



Лекция 3.

Файловая система ОС UNIX

(основные принципы).



Термин “файловая система” - ФС

В литературе термин **ФС** используется для обозначения трех разных понятий:

- Во-первых, **файловая система - это набор правил и конструкций**, описывающих то, как сохраняются файлы на диске. В этом смысле мы употребляем, например, выражение "файловая система FAT32", и "файловая система" здесь тождественна понятию "тип файловой системы".
- Во-вторых, **файловая система - это совокупность всех файлов**, хранимых в компьютере.
- В-третьих (и это значение термина характерно именно **для UNIX-систем**) **файловая система - это совокупность всех файлов на разделе диска или устройстве**.

Термин ФАЙЛ был определён в лекции 2 – в базовых понятиях ОС Unix:

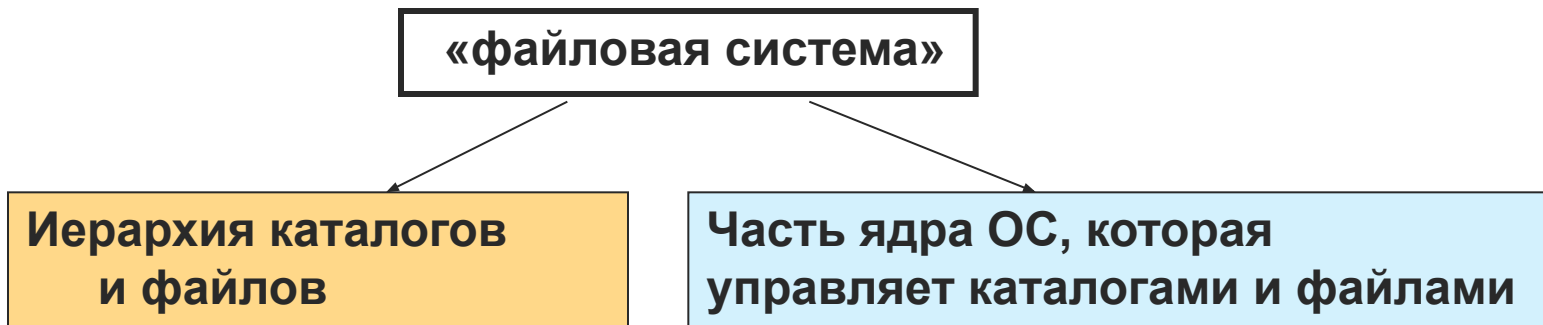
1. **Пользователь** – зарегистрированное в среде UNIX лицо, которому после надлежащей проверки разрешается работать в системе.
 2. **Терминал (~ интерфейс)** – основной инструмент пользователя для работы с системой в интерактивном режиме.
 3. **Процесс** – акт выполнения заранее подготовленной программы (задачи) в отдельном адресном пространстве.
-
4. **Файл** – в ОС UNIX – это универсальная абстракция, означающая структурированную, именованную область внешней памяти и / или последовательность байт, служащую для определения и обращения к физическим устройствам компьютера, либо для связи процессов.



Термин - «файловая система» ОС UNIX

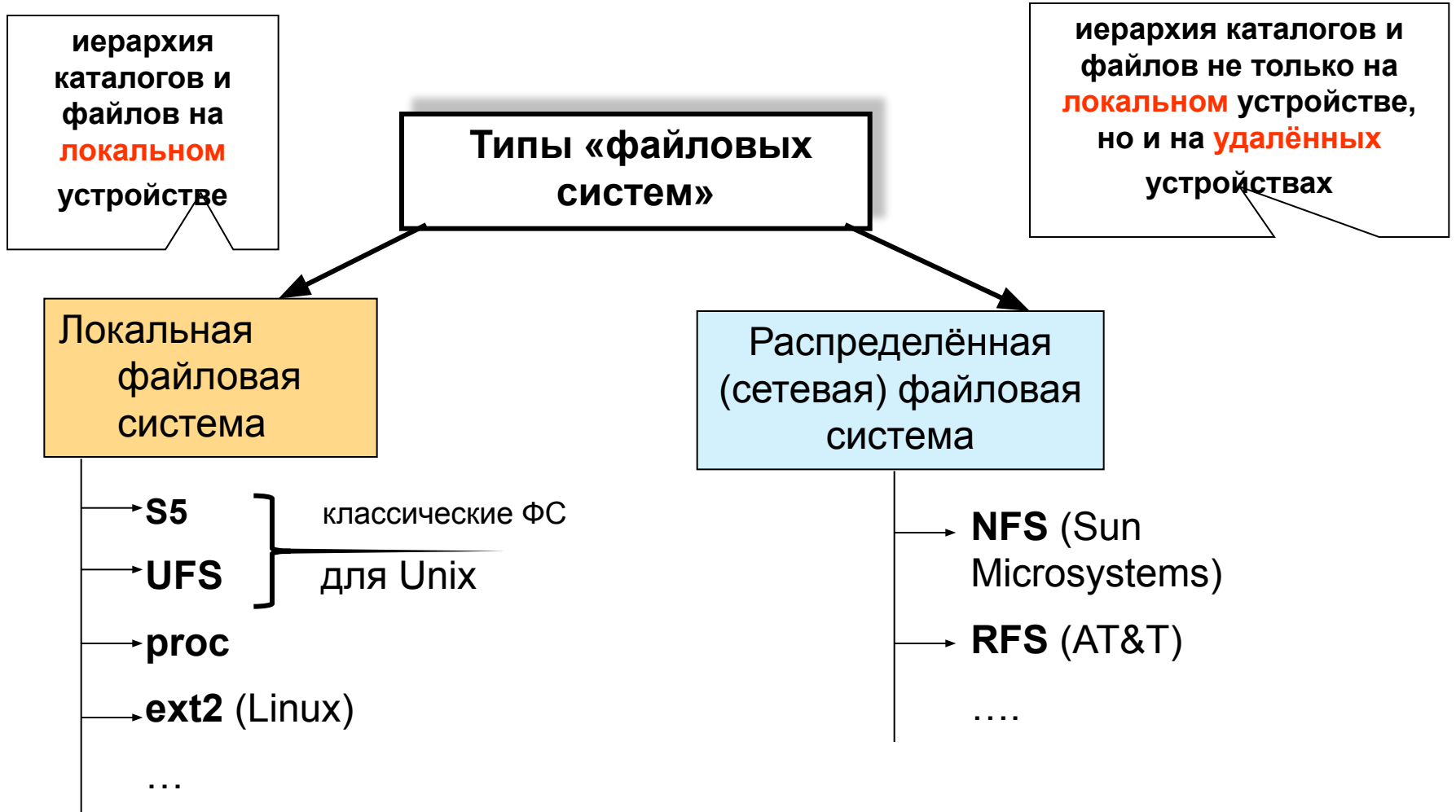
В ОС UNIX термин «файловая система» в основном используется в смысле иерархии каталогов и файлов.

Иерархия каталогов и файлов в ОС UNIX представляет **единое дерево**, которое создаётся с помощью использования концепций монтирования (команда **mount** - **монтирование**).



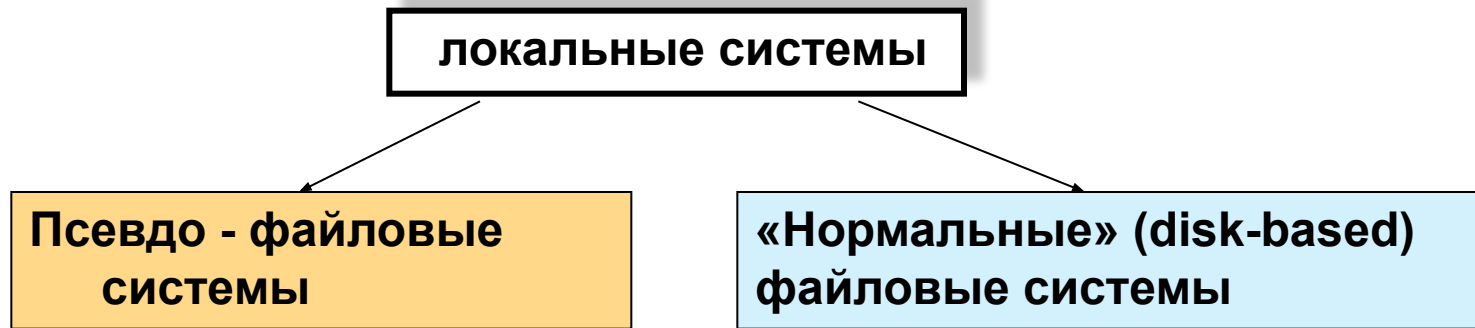
Файловые системы ОС UNIX

Принято различать следующие типы (классы) файловых систем:



Локальные файловые системы ОС UNIX

В локальных файловых системах (ФС) выделяют группы:



Примерами псевдо
файловых систем
являются:
proc, sysfs...

Примеры «нормальных»
файловых систем:
s5, ufs, ext2, ext3, XFS...

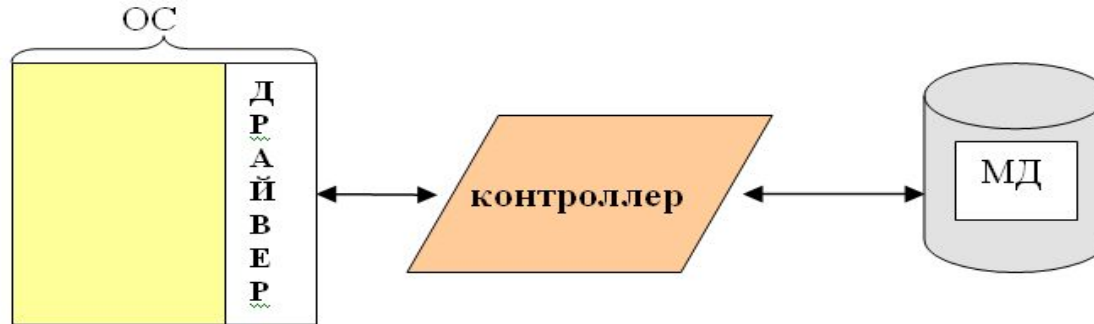
Псевдо - файловые системы располагаются в оперативной памяти.

Файловые системы современных версий ОС UNIX имеют сложную архитектуру (различную для различных версий), но все они используют базовые идеи, заложенные разработчиками UNIX (AT&T и Калифорнийский Университет г. Беркли).

Примеры файловых систем :

- **/proc** – псевдо файловая система, которая используется в качестве интерфейса к структурам данных в ядре. Большинство расположенных в ней файлов доступны только для чтения, но некоторые файлы позволяют изменить переменные ядра.
- **/tmpfs** – псевдо файловая система, которая позволяет некоторые файлы не записывать на физические диски. Эти файлы (являются временными) формируются в оперативной памяти, а затем удаляются. Поддерживает работу с виртуальной памятью
- **/devfs** – файловая система, хранящая информацию о виртуальных консолях.
- **/sysfs** – используется для получения информации о всех устройствах и драйверах
-

В качестве основного запоминающего устройства в ОС UNIX используются жёсткие магнитные диски (МД).



Связь с МД осуществляется через дисковый контроллер (электронная плата, присоединённая к шине периферийных устройств). Контроллер управляет операциями низкого уровня (пересылка данных, чтение/запись, ошибки).

Основные термины для структуры МД:

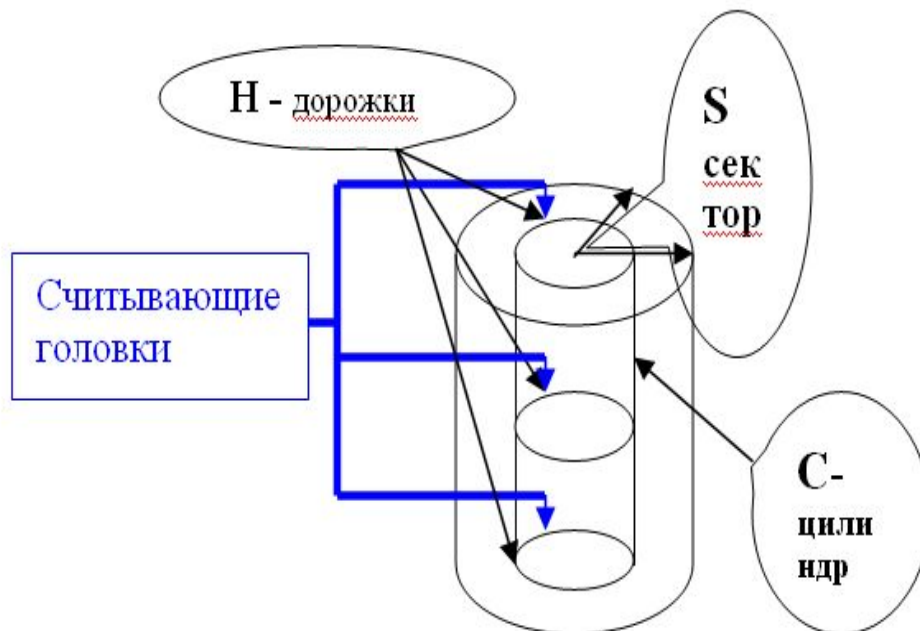
1. **S** - сектор (блок - 512 байт)
2. **H** – головка (дорожка)
3. **C** - цилиндр

физический адрес

(**C**- № цилиндра, **H** - № дорожки в цилиндре, **S** - № сектора)

Физическая модель (схема) МД.

- Доступная BIOS геометрия диска описывается в терминах **цилиндр – головка – сектор (C – H – S)**.
- Головки чтения/записи считывают информацию с концентрических магнитных **дорожек (tracks)**, на которые поделена каждая дисковая пластина.
- Вертикальная совокупность треков с одинаковыми номерами на всех пластинах, составляющих диск как физическое устройство, образует **цилиндр**.



Сектора (блоки) делят пластину, вместе с её треками на радиальные фрагменты размером в **512 байтов**. Обмен с диском возможен минимум на уровне **сектора**. Важно то, что головки диска механически двигаются синхронно по поверхности всех пластин, т.е. если на одной из пластин информация считывается с первого трека, то и все прочие головки перемещаются на ту же дорожку, каждая на своей пластине.

С точки зрения организации файловых систем интересны именно цилиндры, как совокупность треков, к которым осуществляется синхронный доступ и сектора – минимальные кванты дискового пространства.

Основные проблемы, которые возникают при работе с МД:

- 1. Скорость работы с файлами (дальнее перемещение головок чтения/записи)
- 2. Эффективность использования дисковой памяти (внутренняя фрагментация). Файлы могут занимать порядка 5-10% целого блока (сектора) - получается, что оставшаяся часть блока остается незанятой.
- 3. Эффективность восстановления файловой системы (для решения этой проблемы используется концепция журналирования).
- 4. «Маленькие» файлы ($\leq 0,5$ КБ).



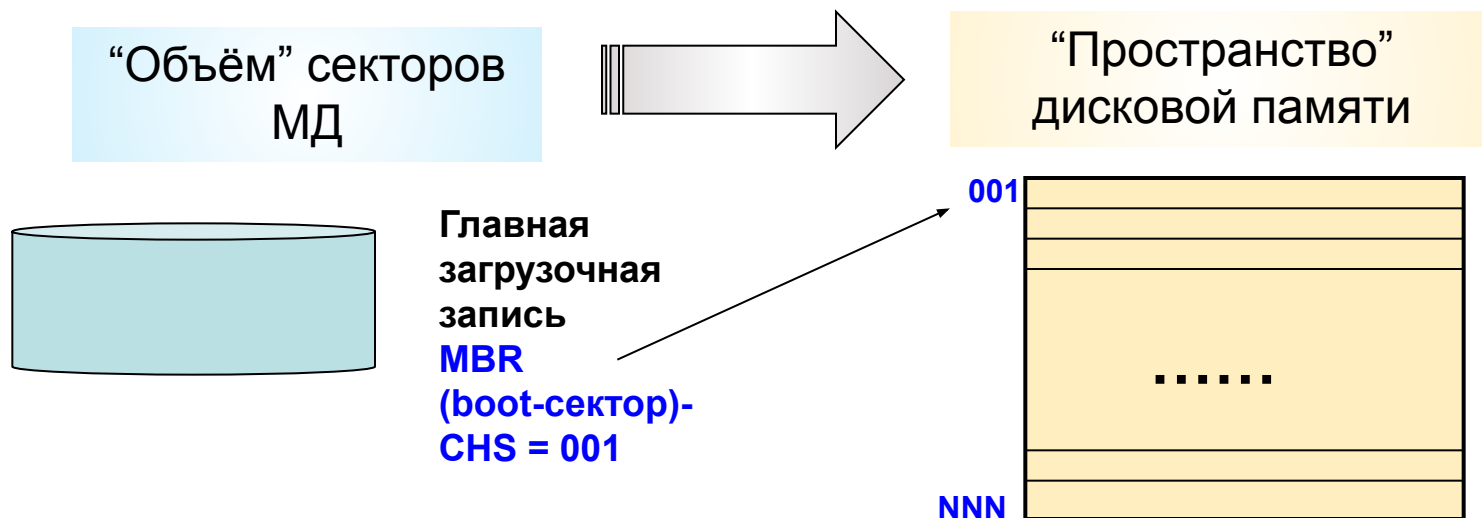
Пути решения вышеперечисленных проблем в различных файловых системах:

№	s5	ufs	ext2	ext3	XFS	ReiserFS	Примечания
1	-	Концепция группы цилиндров	Концепция группы блоков	Концепция группы блоков	Концепция равных по размеру линейных областей	Единая общедоступная среда	-
2	-	Концепция фрагмента	Переменный размер блока	Переменный размер блока	Переменный размер блока	Переменный размер блока	-
3	-	-	-	Концепция журналирования	Концепция журналирования	Концепция журналирования	-
4	-	-	-	-	-	Работа с маленькими файлами	<u>Решение</u> - хранение маленьких файлов в области метаданных

- **Замечание:** **концепция группы блоков** отличается от **концепции группы цилиндров** тем, что в ее основе лежит тот факт, что на современных магнитных дисках количество секторов на дорожке (track) уменьшается по мере приближения к центру магнитного диска.



Логическая модель магнитного диска.



Логическая модель диска основана на том, что вся совокупность (**объём**) секторов диска представляется в формате **линейного пространства**, т.е. как **последовательность номеров секторов (001 ÷ NNN)**.

Распределение ресурсов.

1. Единица распределения ресурсов кратна размеру сектора (В UNIX'е – блок, в DOS'е - кластер).
2. Состояние единицы распределения ресурса (либо занято, либо свободно).

Все пространство дисковой памяти разделено на части (разделы).
Любая («нормальная») файловая система создается в одном разделе (т.е. файловая система не может располагаться в нескольких разделах).

Разделы (partitions) диска.

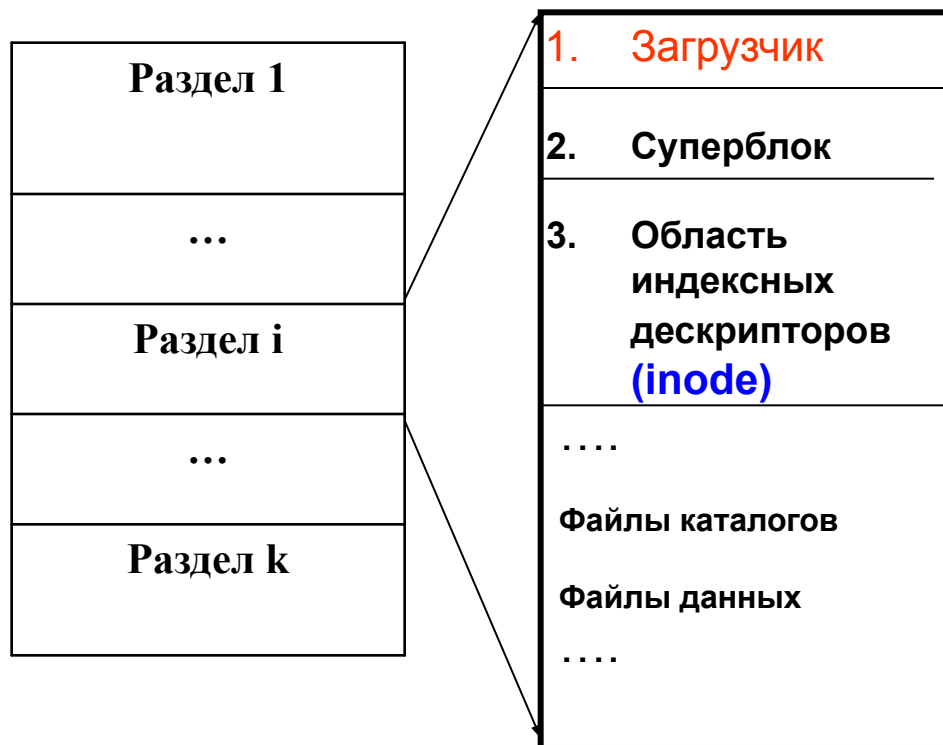
- Для распределения совокупности секторов дисковой памяти в **линейное пространство** необходима единица размещения. Единицу размещения дисковой памяти принято называть **блоком** (в системах типа Windows – **кластером**).
- **Блок** включает один или несколько **секторов**.
- Все линейное дисковое пространство обычно делится на несколько частей – **разделов (partitions)**. В один **раздел** объединяется группа **смежных цилиндров**. Разделение всего дискового пространства на разделы полезно по нескольким причинам. Например, это позволяет структурировать хранение данных и исключить (уменьшить) «дальние перемещения» головок чтения/записи и тем самым повысить скорость выполнения операций чтения и записи.
- Очевидно, что для каждого раздела следует хранить **информацию о его начале и конце** (т.е. номера первого и последнего из задействованных в разделе цилиндров).



Схема разбиения дискового пространства на разделы

Элементы файловой системы

- Файловая система каждого из разделов диска состоит из нескольких структурных элементов: загрузчик, суперблок, таблица индексных дескрипторов, блоки описания файлов, каталоги и собственно файлы.



Загрузочный блок (boot block) - это, как правило, часть метки диска (disk label). В загрузочном блоке записана **маленькая программа**, которая при старте системы загружает ядро ОС с диска в оперативную память.

Загрузочный блок располагается в первом секторе диска. Загрузочный блок имеет смысл только для **первого раздела жесткого диска**, однако место для него резервируется в каждом разделе.

Организация файловой системы S5.

Замечание. Обозначение дисков (Linux):

- `/dev/hda` – первый диск
- `/dev/hdb` – ...
- `/dev/hdc` – ...

Разделы:

- `hda1`
- ... используются только первичные разделы
- `hda4`

Логические разделы нумеруются, начинается с 5.

Суперблок – содержит самую общую информацию о ФС (размер ФС, размер области индексных дескрипторов, их число, список свободных блоков, свободные индексные дескрипторы и т. д.). Суперблок всегда находится в оперативной памяти. Различные версии ОС UNIX способны поддерживать разные типы файловых систем. Поэтому у структуры суперблока могут быть варианты (сведения о свободных блоках, например, часто хранятся не как список, а как шкала бит), но всегда суперблок располагается за загрузочным блоком.

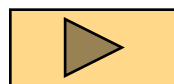
Область индексных дескрипторов состоит из **inode**'ов. С каждым файлом связан один inode, но одному inode может соответствовать несколько файлов. В inode хранится вся информация о файле, кроме его имени. Область индексных дескрипторов имеет фиксированный формат и располагается непосредственно за суперблоком

Размер индексного дескриптора фиксирован - 128 байт.
Размер таблицы индексных дескрипторов задается при создании файловой системы на разделе.

Область данных – в ней расположены как обычные файлы, так и файлы каталогов (в том числе корневой каталог).



Разделы диска



(на слайд 22)

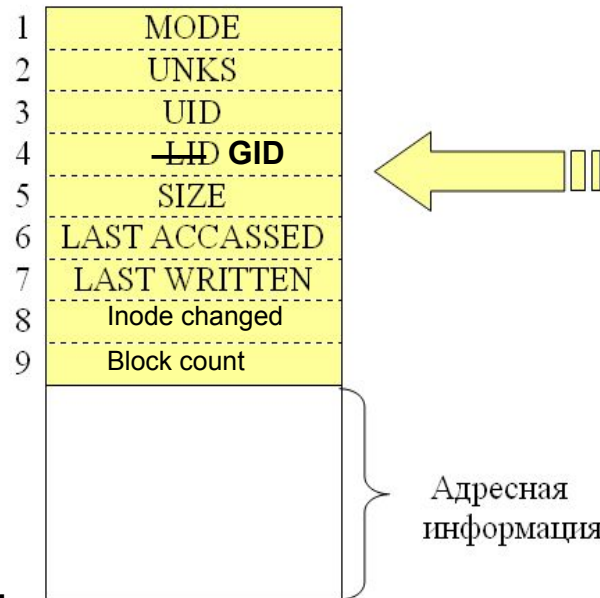


Структура индексного дескриптора (**inode**).

Вся информация о файлах, кроме их содержимого и имени, находится в так называемых **дескрипторах (описателях) файлов**.

Каждому файлу соответствует один дескриптор. Он имеет фиксированный формат и располагается **непрерывным массивом**, начиная со второго блока.

Размер одного **индексного дескриптора** зависит от типа файловой системы, однако чаще всего его размер равен **128 байт**.



1. Тип и права доступа.
2. Число ссылок (счётчик числа ссылок на файл).
3. Идентификатор владельца.
4. Идентификатор группы, к которой принадлежит владелец.
5. Размер файла в байтах.
6. Время последнего доступа.
7. Время последней записи.
8. Время последней модификации **inode**.
9. Размер файла в блоках.

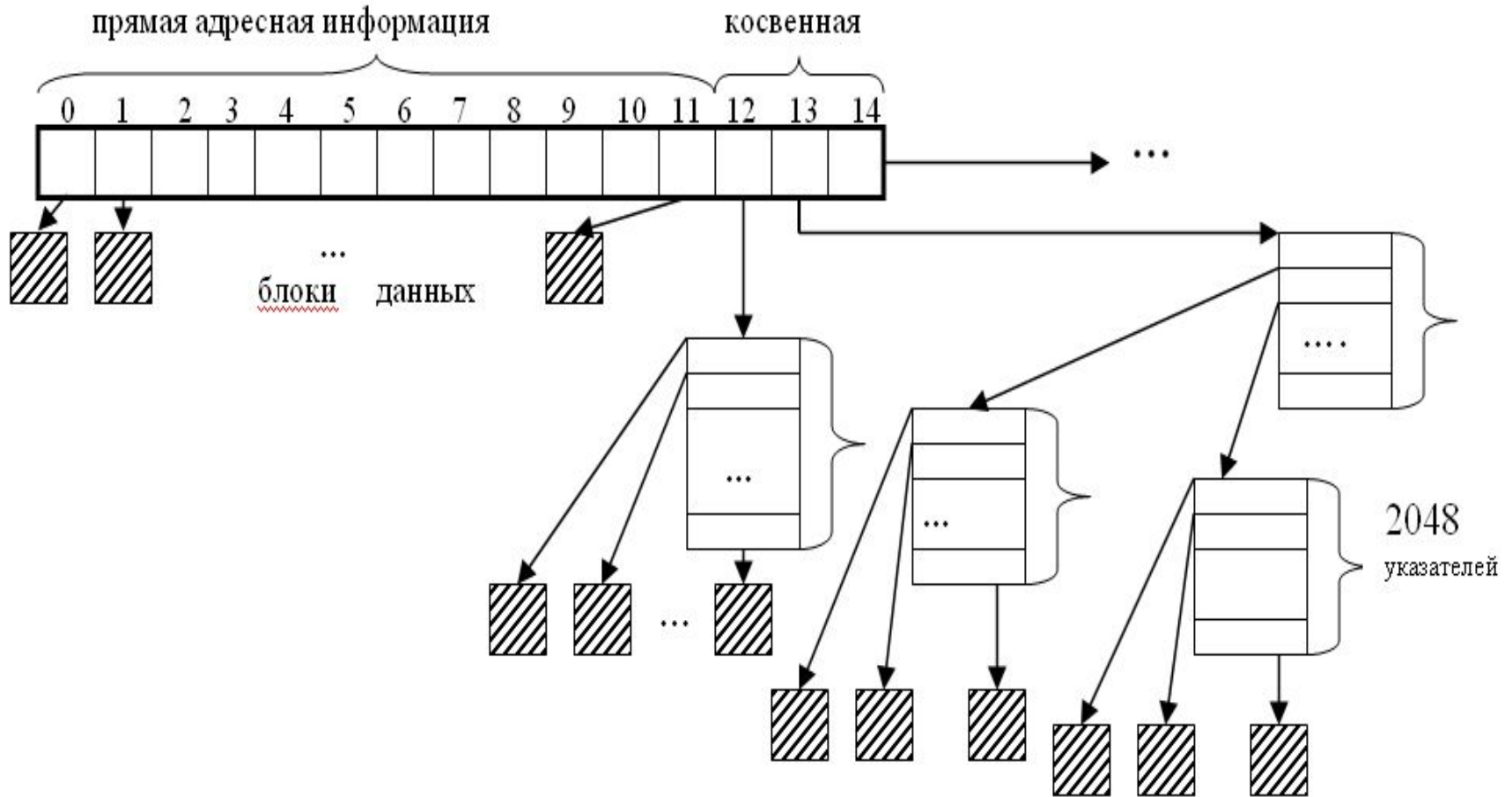
Общее число дескрипторов - описателей максимального числа файлов - задаётся в момент создания ФС. Описатели нумеруются натуральными числами.

Первый описатель закреплён за файлом «плохих» блоков. Второй - описывает корневой каталог ФС.

Назначение прочих описателей не имеет фиксированного предназначения. Зная номер и размер описателя нетрудно вычислить его координаты на диске.



Структура адресной информации **inode** в системе **S5** .



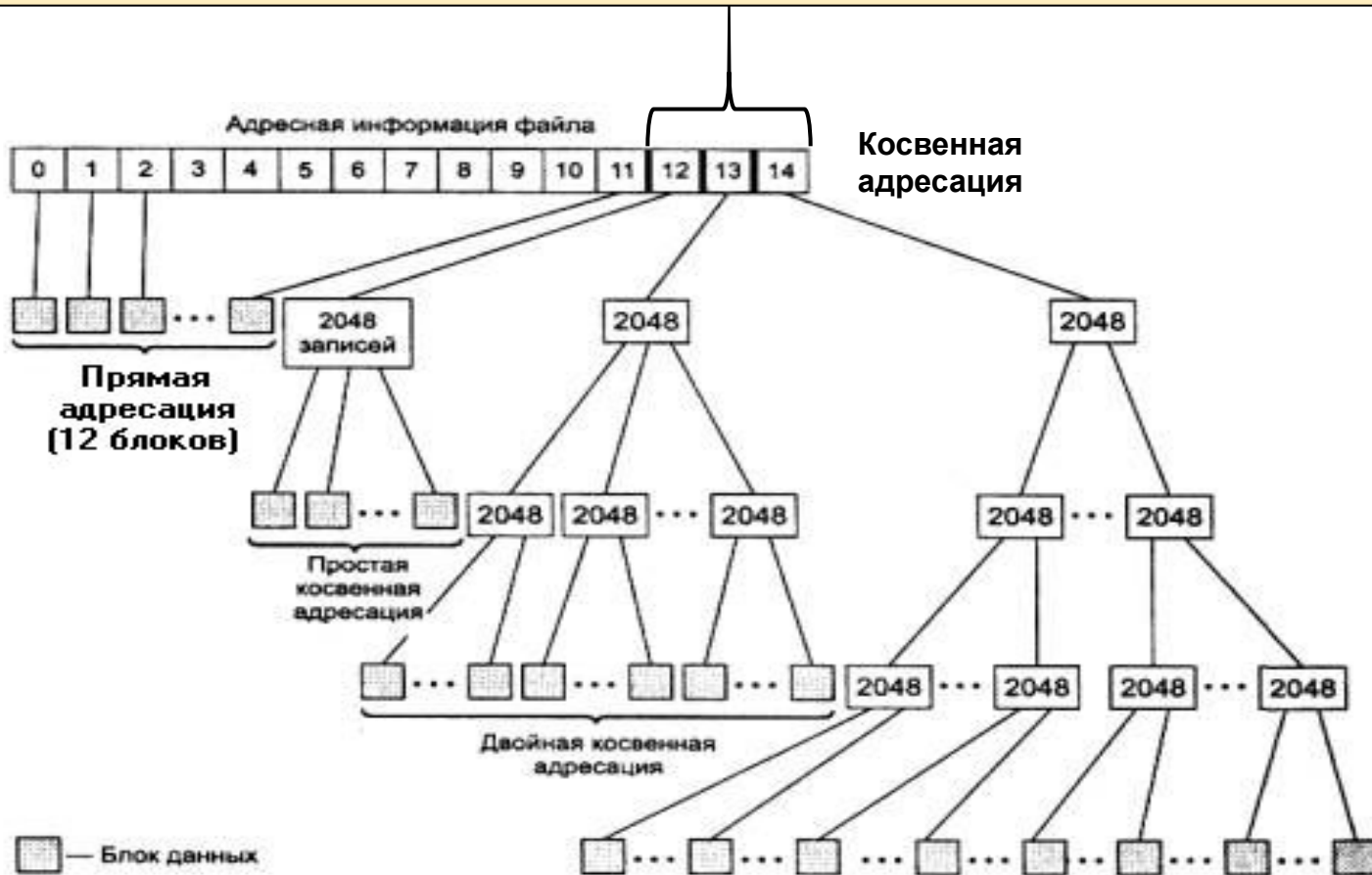
Замечание:

- Данный метод адресации задействован и в файловой системе **NTFS**, используемой в ОС Windows NT/2000/XP. Здесь он дополнен достаточно естественным приемом, сокращающим объем адресной информации: **адресуются не блоки файлов, а непрерывные области, состоящие из смежных блоков диска.**
- Каждая такая область, называемая **отрезком (run)**, или **экстендом (extent)**, описывается с помощью двух чисел: начального номера кластера и количества кластеров в отрезке.
- Так как для сокращения времени операции обмена ОС старается разместить *файл в последовательных блоках, то в большинстве случаев количество последовательных областей файла будет меньше количества кластеров файла* и объем служебной адресной информации в NTFS сокращается по сравнению со схемой адресации, используемой в различных версиях ОС UNIX.

Структура адресной информации индексного дескриптора **inode** файловой системы **ufs**.

ФС **ufs** является развитием **S5**. В **ufs** следует отметить две особенности, которые призваны решить две проблемы:

- 1- дальние перемещения головок чтения-записи (**концепция группы цилиндров**)
- 2- внутренняя фрагментация (**концепция фрагментов**)

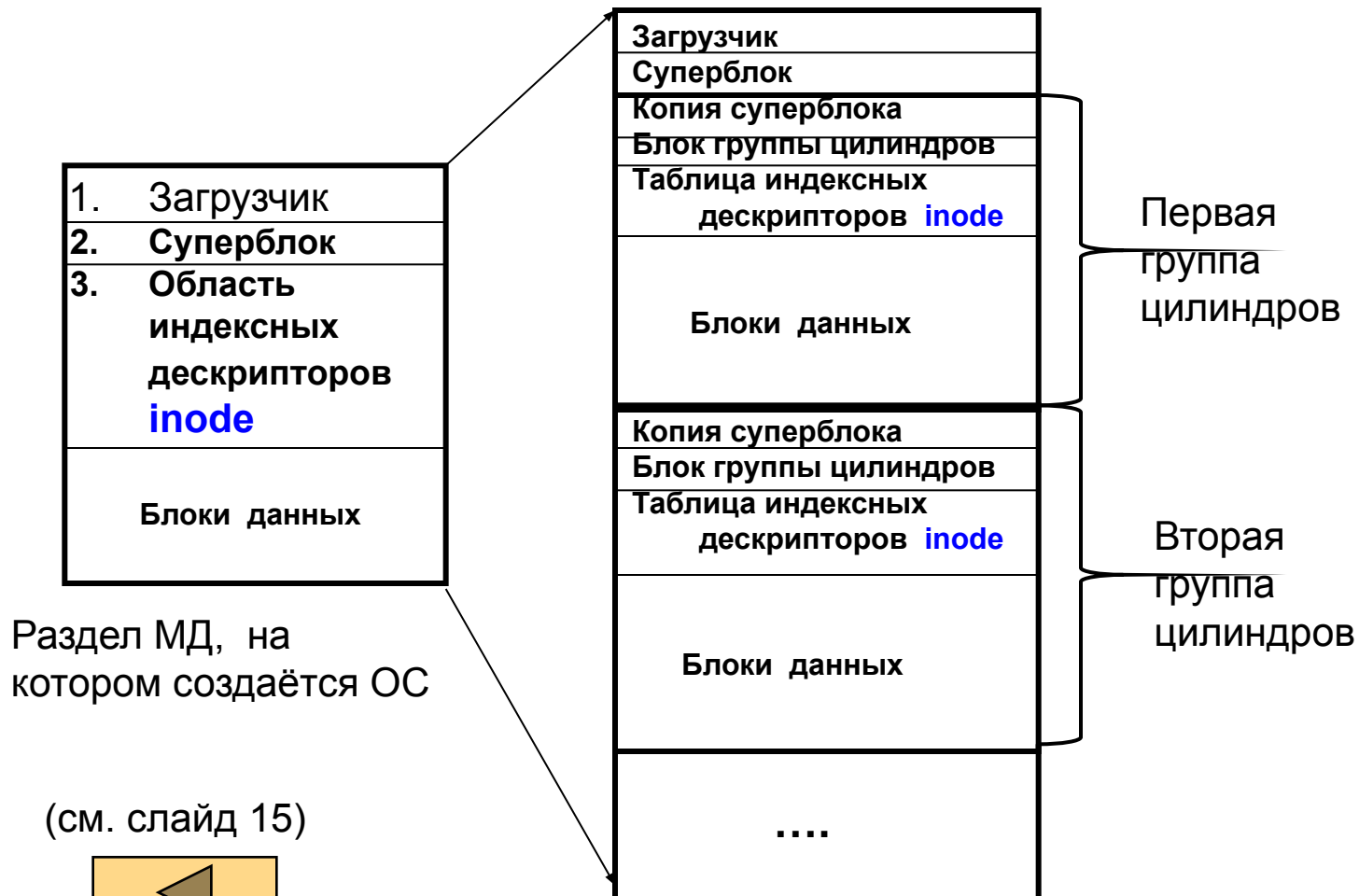


Концепция цилиндров

- После создания файловой системы файлы записываются в последовательные блоки. В дальнейшем (когда файлы создаются, удаляются, изменяются) файлы занимают любые свободные блоки. Таким образом, файл может быть разбросан по всему диску.
- Таблица индексных дескрипторов находится в начале ФС, поэтому по мере заполнения ФС и увеличения фрагментации всё больше времени тратится на частые и дальние перемещения головок чтения/записи. Для устранения этого недостатка используют концепцию групп цилиндров.
- Первоначально эта концепция появилась в ФС **ufs**. По умолчанию **ufs** использует **группы цилиндров, состоящие из 16** цилиндров.
- Каждая группа цилиндров описывается своим **блоком группы цилиндров.**

Концепция группы цилиндров

- Концепция группы цилиндров может быть проиллюстрирована следующим рисунком



Замечание:

- Файлы по-прежнему остаются фрагментированными, однако схема, которую использует **ФС ufs**, значительно сокращает фрагментацию по сравнению с **S5** (где данная концепция не используется).
- ФС стремится **размещать каталоги и входящие в них файлы в одной и той же группе цилиндров**. Таким образом, чтобы, например, прочитать файл, потребуется переместить головки максимум на 16 цилиндров.
- **Большие файлы распределяются между группами цилиндров** так, чтобы занимать не более 2 мегабайт в каждой из групп. Это предотвращает заполнение группы цилиндров одним файлом.
- Выигрыш достигается за счёт того, что дальнейшие перемещения головок осуществляются только после того, как прочиталось или записалось **2 Мб** информации.
- Эффективность схемы размещения файлов падает, если системе не хватает места для перемещения информации. Процессы чтения и записи замедляются, если свободно менее 10% ФС.
- Необходимый запас автоматически резервируется ОС и **только суперпользователь** имеет право его использовать.

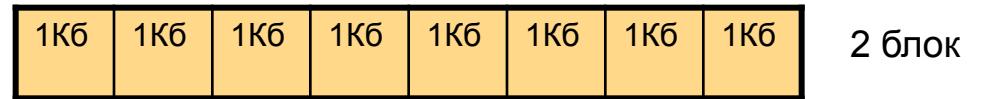
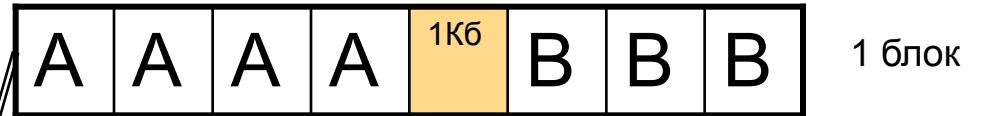
Блоки и фрагменты (в ufs).

- Преимуществом большого блока является то, что ускоряется обмен данными с диском при передаче больших объёмов информации.
- Недостаток: блоки больших размеров неэкономно используют дисковое пространство.
- Для борьбы с внутренней фрагментацией используется **метод разбиения блока на фрагменты**, которые можно распределять таким образом, чтобы файл мог и не занимать весь блок целиком.
- **Размер фрагмента не меньше размера сектора.**

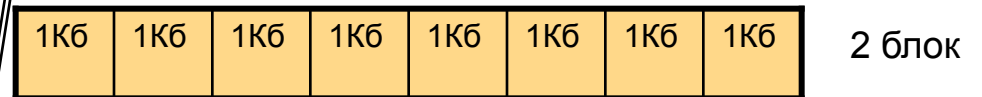
фрагмент = 1024 байт = 1Кб

блок = 8Кб = 8192 байта

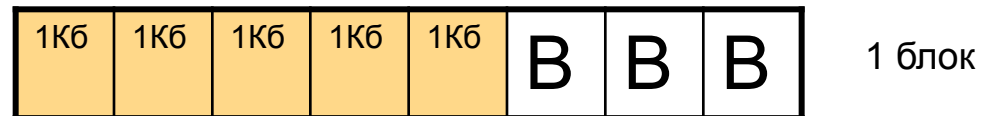
Пример:



Файл А увеличился на 1 кб



Файл А увеличился ещё на 1 кб



Взаимосвязь между элементами каталогов и описателями файлов.

1. Каталоги – это файлы особого типа (хранятся они в области данных).
2. Структура файла каталога очень проста. Каталог - это таблица, каждый элемент которой состоит минимум из 2-х полей :

имя файла	№ индексного дескриптора-inod
-----------	-------------------------------

3. Способ представления имени зависит от типа ФС. Никакой другой информации в элементе каталога нет, т.е каталог только отображает **имя файла** → **номер inod`a**

В любом каталоге содержатся два стандартных имени: “ . ” и “ .. ”.

Имени “ . ” - соответствует **inod** самого этого каталога.

Имени “ .. ” - соответствует **inod** “родительского” каталога.

(родительским называется каталог, в котором содержится имя данного каталога)

Взаимосвязь между элементами каталогов и описателями файлов.

4. Один индексный дескриптор может быть связан с несколькими именами файлов.

имя файла1 - abc	№ -inod = 1013
имя файла2 – pm3x	№ -inod = 1013

Такие ссылки называются жёсткими и могут использоваться только внутри одной ФС (нельзя создавать ссылку для файла из другой ФС. Более того – жёсткая ссылка может указывать только на файл – ссылка на каталог может привести к зацикливанию в ФС.

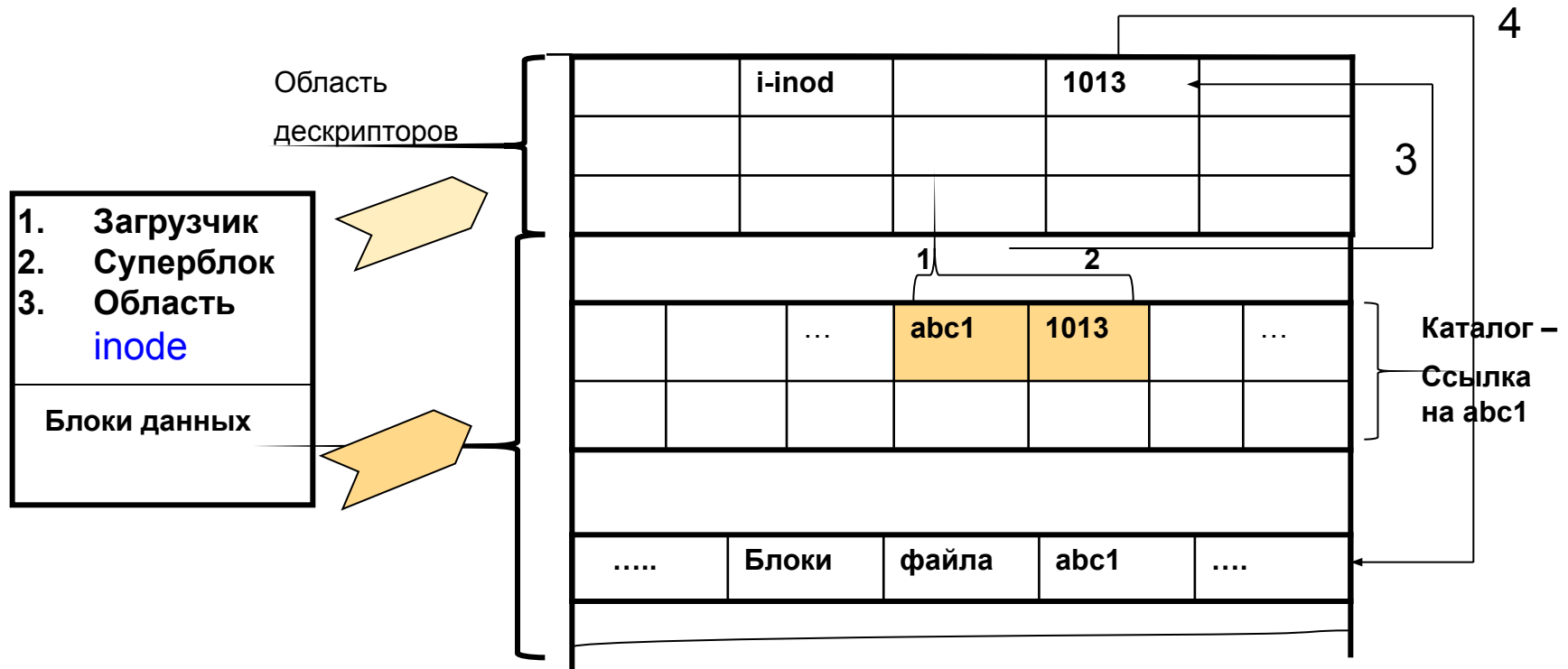
Пример: `$ln abc pm3x`

5. Существует ещё один тип ссылок. Это ссылки содержат только имя файла и называются символическими [**ключ – S в команде ln**]

Так как символическая ссылка не указывает на индексный дескриптор, то возможно создание ссылки на файлы, расположенные в другой ФС. Эти ссылки могут указывать на файлы любого типа – даже несуществующие.

ВЫВОД: С точки зрения ФС - любой каталог представляет собой обычный файл со своим описателем.

Взаимосвязь между элементами каталогов и `inod`s`



Чтобы получить доступ к файлу **abc1** ядро ОС выполняет действия

1. Находит имя **abc1** в каталоге, содержащем ссылку на этот файл.
2. Выбирает номер инд. дескриптора (**1013**) файла **abc1**.
3. По номеру **1013** находит `inode` в области дескрипторов.
4. Из `inode` – адреса блоков данных файла **abc1** – по этим адресам считывает блоки данных

Монтирование ФС.

- Каждый **раздел диска** чаще всего **содержит свою файловую систему**. Чтобы удобно представлять данные в виде дерева, недостаточно хранить содержимое файлов, их имена и структуру этого дерева. Нужно решить массу технических задач, связанных с быстродействием, надежностью, распределением свободного места и т. д.
- Одну из доступных файловых систем ядро считает **корневой**. Эта файловая система **монтируется** на корневой каталог, в результате чего ее содержимое становится доступно в виде дерева каталогов, растущего непосредственно из /. Любой из каталогов текущего дерева может служить точкой монтирования другой файловой системы.

При загрузке корневая ФС автоматически монтируется ядром.

Администратор должен позаботиться о том, чтобы другие необходимые ФС были также подмонтированы. Обычно они монтируются при загрузке, однако это может быть сделано и после процесса загрузки – командой:

mount [-опции] [файловая система] [точка монтирования]

Поясним термин - монтирование файловых систем.

Пример:

- **#mount /dev/hda9 /home**

Чтобы сделать ФС недоступной (отмонтировать):

- **umount [-опции] [ФС или точка монтирования]**

- После выполнения команды mount вида (**mount /dev/hda9 /home**) содержимое файловой системы, лежащей на разделе, становится доступно в виде дерева, растущего из **каталога**.
- Список всех файловых систем, которые монтируются по ходу начальной загрузки, обычно лежит в файле **/etc/fstab**.
- Помимо дисковых файловых систем там можно встретить упоминание **файловых систем в памяти** (временных) или **procfs** – это ФС, позволяющая в виде дерева каталогов представлять структуру **процессов UNIX**.
- Некоторые устройства (например, CD-ROM) помечены **noauto** в знак того, что при старте их монтировать не надо. Запись в **fstab** служит только напоминанием, какое именно устройство какой точке монтирования соответствует.

Замечания: названия стандартных каталогов

В UNIX существует довольно строгая договоренность относительно того, **как должны называться стандартные каталоги** системы и для чего их следует использовать.

Регулярно выпускается **документ, именуемый FHS (Filesystem Hierarchy Standard)** Во многих системах есть отдельная страница руководства **man hier**, подробно описывающая основные каталоги и их назначение.

Содержимого каталогов корневой файловой системы должно быть достаточно для аварийной загрузки и "лечения" UNIX.

- в **/bin** и **/sbin** должны лежать только самые необходимые пользовательские и системные утилиты,
- в **/lib** - все, что необходимо для работы этих утилит;
- в **/dev** UNIX хранит всевозможные файлы-устройств,
- в **/boot** - все, что необходимо для досистемной загрузки.
- в специальном каталоге **/tmp** - кто угодно и когда угодно может (временно) хранить свои файлы.
- важен каталог **/etc**, содержащий все настройки системы (включая файлы паролей и настройки программных продуктов).

Содержимое этих каталогов занимает, как правило, не очень много места; его удобно копировать на какой-нибудь резервный носитель



далее: названия стандартных каталогов

- Каталог **/var** предназначен для файлов, размер (и количество) которых все время меняется: для системных журналов (**/var/log**), почты (**/var/mail**), очередей (на печать, на выполнение - **/var/spool**) и т. п..
- Каталог **/mnt** содержит временные точки монтирования, то есть пустые подкаталоги, на которые при помощи `mount` можно временно отобразить содержимое какой-нибудь файловой системы (например, того же CD-ROM), не опасаясь, что какие-то файлы при этом не будут видны.
- Каталог **/home** принято отводить под домашние каталоги пользователей.
- Каталог **/usr** содержит все то, чего не было в **/**, и что необходимо для штатной работы системы.
- Многие каталоги называются так же, как и подкаталоги корневого: **/usr/bin**, **/usr/sbin**, **/usr/lib** и другие; их назначение повторяет назначение одноимённых корневых каталогов.
- Подкаталог **man** содержит страницы помощи, **info** - info-систему, **doc** - прочую документацию, **locale** и **nls** задают язык диалога с пользователем (например, русский) и прочие особенности национальной формы представления данных (даты, времени, денежных единиц)
- И т.д.



Создание ФС.

ФС в разделе диска создаётся командой **newfs**.
В этой команде требуется указать имя раздела и строку аргументов, которые будут переданы команде **mkfs**.

mkfs – создаёт ФС. Команде **mkfs** требуется указать параметры:

- а) *имя раздела*, где создаётся ФС;
- б) *размер блоков* раздела;
- в) *количество* описателей файлов.

ФС в минимальном варианте содержит корневой каталог и каталог **lost+found**.

**СПАСИБО
за внимание!**

Конец лекции №3