Содержание

1 Лекция 4

1.1 Монтирование и файловые системы

Путь файла начинается с корневого каталога, который обозначается /. Монтирование - система действий, в результате которой файловая система становится доступной. Для монтирования требуются права и привелегии пользователя. Для монтирования используется команда mount, имеющая следующий синтаксис:
mount ключи -t тип_файловой_системы -o опции_файловой_системы устройство каталог_назначен

Для размонтирования приеняется команда unmount:
unmount ключи -t тип_файловой_системы -o опции_файловой_системы

Кроме основной команды существуют дополнительные команды, в которых например может указываться имя файловой системы, например mount_nfs (Network File System). Если подмонтирована файловая система windows, то mount_ntfs.
Наиболее часто в команде mount используется два параметра - имя устройства, или другого ресурса, который содержит монтируемую файловую систем и точку монтирования.
Точка монтирования - каталог к которому подмонтируется файловая система. Точка монтирования должна существовать, иначе возникнет ошибка.
Когда файловая система смонтирована в существующую директорию, все файлы и поддиректории этой смонтированной файловой системы становятся файлами и поддерживают точку монтирования.
Если директория точки монтирования содержала в себе какие-либо файлы и поддиректории, то они не теряются, а становятся невидимыми.
Иногда может оказаться нужным явно указывать при монтировании тип файловой системе. Для этого в команде mount используется опция -t. Это нужно для того, чтобы отследить попытку монтирования файловой системы, использующую новый тип. Unix/Linux могут поддерживать большое количество файловых систем. Существует структура, описывающая тип файловой системы.
Рассмотрим пример:
mount /dev/sda1 /mnt - опции -t -o отсутствуют, данная команда пробует монтировать раздел sda1 с файловой системой ext3 в каталог /mnt в режиме только чтения. Если в системе нет библиотек для работы с той или иной файловой системой, или система в указанном разделе не является ext3, будет выдано сообщение о невозможности монтирования.
Если требуется включить режим записи, то необходимо добавить # mount -o rw /dev/sda1 /mnt.

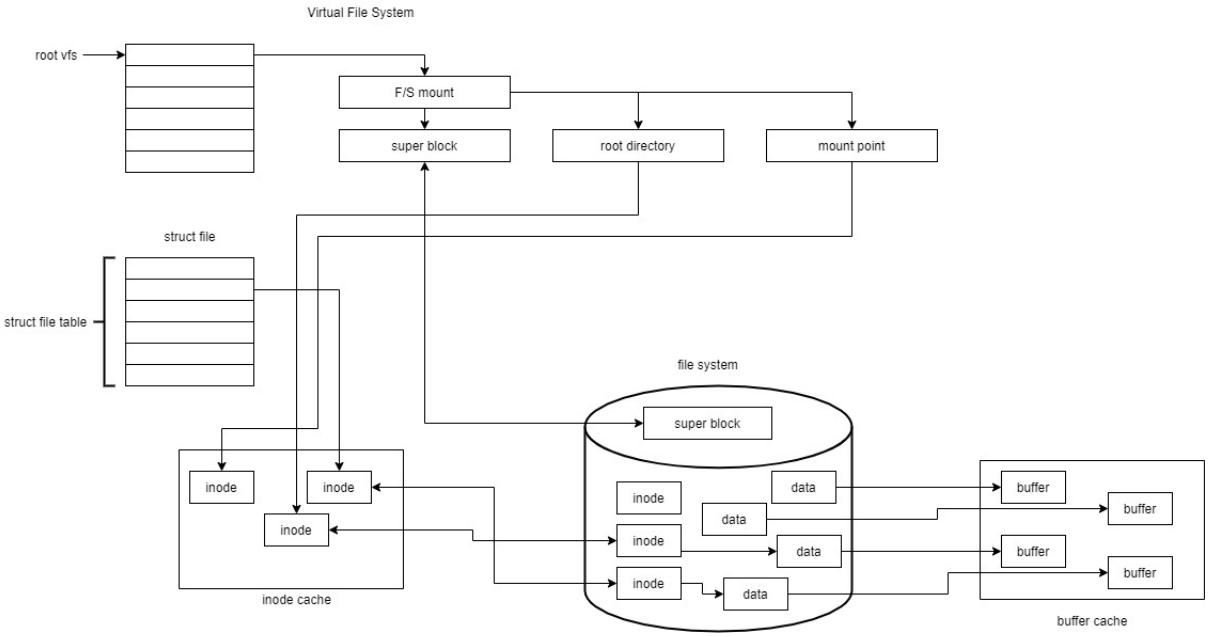

Рисунок 1.1 — Схема работы файловой системы

Структура superblock предназначается для подмонтированных файловых систем. Данная структура содержит всю необходимую информацию для обращения к файлом конкретной файловой системы. struct inode - струткура физического файла. struct ientry - структура, описывающая элемент каталога и предназначена для доступа к файлам. struct ifile - структура, описывающая открытый файл, при этом открытый файл - это файл, который открыт каким-то процессом.
Пользователя для системы не существует. Для системы существуют только процессы, которыми она управляет.
inode - два варианта, не являющиеся копиями. В ядре существуют структуры, важные для действий в ядре. Мы видим, что inode существует дисковый и ядре. Нужна точка монтирования, нужен корневой каталог.
Есть кеш inode и буферный кеш (кеш данных), в системе всё буферизируется. Кеши построены по принципу LRU (last recently used). Поскольку они не могут быть любого нужного размера (ограничены возможностями физического хранения в ядре). По своему назначению - хранят данные, к которым были последние обращения. Также здесь присутствует системная таблица открытых файлов. В этой системной таблице находятся дескрипторы всех открытых в системе файлов. Причём если файл был открыт несколько раз, в этой таблице будет существовать соответствующее количество дескрипторов открытого файла. Так ак открытые файлы - это структура, предназначенная для обслуживания процессов (процессы открывают файлы). При этом разные процессы могут открывать один и тот же файл. У этого файла может быть один и тот же inode (существуют hard линки) - система не различает имена файлов (первое, второе и тп.) Все имена системы - хардлинки.

1.2 Суперблок

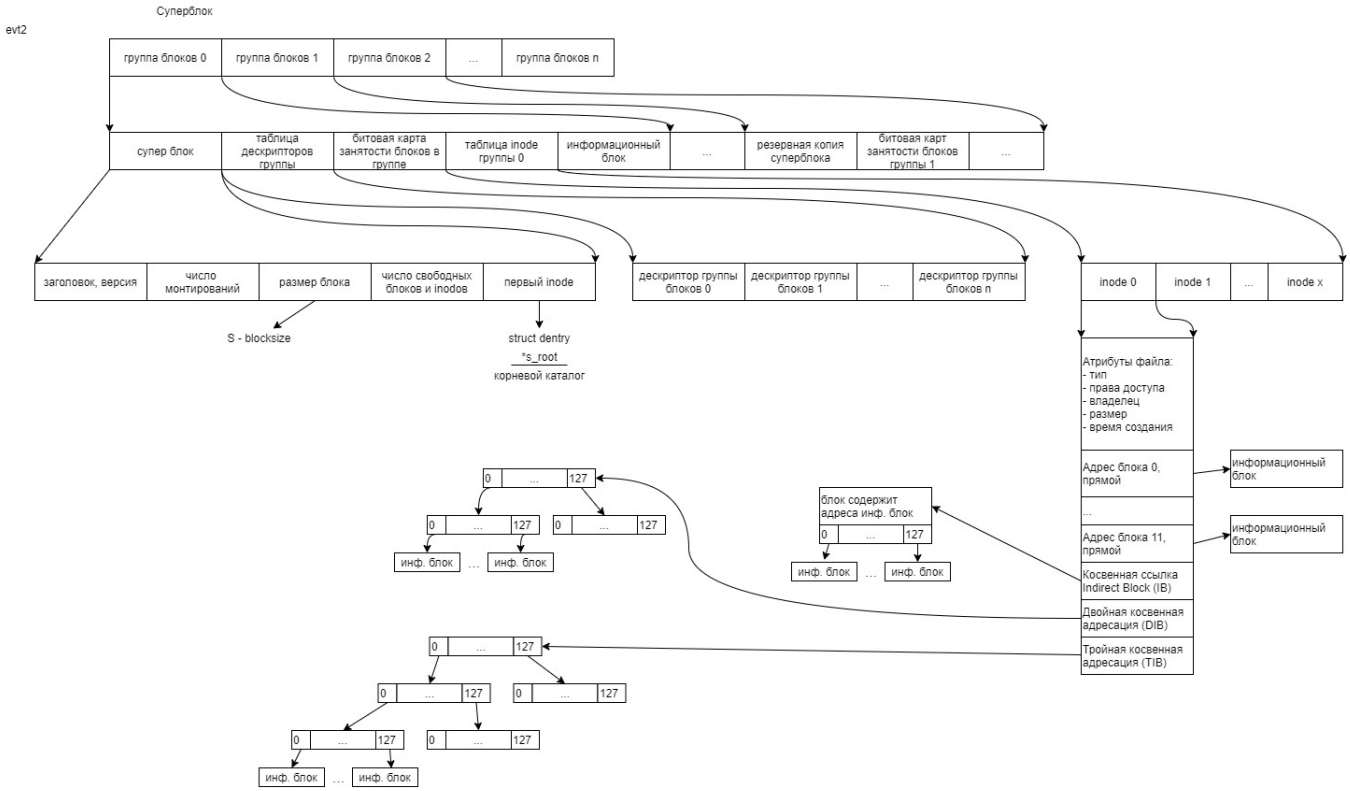

Рисунок 1.2 — Схема работы суперблока

В данном примере мы рассмотрим раздел жёсткого диска (раздел вторичной памяти) с файловой системой ext2 (родная файлова система linux).

Блок - минимальная адресуемая единица физического носителя (вторичной памяти). В разных системах размер может отличаться. Каждый блок имеет уникальный адрес.

Любая файловая система предназначена для обеспечения долговременного хранения и доступа к файлам. Без доступа все это не нужно. Кроме того, что файловая система предназначена для хранения информации, однако в первую очередь она должна обеспечивать доступ к этой информации. Доступ - многоуровневый (вторичная память - внешнее устройства) на каком-то последнем этапе происходит обращение к устройству.

inode описывают физический файл, и они же так называемые дисковые inodeы содержат информацию об адресных блоках вторичной памяти, в которой находится файл, который описывается конкретным inodom.

Очевидно, что суперблок должен содержиться список inodов, но он должен содержать первый inode - inode корневого каталога.

Файловая система ext2 существует достаточное большое количество времени по причине того, что она хорошо написана. Данная система обеспечивает хранение и доступ к очень большим файлам. По аналогии с физическим адресным пространством оперативной памяти, если первые операционные системы требовали. чтобы процессу было выделено непрерываное адресное пространство, и это оказалось невозможным при быстром росте прикладного ПО. Такое требование

в современных системах физически обеспечить невозможно. Не существует достаточно большого объекма физической памяти для обеспечения одновременной работы большого объема фзической памяти для обеспечения одновременной работы большого количества приложений с высокими требования к физической памяти, что характерно и для файлов. Размеры файлов также постоянно растут. Это было понято разработчиками Linux/Unix. Ими было предложено решение, когда файлу выделялось не непрерывное адресное пространство, а была обеспечена возможность выеления свободных участков адресного пространства. В результате было обеспечено хранение и доступ к файлам очень большого размера.

Практически все источники приводят эту информацию о файловой системе ext2 как наиболее яркий пример работы с файлами очень большого размера.

Видно, что суперблок хранит информации о inodax, обеспечивая доступ к дисковому inody, а дисковый inode хранит информацию об адресных блоках вторичной памяти, в которых располагается данные данного файла.

В этой файловой системы существует несколько типов адресации - прямая, косвенная, двойная косвенная и тройная косвенная.

Для адресов 12 блоов (0 - 11) используются для прямой адресации. Соответственно это непосредственно адрес информационного блока.

Следующие блоки содержат косвенные ссылки (indirect block). Этот блок содержит адреса инфомационных блоков.

```
struct super_block {
struct list_head head s_list;
dev_t s_dev;
unsigned long s_blocksize;
struct file_system_type *s_type;
unsigned long s_flags;
unsigned long s_magik;
struct dentry *s_root;
struct rw_semaphore s_unmount;
...
struct list_head s_mount;
struct block_device *s_bdev;

const struct dentry_operations *s_d_op;
...
/* s_node_list_lock protects s_inode*/
spin_lock_t s_inode_list_lock;
struct list_head s_inodes;
...
struct list_head s_inodes_wb;
```

};

Суперблоки объединены в списокк. В системе будет существовать столько суперблоков, сколько смонтировано файловых систем. При этом может быть смонтировано несколько файловых систем одного и того же типа, поэтому первое поле представляет из себя список.
Очевидно, что смотированая файловая система должна находиться на каком-то девайсе или на части ос, на части адресного пространство оперативной памяти или на внешнем устройства (это должен быть физический носитель) - второе поле.
Важное значение имеет размер блока (он является минимальной адресной единицей на вторичной памяти) - третье поле.
Тип файловой системы - важнейшее понятие в Linux, так как может поддерживать большое количество файловых систем. Структура file_system_type предназначена для регистрации типа конкретной файловой системы:
Есть флаги для работы с файловой системой
Есть магическое число для обеспечения надёжной работы
Указатель на root
Семафоры для чтения-записи

На суперблоке определён набор операций, которые с ним можно выполнять - struct super_operations.
В любой из перечисленных структур есть ссылка на операции, которые могут выполняться на этом суперблоке, фактически таблицу. Дентри являеся элементом пути к файлу. Каждый элемент пути имеет элемент inode и хранится на диске. Каждая структура содержит средства взаимоисключения. Список inodов должен защищаться, для этого используется спинлок.
Кроме этого есть список inodов. В этом списке находятся все inody - дескрипторы физических файлов, созданных в конкретной файловой системе.
s_inodes_wb - Write Back inodes, список грязных inodов. Грязным файлом называется изменённое значение.
struc super_operations
{
struct inode *(*alloc_inode)(struct super_block *sb);
void (*destroy_inode)(struct inode *);
void (*free_inode)(struct inode *);
};

Очевидно, что поскольку любая файловая система предназначена для хранения физических данных. Причём не просто данных, а поименованных. Мы видим, что это работа с inodом.