Superblock и Inode.

- **Struct super_block** - стр-ра, описывающая смонтированную ФС.

  Это стр-ра для монтирования ФС

- super_block содержит метаданные для обращения к файловой ФС, которую представляет super_block.

- АП диска делится на разделы (partitions) и один раздел занимает одна ФС.

  выглядит это след. образом → фото на тетради
  схема "Раздел плоского диска с ФС ext2"

- Мы обсуждаем ФС EXT2   пояснение

- Раздел - выделенное на диске адрес. пространство и оно характеризуется группами блоков.
  (группа блоков 0, группа блоков 1, ... гр. бл n)

- Благодаря VFS Unix/Linux поддерживают большое кол-во разных ФС. Для этого в ядре определена стр-ра struct file_system_type. Мы можем создать свой тип фс, определив поля стр-ры file_system_type.

• Мы можем также зарегистрировать свою ФС, т.е. проинициализир-ть струк file system'ы в ядре.

В системе может находиться только 1 стр-ра, описывающая конкрет. тип ФС. Т.е. тип ФС определяется единожды.

• С помощью proc можно посмотреть ФС, зарегистр-ые в ядре.

А вот super_block для одной и той же ФС может смонтирован много раз.

↓
поэтому в struct super_block есть поле "число монтирований"

• ФС монтируется в дерево каталогов

• ext2 — ранняя ФС Unix/Linux. Особенность ext2: она позволяет создавать и адресовать очень большие файлы.

• Я 2 стр-ры struct inode:
1) дисковый inode (описывает файл и предоставляет инф-ию об адресах блоков, в которых находится инф-ция

данного файла, т.е. у дискового inode есть адресная часть) (*)

2) <u>адрёсность inode</u> (описывает файл в дре. Сделано это для устра- нение доступа к памяти зовись inode, т.е. обраще- ние к диску это длительное дейст-твие)

• В системе вся инф-ция о файлах кэши- руется. Т.е. есть <u>кэши inode</u> — это всё для сокращение доступа к инф-ции, которую хранят файлы

чтобы адресовать большие файлы
(*) в <u>адрес. части inode</u> есть Знач: (на схеме показано)

1) <u>адрес блока прямой</u> (прямй адрес будт иметь <u>11 блоков</u>, затем следную указ- по это косвенный адрес)

2) <u>косвенный адрес</u>

3) <u>двойной косвенный</u>

4) <u>тройной косвенный</u>

- Все типы файлов имеют inode, в том числе директории. Инф-ция о директориях также хранится во внешней памяти.

ext2 — second extended filesystem (дисковой)

- inode содержит 15 блоков указателей:

  - 12 блоков прямой адресации (соответствующий элемент содержит прямой адрес блока)

  - 12й содержит адрес 1024 записей (это простая косвенная (адресация))

  - 13й блок содержит 1024 записи: каждая запись содержит указ-ль на 1024 записи и уже они содержат адреса блоков данных
    (двойная косвенная)

  - 14й ...

15

```
| 0 | 1 | 2 | ... | 11 | 12 | 13 | 14 |
```

[1024 записи] 4кб

блок данных

[ ][ ] ...

Простая косвенная адресация

[1024 записи] 4кб

[1024 записи] ... [1024]

[ ][ ] ... [ ][ ] ... [ ]

двойная косвенная адресация

[1024]

[1024] ... [1024]

[1024] ... [1024]   [1024] ... [1024]

[ ] ... [ ]   [ ] ... [ ]

...

Тройная косвенная адресация

• Имя файла в Unix/Linux не своё его идентификатора. Идентификатор файла — номер inode. Этот номер нужен, чтобы обратиться к inod'у в таблице inod'ов

? (В файл записываем новые акции. Для этих новых данных надо выделить место на диске — это будет блок. Этот блок надо адресовать. Его адрес надо включить в соотв. стр-ру и это super_block)

Левая страница

В ядре определена стр-ра list_head: в нее //2 поля -next и -prev. Это стр-ра для создания двусвязных списков.

```
struct super_block
{ struct list_head s_list; // двусвяз. список суперблоков
  dev_t s_dev; // устр-во (девайс) (индекс нашел)
  ...
  unsigned long s_blocksize; // размер блока
  loff_t s_maxbytes; // макс. размер файла
  struct file_system_type *s_type;
  const struct super_operations *s_op; // функц. ядра, определенные на super_block
  const struct export_operations *s_export_op; // функц, кот. не были учтены в пред. поле
  unsigned long s_flags; // флаги монтирования ФС
  ...
  unsigned long s_magic; // ключ к смонтиров-й ФС
  struct dentry *s_root; // точка монтирования
  struct rw_semaphore s_umount; // read/write семафор
  int s_count; // счётчик ссылок на super_block
  ...
  struct block_device *s_bdev; // связанное блочное устр-во
  ...
  char s_id[32]; // текстовое имя
  ...
  unsigned int s_maxlinks; // макс. к-во ссылок
  fmode_t s_mode; // права доступа при монтировании
  const struct dentry_operations *s_d_op; // dentry операции на конкр. super_block
  (*) наш по inode
  spinlock_t s_inode_list_lock;
  struct list_head s_inodes; // список inod'ов (он защищён spin-lock-ом); все inod'ы
}
```

Правая страница

Проинициализировав спец. образом поле s_type file_system_type можно создать собств. ФС

создать стр-ру ядра ⟶ Проинициализ-ть поле s_type ✓

→ Суперблок сможет смонтировать конкретную ФС (напр. ext2)

(кот. работают с super_block)
функц ядра, определенные на super_block
кот. не были учтены в пред. поле

• Все стр-ры ядра содержат в первом поле ср-ва взаимоисключения. Т.е. на каждую стр-ру определены спец. для этой стр-ры ср-ва взаимоискл-я. В ядре есть семафоры ядра, т.е. постоянн-е кричатели. Это связано с тем, что в ядре потокам ядра часто требуется прочитать инф-цию (читать могут параллельно), а наши потоки могут изменять данные (тогда они будут работать в режиме взаимоисключения).

Spin-lock построен на команде test-and-set там идёт while, в кот. вызывается команда test-and-set

```
struct list_lru  s_dentry_lru  __cacheline_aligned_in_smb
struct list_lru  s_inode_lru        — " — ;
```

Это тоже есть в нашем super_blocke

T.e. эта
инф-ция
кэшируется

· Действия, определённые
в системе на super_block

```
struct super_operations
{
    struct inode * (*alloc_inode)(struct
                        super_blocke  *sb);
```

Это inode связан → создаёт и
с конф. super_block'ом инициализирует новый объект inode

```
(1)   void (*dirty_inode)(struct inode *, int flags);

(2)   int (*write_inode)(struct inode *, struct write_back_control
                                        *wbc);

(3) int (*drop_inode)(struct inode *);
    ...

(4)   void (*put_super)(struct super_blocke *);
    ...

    int (*freeze_super)(struct super_blocke *);
    int (*freeze_fs)(struct super_block *);
    ...
}
```

↑ + destroy_inode — удаляет inode

(1) — вызывается подсистемой VFS , когда в инод
вносятся изменения

(2) – записывает inode на диске и помечает его как „грязность"

(3) – сбрасывает inode

(4) – вызывается при размонтировании ФС

$\boxed{\text{Лекция } 04.04.}$

dentry и inode

интерфейс

В основе ↓VFS лежат 4 структуры:

- struct super_block
- struct dentry
- struct inode
- struct file

- Ф-ии, опред. на super_block, составляют struct super_operations

- В struct super_operations нет ф-ии
  – создаёт объект суперблок
  alloc_super() — эта ф-ия размещает и инициа-
  лизирует структуру super_block.

- static struct super_block *alloc_super(struct
  file_system_type (*type, int flags, struct
  user_namespace *user_ns)
  {

  struct super_block *s = kzalloc (sizeof (struct
  super_block), GFP_USER);