Разделенный планировщик*. Здесь планировочный модуль помещается в адресную часть каждой программы пользователя, то есть каждый решает сам чё ему делать (нет единого главного планировщика). Например, процесс может осуществить программный вызов для постановки себя в очередь на исполнение, то есть любая программа может использовать собственную стратегию планирования.

## *50. Структуры монолитной, структурированной, микроядерной ОС и их особенности.*

1. *Монолитные ОС*

Такая система представляет собой наборы процедур, каждая из которых может вызывать другую.

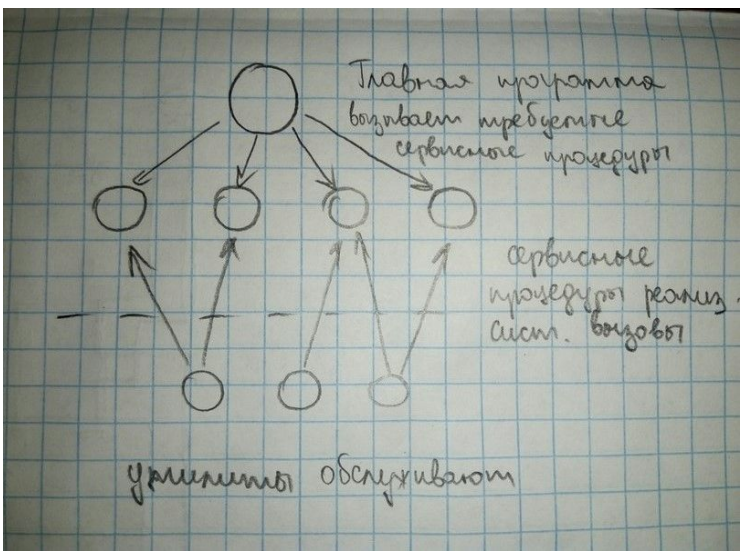
Для построения монолитной системы необходимо выполнить компиляцию всех необходимых процедур и связать в единый объектный файл с помощью компоновщика.




2. *Структурированные монолитные ОС*

При обращении к системным вызовам параметры помещаются в строго отведенные места, например, в регистр или стек, а затем выполняют команду прерывания, известную как вызов ядра (супервизора), которая переключает машину из режима пользователя в режим ядра, затем проверяются параметры вызовов, фиксируется таблица ссылок выхода и вызывается соответствующая процедура.

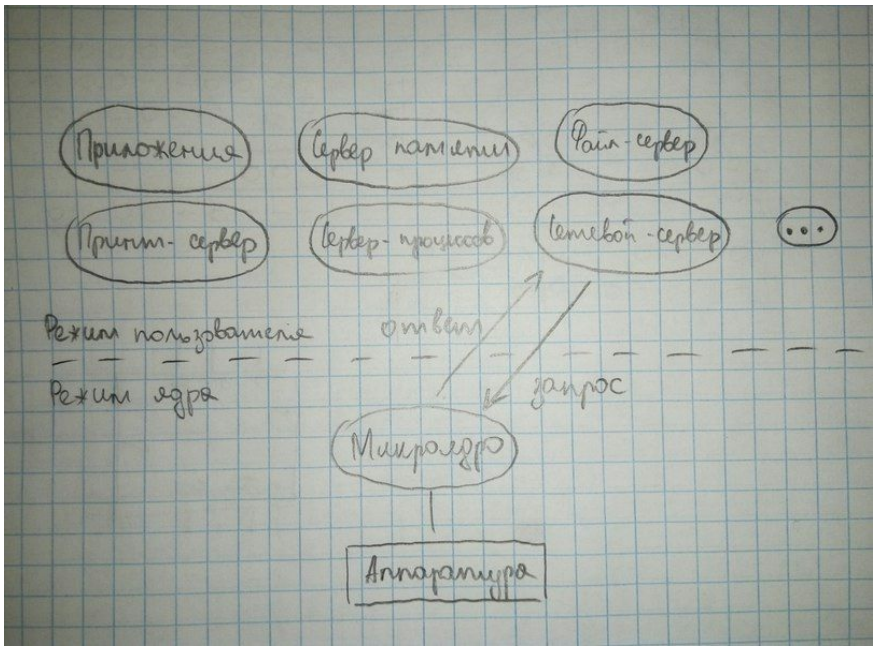
В этой модели для каждого системного вызова имеется одна сервисная процедура.



3. *Микроядерные ОС*

Здесь компоненты используют средства микроядра для обмена сообщениями, а ядро проверяет законность сообщений, пересылает их между компонентами и обеспечивает доступ к аппаратуре. Такой подход позволяет микроядерную ОС использовать в распределенных средах. Объясняется это тем, что микроядру всё равно, откуда пришло сообщение.

Однако пересылка сообщений производится медленнее обычных вызовов функций. Соответственно, важной задачей становится оптимизация пересылок сообщений.




## *51. Схема взаимодействия файловой системы. Уровни файловой системы.*


<u>На уровне идентификации</u>

1.По символьному имени файла определяется его уникальное имя ( в тех ОС, где один файл может иметь несколько символьных имен)

2.По уникальному имени определяют атрибуты файла

3.Сравниваются полномочия пользователя или процесса с правилами доступа файла.

<u>Логический уровень</u>

1.Определяются координаты логической записи в файле

2.Алгоритм работы зависит от конкретной логической модели

Физический уровень

1.Определяется номер физического блока, который содержит требуемую логическую запись

## *52. Планирование процессов. Алгоритм лотерейного планирования.*

Планирование - управление распределением ресурсов вычислительной системы между различными пользователями или процессами путём передачи им управления согласно определенной стратегии планирования.

Процессам раздаются “лотерейные билеты” на доступ к ресурсам и процессорам. Планировщик случайным образом выбирает билет и его обладатель получает ресурсы. Наиболее важным процессам выдают дополнительные билеты. В итоге любой процесс получает некоторый процент ресурсов примерно равный проценту его “билета”.
Взаимодействующие процессы могут обмениваться билетами.