Лекция №3. 07.03

Файловая подсистема

- Писать Unix разработчики начали с файл. системы
- Ни одна развитая ОС без файл. системы не может функционировать, вплоть до памяти — всё на основе файлов
- В Unix всё файл (Устройство тоже файл)

↓

За что не возьмись, везде файловые дескрипторы

- Файлы предназначены для долговременного хранения информации

Опр. – **Файл** – это ∀ поименованная (т.е. имеющая имя) совокупность данных, может быть бессмысленная, во вторичной памяти. ∃ временные файлы, кот. могут храниться в оперативной памяти.

Основная задача файлов – долговременное хранение данных

~~Файл – поименованное место на диске~~ – НЕВЕРНО

- Современные системы поддерживают так называемые "очень большие файлы" и т.е. на диске они хранятся вразброс, т.е. у этих файлов нет одного единственного места на диске — они хранятся в свободных блоках диска и каждый этот блок адрес

ется.

Интерфейс.

- В Unix был реализован интерфейс VFS/Vnode
  Virtual File System
- В Linux — VFS — это будем изучать
  В Linux разработчики удачно назвали Virtual node (vnode)

исходное название

inode — одна из основных структур, представляющая в системе файл.
(сокр. от index node)

Фактически inode это дескриптор файла

Этот (VFS) интерфейс представлен в системе 4 основными структурами:

- struct superblock
- struct inode (**)
- struct dentry
- struct file

Название Virtual: Смысл виртуал. интерфейса в том, что Unix поддерживает большое кол-во раз-

ных файловых систем.

· Unix, Linux - ОС с монолитным ядром, ядра минимизированные.

· Unix имеет иерархич. структуру - эти уровни иерархии располагаются по отношению к аппаратному обеспечению

· Файловая система имеет иерархич. структуру

Уровни файловой системы:

(1). В Unix имя файла не св-е его идентификатор. Когда говорят имя файла, которое всегда подразумевается полное имя файла, начиная с корневого каталога "/", содержит все элементы пути до очередного "/" и завершается "коротким именем" (short name) - это собственное имя файла, кот. имя дано файлу (но перед этим файл был размещен в опред. директории (выбранной папке).

(2) Идентификатор файла это (inode) номер inode.

(3) Когда создается файл для него устанавливаются права доступа, файл имеет тип (в Unix 7 типов файлов)

read write execute
user group other

Когда происходит непосредственное обращение к файлу, то система проверяет, имеет ли данный пользователь обратиться к этому файлу.

(4) Есть
- текстовые файлы (или символьные, байт-ориентированные)
- бинарные файлы (или двоичные, блок-ориентированные)

Такие как в программа считает, что она начинается с 0го адреса. И файл в логическом представлении начинается с нуле-

вого смещение.

(5) Очевидно, что это смещение для того чтобы реально обратиться при чтении к нужным данным, кот. находятся на диске, необходимо получить физический адрес этих данных — это обратная работа системы.

разработка 1963–1964

• Linux поддерживает стандарт FHS (The Filesystem Hierarchy Standard). Этот стандарт иерархии ФС определяет структуру каталогов и содержимое каталогов.

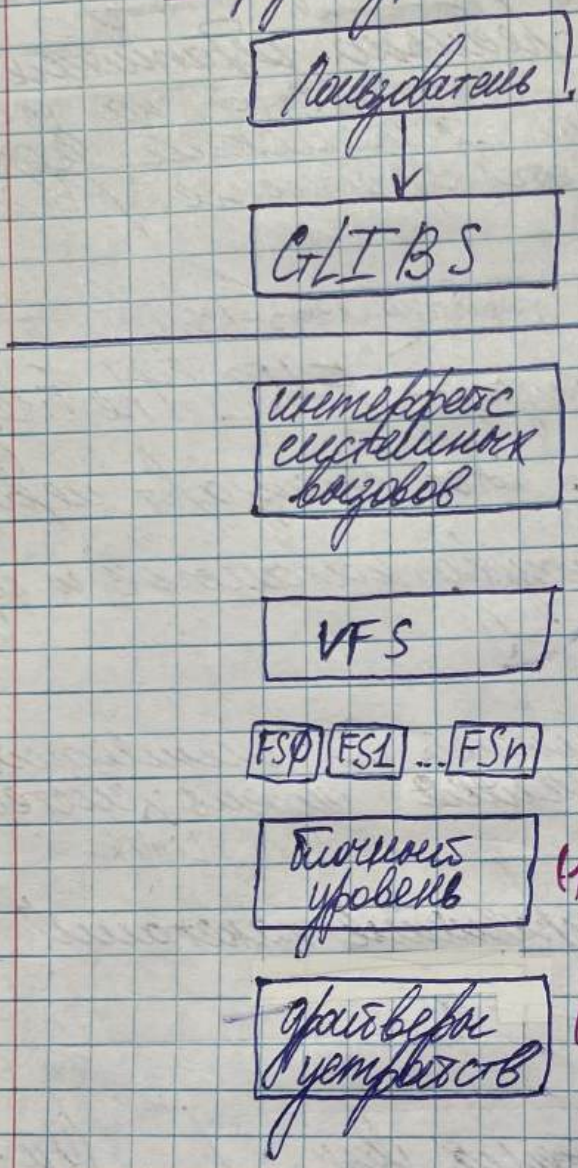
• GoboLinux — в этой свободной ОС меняется исторически сложившаяся модель систем Linux и

вводится понятие „управление пакетами"

• Ubuntu поддерживает FHS

Корневая файлов. система обозн. „/". При этом по стандарту FHS корневой каталог и его ветви обязательно должны составлять единую ФС, расположенную на одном носителе (диске, дисковом разделе и т.д.)

Структура VFS



• VFS определяет интерфейс к которым должны подчиняться конкретные ФС, чтобы работала ОС

(1) Несмотря на то, что внешние устройства похожи на запоминающие, системой они рассматриваются как блочные (block device)

В системе имеется 2 типа девайсов (устр-в):

- character (символьные устройства)
- block (блочные устр-ва)

часто выделяются отдельно сетевые устройства в сетевой подсистеме, но они рассматриваются как символьные.
(stream-потоки байтов)

2) В символьном устр-ве управление ему. программный, кот. наз-е драйвер драйвера обеспечивает возможность обращения к физическим записям на диске.

(устройство диска – фото в беседе)

Физ. диск имеет концентрические окружности. Аппарат. средства должны обеспечивать адресацию информации, кот. находится на диске.

Важная информация:

- Формат диска, иначе способ разделения потоков данных на независимо адресуемые порции, называемые секторами, внутри каждого из которых имеются адресные метки (они устанавливаются аппаратно)
- Адресные метки и метки данных, которые отмечают различные виды информации, хранящиеся в пределах сектора.

Др.: - Сектор – участок дорожки магнитного диска, представляющий собой наименьший размер порции данных, которая может быть изменена в результате перезаписи. (это концентрич. окруж-ть)

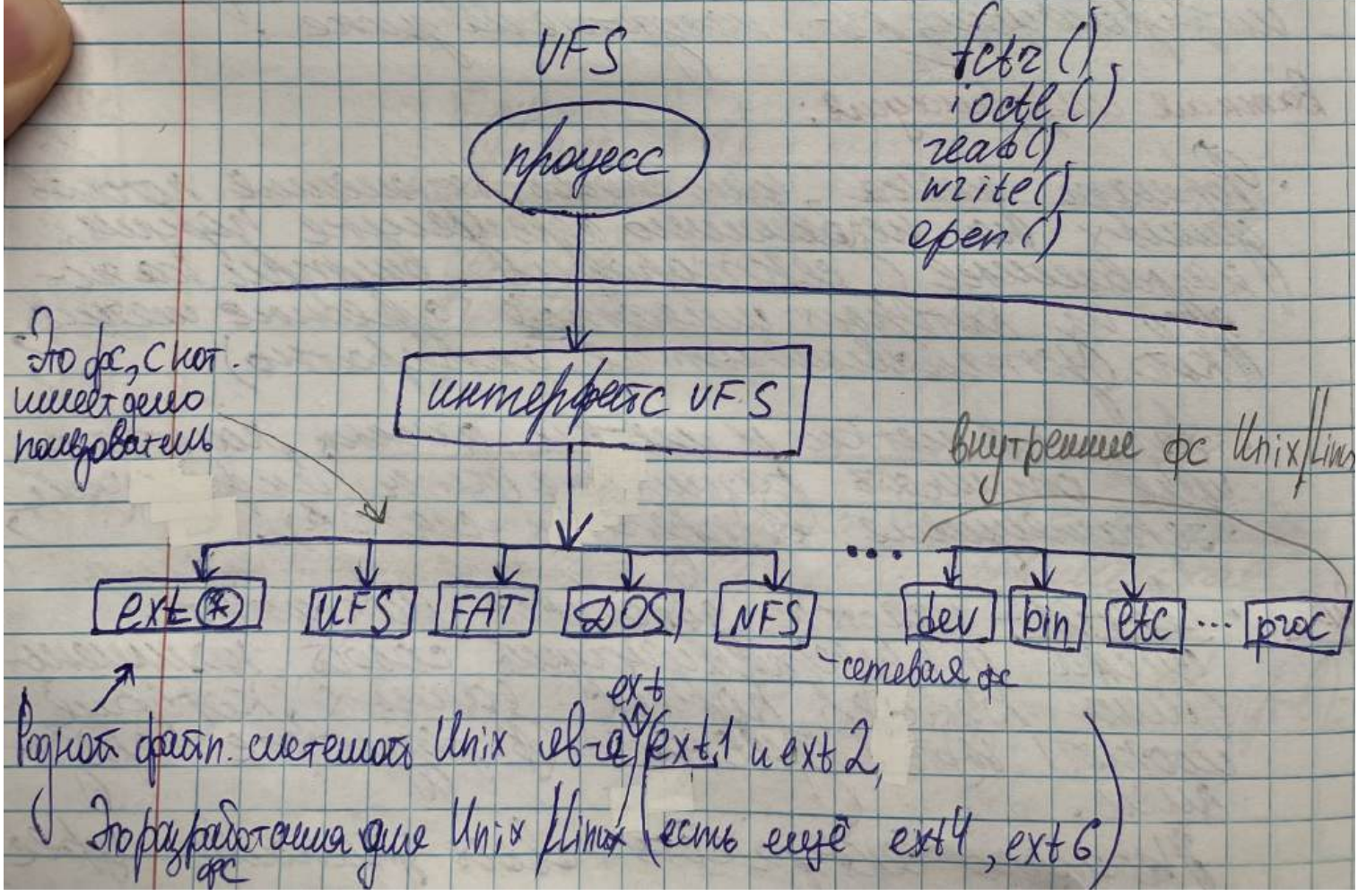
У каждого сектора есть свой уникальный адрес,

кат. состоит из номера дорожки и номера сектора.

Пр • Адресная метка на диске — это спец. код, т.е. часть этой дорожки, область на дорожке магнит. диска, которая предшествует адресной информации. Т.е. также как в физ. памяти, на диске вся инф-ция адресуется (имеет уникальн. адрес)

Ж

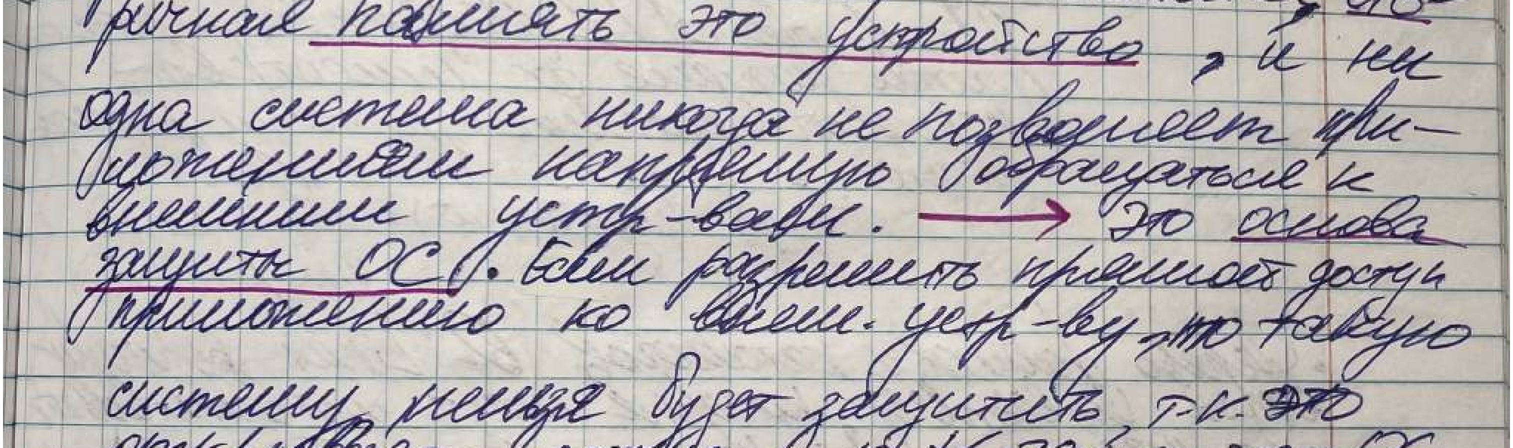
(Архитектура) Структура VFS?



· ∀ ОС для работы с файлами предоставляет соотв-ующий интерфейс - это сис. вызовы

. API (application program interface) - набор ф-ций, которые предоставляет ОС. В результате вызова этих API система переходит в режим ядра. Именно ядро выполняет все основные действия с файлами. - почему? - потому что файлы находятся во вторич. памяти, вторичная память это устройство, и ни одна система никогда не позволяет приложениям напрямую обращаться к внешним устр-вам. → это основа защиты ОС. Если разрешить прямой доступ приложению ко внеш. устр-ву, то такую систему нельзя будет защитить т.к. открывает доступ к ∀ части этой ОС.

· Поэтому все через переход в режим ядра, через этот интерфейс.



В Unix/Linux вызов write() приведет к вызову sys_write() уже в реж. ядра. И это в свою очередь приведет к вызову функ write() ОС Unix/Linux. И в результате будет выполнено обраще-

ние к физ. уровню.

Монтирование ФС

Чтобы сделать ф/с доступной в Unix/Linux определено „монтирование фс".

Пример: подключение к вычислительной системе флешки. У флешки своя ФС. Система скрыто от пользователя (от нас) выполняет действия по монтированию ФС флешки к общему дереву каталогов у которого 1 корневой каталог, хотя каждая фс имеет свой корневой каталог, поэтому разница . и ..
(текущ.) (родительский)

Unix/

- Дерево файлов и каталогов в Linux формируется из веток, которые могут соответствовать различным физ. носителям. Т.е. можно сказать, что оно формируется из отдельных ФС.
- ФС Unix/Linux это одно большое дерево с корневой директорией („/").
- При монтировании фактически монтируется ФС, расположенная на каком-либо устройстве.
- Монтирование корневой ФС в-е часть процесса инициализации ос.
- Монтирование — это система действий

в результате которой ФС устройства становится доступной.

Базовая форма команды mount принимает на вход 2 пар-ра:

#mount <устройство> <точка монтирования>
(1) (2)

(1) — имя устройства или другого ресурса, содержащего маршрутизатор ФС

(2) — точка монт-я — это каталог, к которому подмонтируется ФС.

- Когда файлов. система монтируется в существующую директорию, все файлы в поддиректории этой монтируемой ФС становятся файлами в поддиректории точки монтирования.
- Если директория т. монтирования содержала какие-либо файлы поддиректории, они не теряются, но становятся невидимыми, поэтому лучше использовать пустые каталоги

- Полное представление команды mount:

mount книги -t <тип_файловой_системы> -O <опции_файловой_сист>
устройство точка_монтирования

путь к физ. устр-ву или каталог имя каталога, куда будет смонтирована ФС

umount - отмонтирование

∃вариации, такие как mount.nfs - освобождает от необходимости указания типа, ключей, опций
network filesystem

· Структуры (**) связаны меж собой. Когда создаётся файл задействуются все эти структуры, но в разный момент работы с файлом.

Взаимосвязь объектов виртуальной ФС

· Всё идёт от процессов. Пр-сы создают файлы, записывают в файлы инф-цию, читают инф-цию из файлов. - Процессы заинтересованы в работе с файлами.

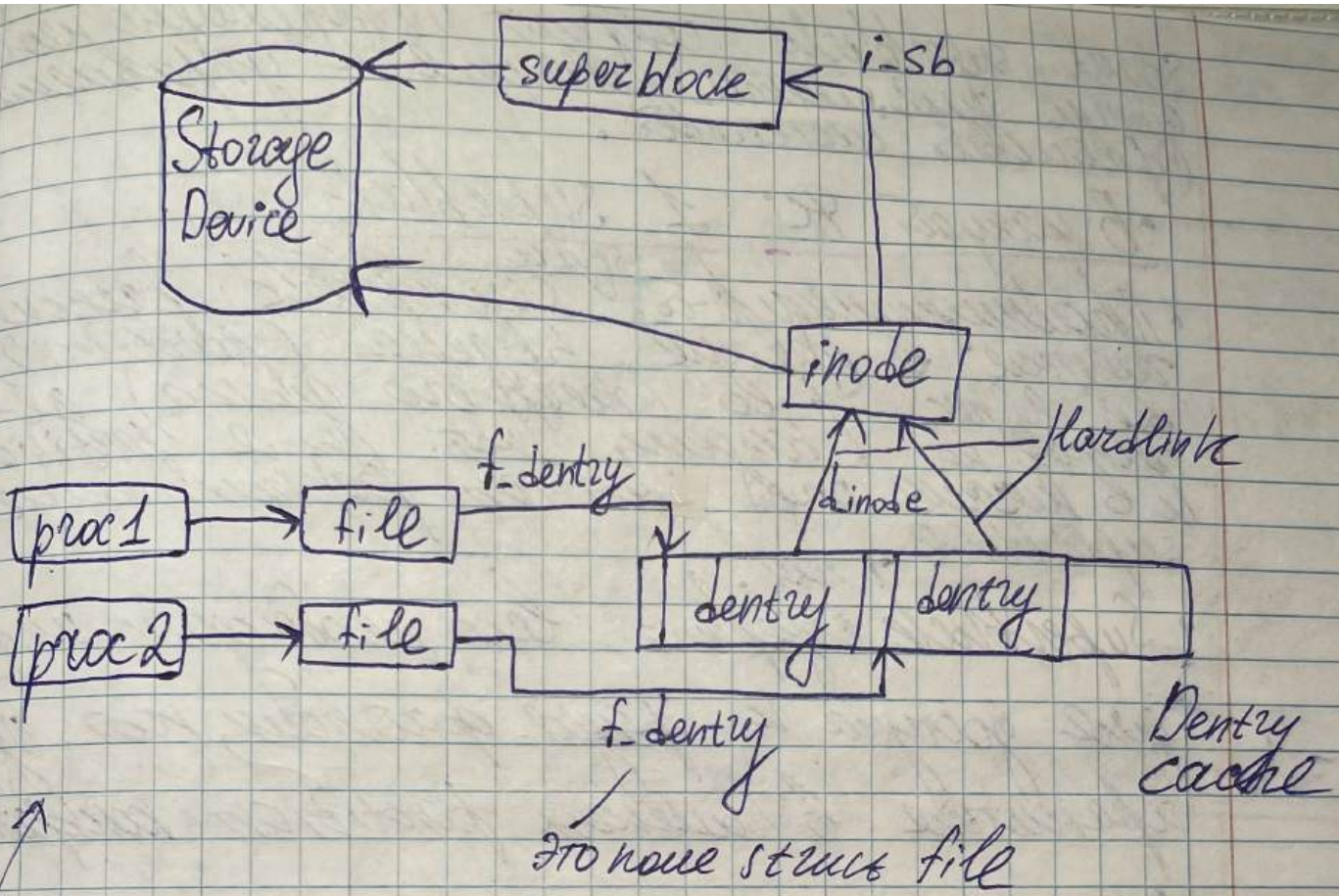
open() возвр. файл. дескриптор

· Файл это структура. Они называют проинициализированные процессом структуры „объектом"

· В результате вызова соотв. сис. вызовов инициализируется наша структура struct file.

· Для доступа к файлу необходимо указывать полное имя файла и это элемент каталога dentry (сокр. от Directory Entry)

struct dentry



· В struct dentry есть наше d_inode.
· dentry это инф. файл, указ. есть inode. Система не различает файлы и dentry (они хранятся на диске). обеспечивает единый. уровень работы с файлами
· Дерево каталогов - основа обеспечения доступа пользователя к файлам.

Демонстрирует связь объектов на основе структур ядра

SuperBlock (наше такое контейнер для мета-данных высокого уровня)
Структура, описывающая - superblock, это struct superblock

Struct superblock - содержит информацию необходимую для монтирования и управления файловой системой.

- В каждой ФС 1 superblock.

- Располагается struct superblock загружается в начале раздела (партиции) т.е. все физ. ур-во (жесткого диска) делится на разделы кот. на-е partition и в начале этой partition располагается superblock, кот. содержит всю необходим. инф-цию для доступа к другим файлам, кот. будут храниться в рамках partition конкретной ФС.

- Superblock содержит инф-цию кот. обеспечивает доступ к другим данным т. е. данным, находящимся на физ. носителе.