MC504/MC514 - Sistemas Operacionais

Sistemas de Arquivos (Parte III)

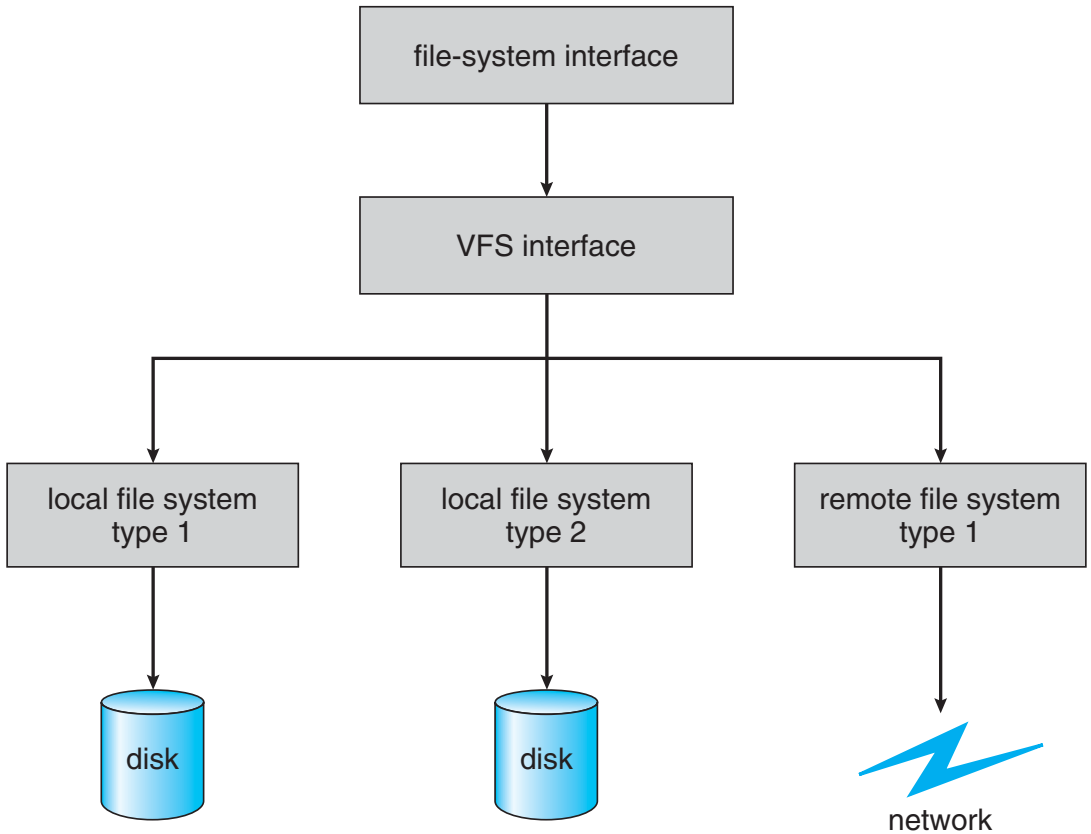
Islene Calciolari Garcia

Instituto de Computação - Unicamp

Segundo Semestre de 2016

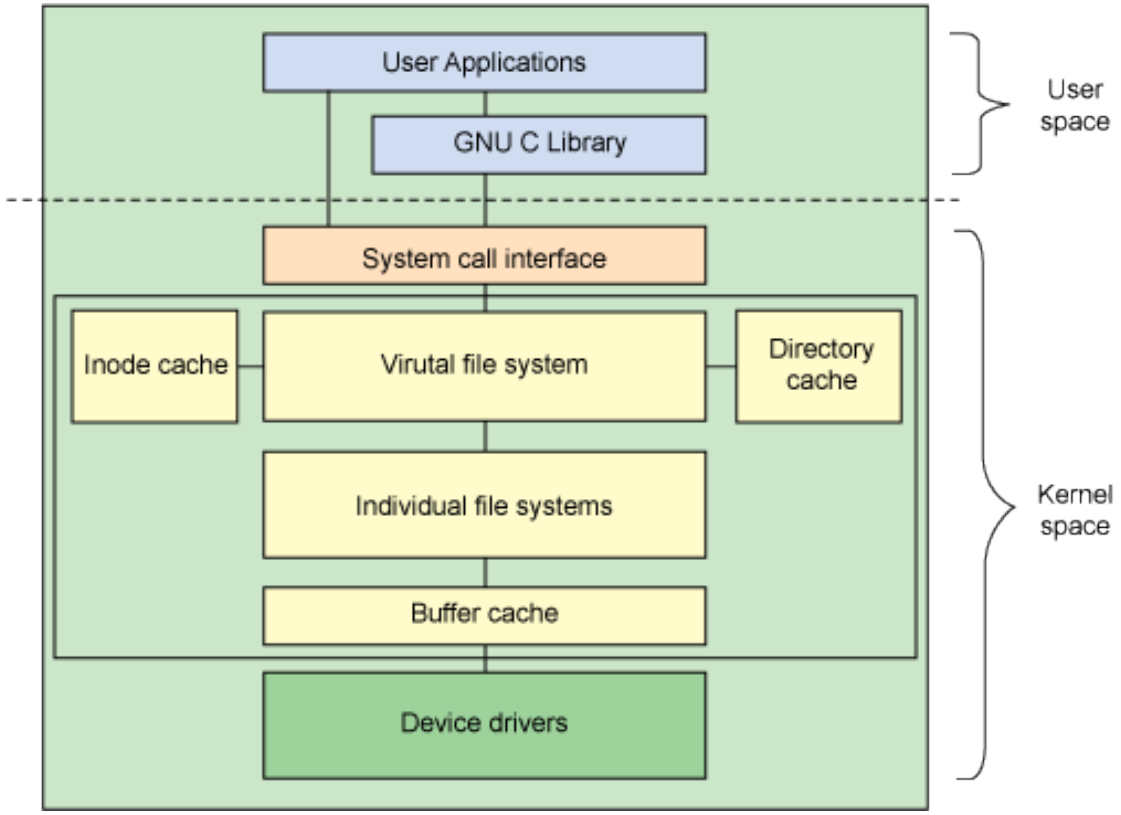
Sumário

Virtual File System



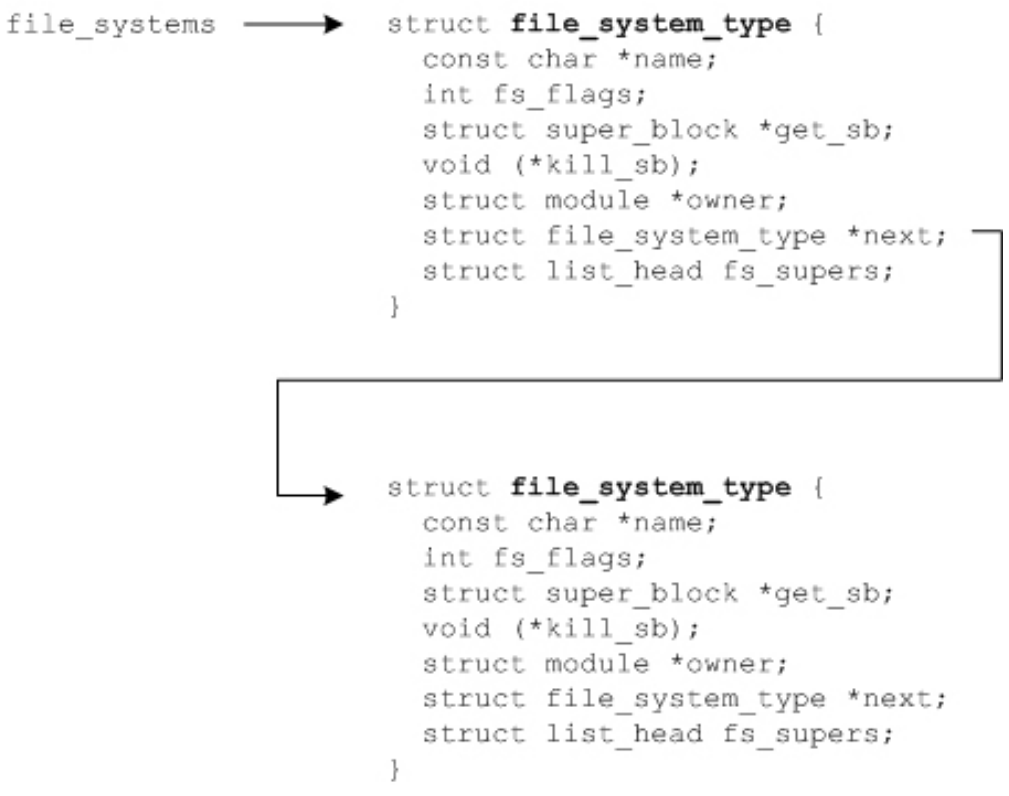
Silberschatz: Figura 12.04

Arquitetura em camadas



M. Tim Jones

Sistemas de arquivos implementados

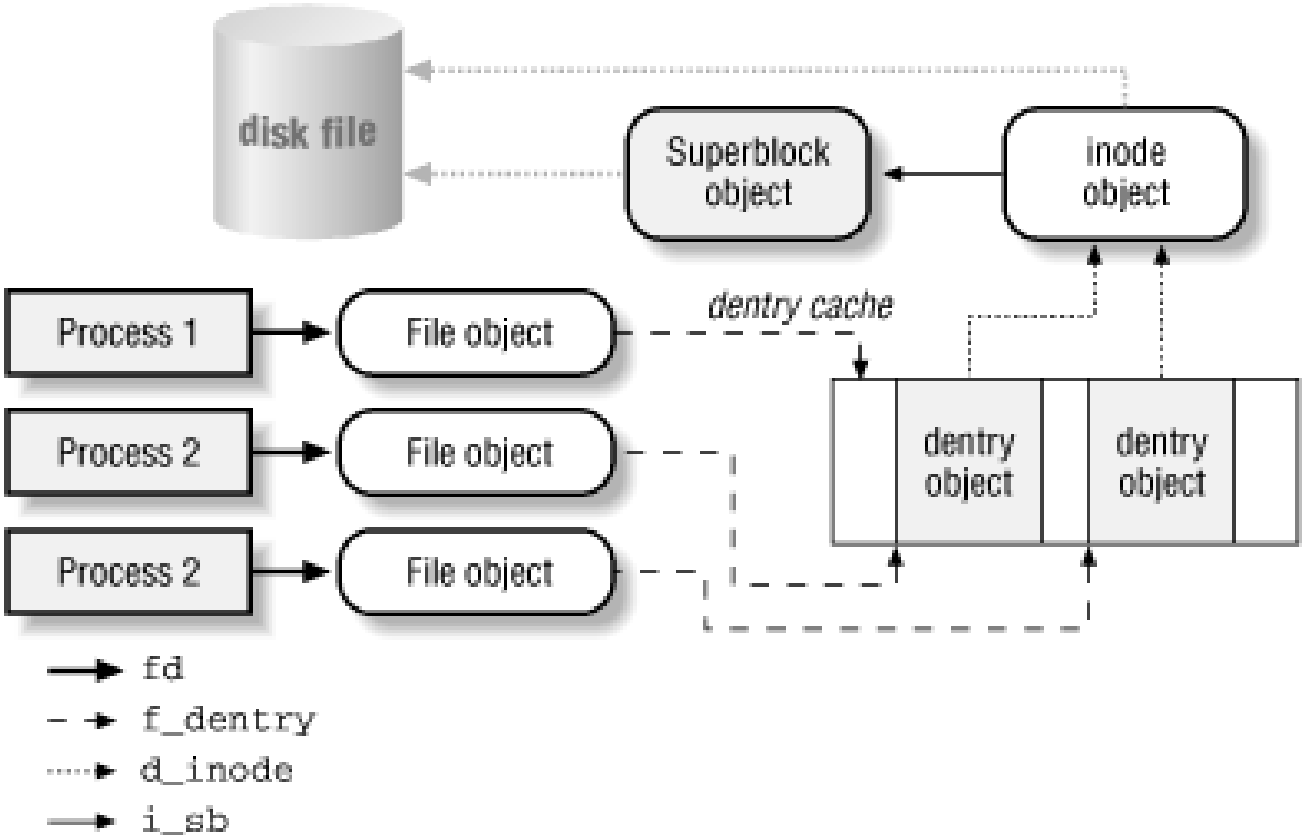


M. Tim Jones

- sistemas de arquivos podem ser adicionados e removidos dinamicamente
- *supported* $\neq$ *mounted*

```
cat /proc/filesystems
```

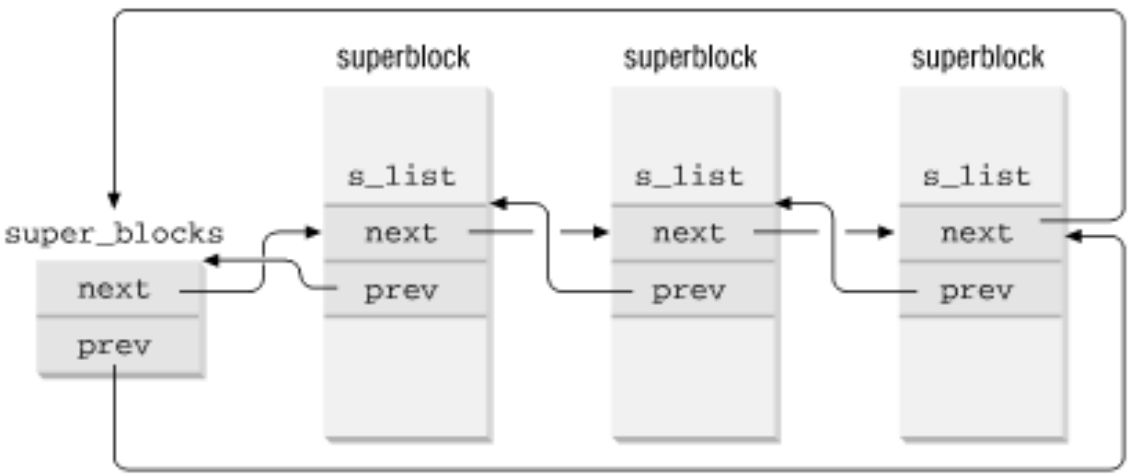
Objetos do VFS e interações com os processos



Understanding the Linux Kernel: Figura 12-2

Superblock

- Metadados sobre os sistemas de arquivos
- Estrutura existente em disco e em memória



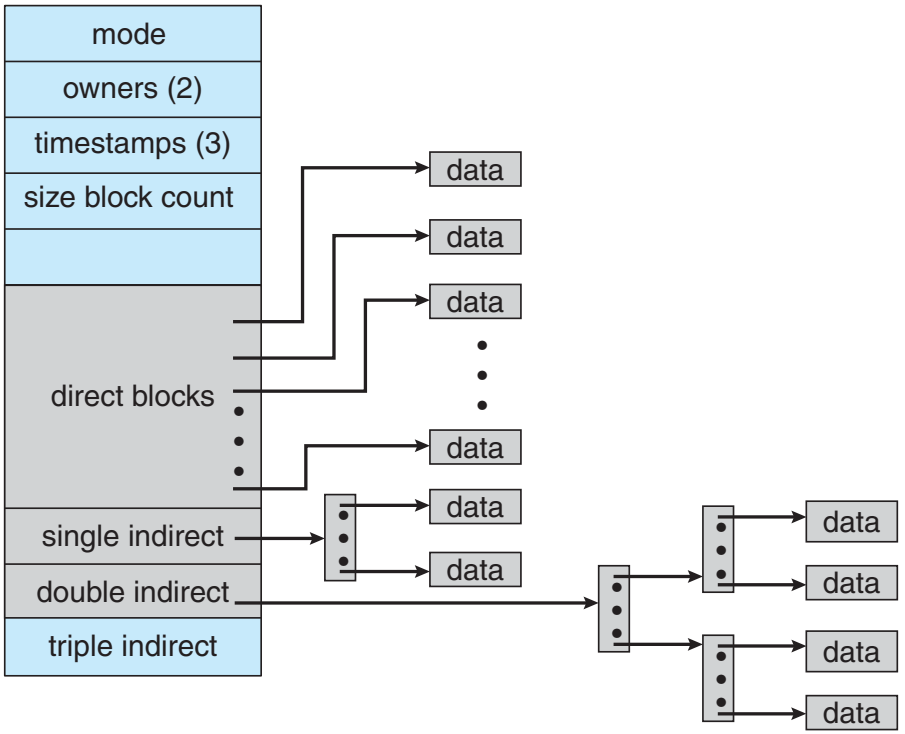
Understanding the Linux Kernel: Figura 12-2

Estruturas do superblock

```
struct super_block {
    struct list_head                    s_list;
    unsigned long                       s_blocksize;
    unsigned char                       s_blocksize_bits;
    struct file_system_type             *s_type;
    const struct super_operations       *s_op;
    unsigned long                       s_flags;
    struct dentry                       *s_root;
    struct rw_semaphore                 s_umount;
    struct list_head                    s_inodes;
    struct list_head                    s_dirty;
    struct list_head                    s_io;
    struct list_head                    s_more_io;
    struct list_head                    s_files;
    char                                s_id[32];
    void                                *s_fs_info;
    fmode_t                             s_mode;
    struct mutex            s_vfs_rename_mutex;
    ...
};
```

```
struct super_operations {
    struct inode*(*alloc_inode)(struct super_block *sb);
    void (*destroy_inode)(struct inode *);
    int (*write_inode) (struct inode *, int);
    void (*delete_inode) (struct inode *);
    void (*write_super) (struct super_block *);
    int (*sync_fs)(struct super_block *sb, int wait);
    int (*statfs) (struct dentry *, struct kstatfs *);
    int (*remount_fs)(struct super_block *, int *, char *);
    void (*umount_begin) (struct super_block *);
    int(*show_options)(struct seq_file *,struct vfsmount *);
    ...
};
```

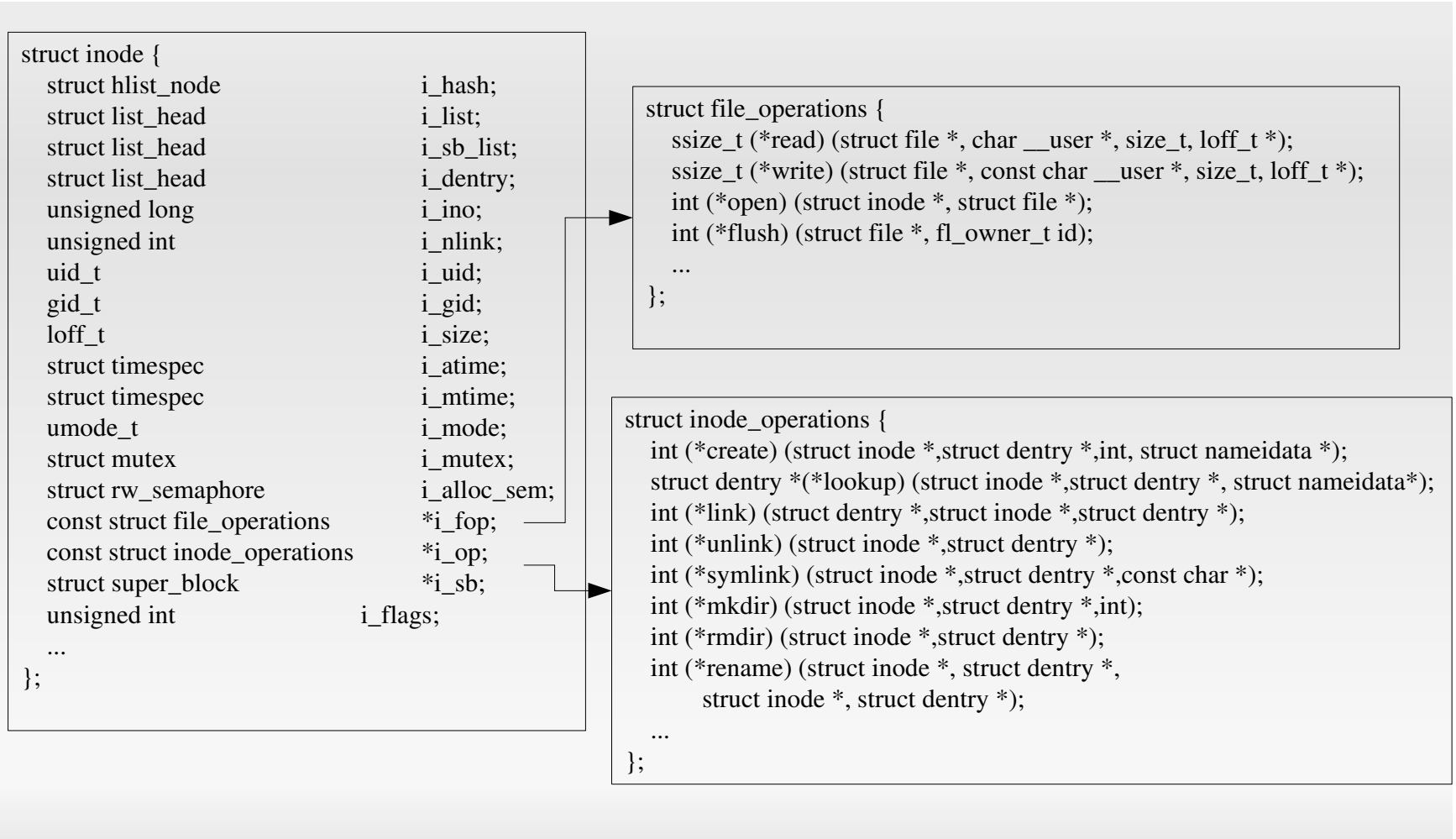
I-node



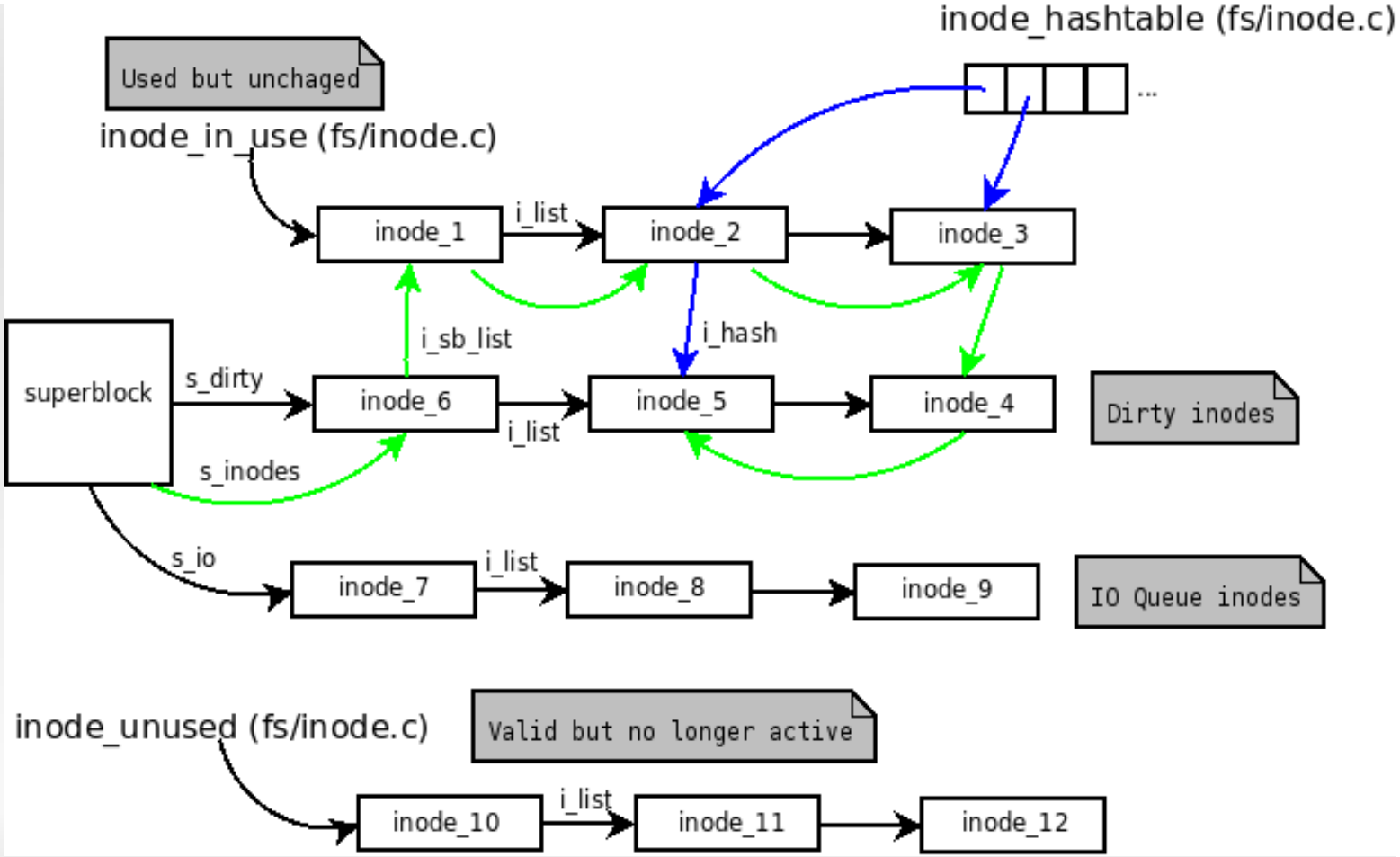
Silberschatz: Figura 12.09

- metadados de um arquivo ou diretório
- uma estrutura em disco e outra em memória

Estruturas do inode



Inode cache

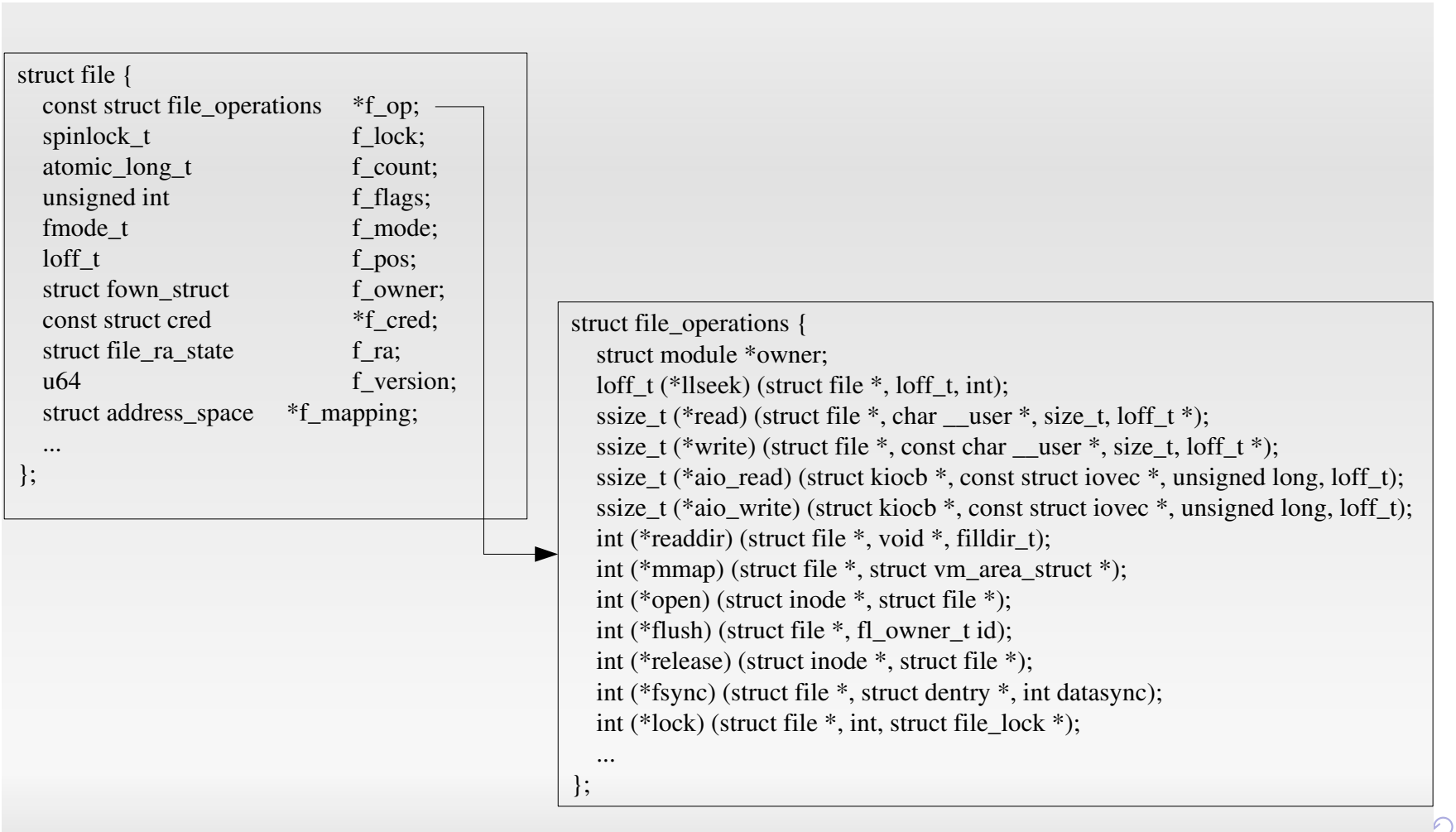

inode_hashtable (fs/inode.c)
Used but unchaged
inode_in_use (fs/inode.c)
inode_1
i_list
inode_2
inode_3
i_sb_list
i_hash
superblock
s_dirty
inode_6
inode_5
inode_4
Dirty inodes
i_list
s_inodes
s_io
inode_7
i_list
inode_8
inode_9
IO Queue inodes
inode_unused (fs/inode.c)
Valid but no longer active
inode_10
i_list
inode_11
inode_12

Dentry

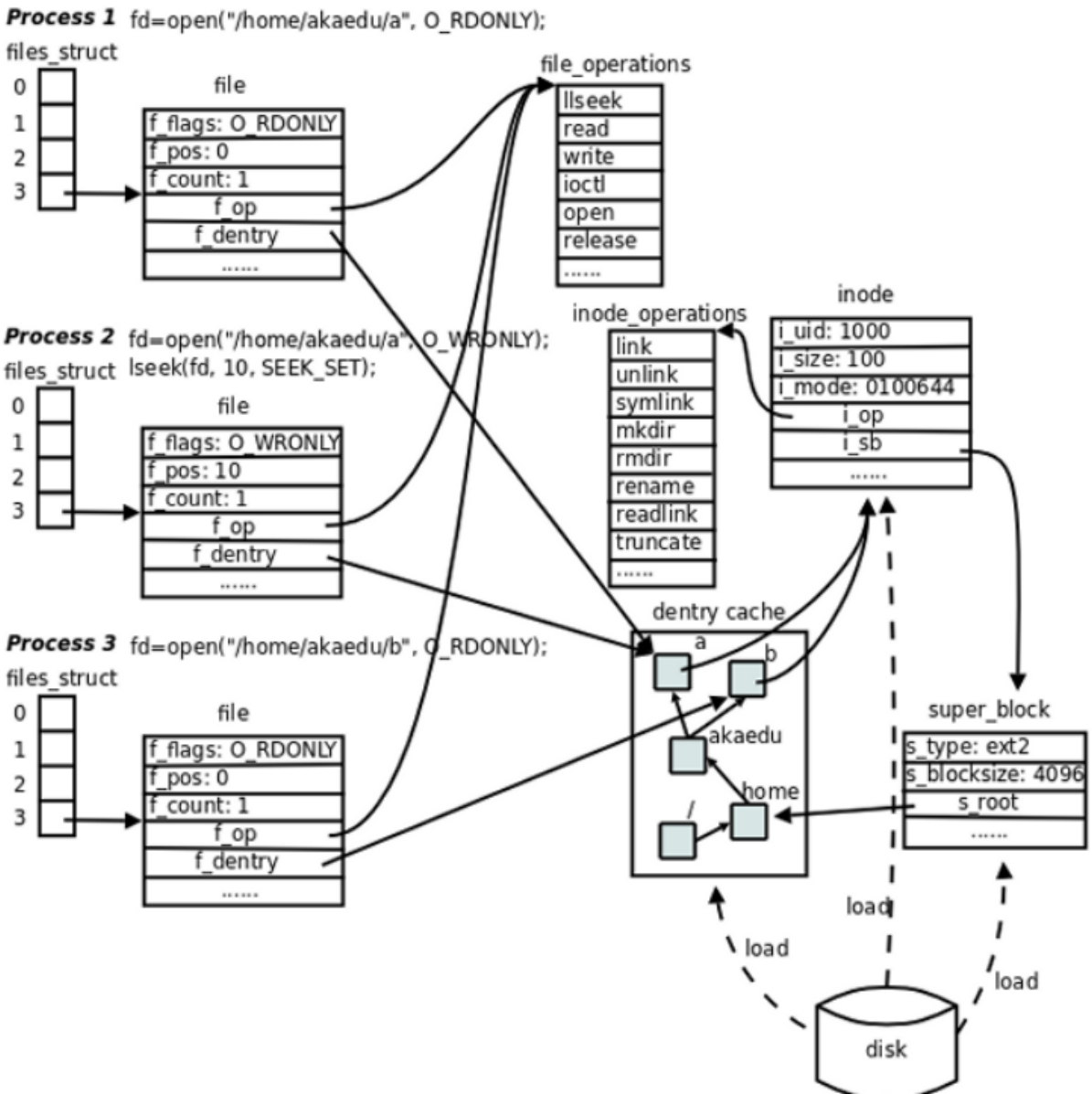
- Estrutura em memória
- Mapeamento de nomes e i-nodes
- Estados de um dentry no cache
  - Free: sem uso e sem informações válidas.
  - Unused: contém informações válidas.
  - In use: em uso e com informações válidas.
  - Negative: corresponde a um caminho inválido.

Estrutura do file object

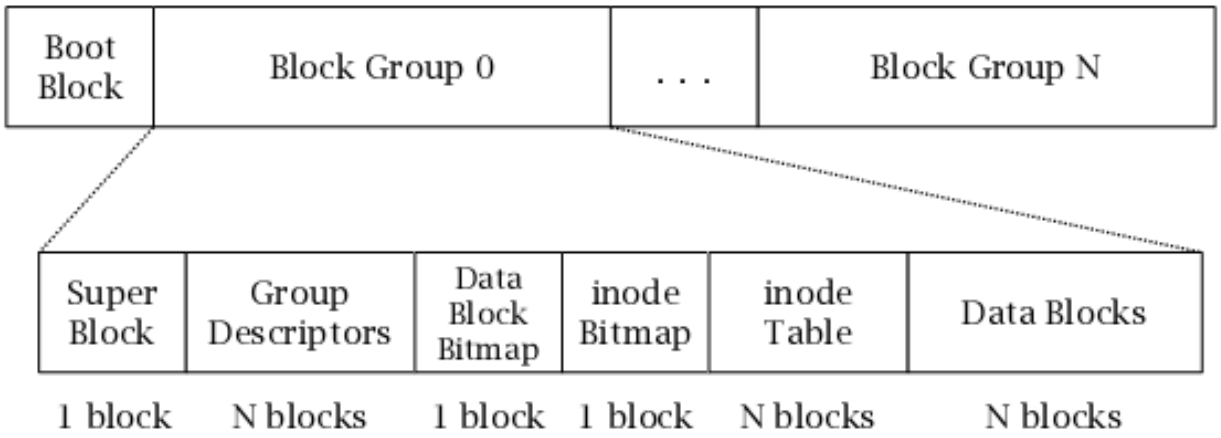
- Estrutura em memória
- Arquivo aberto



File descriptor table

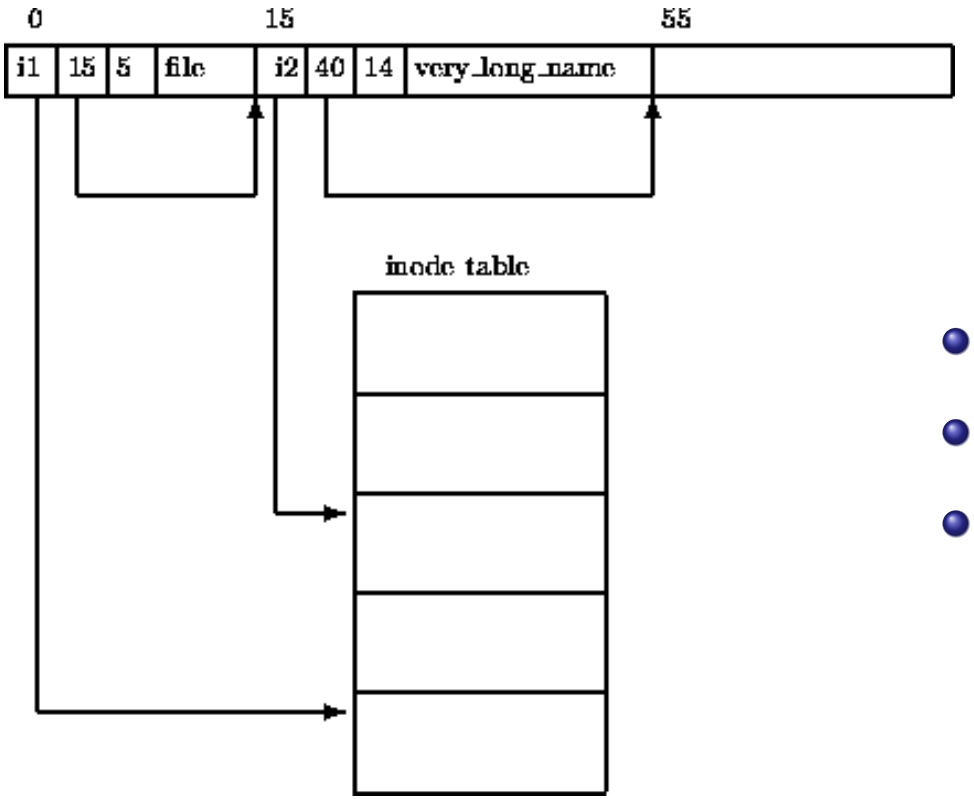


Estrutura do Ext2



- Super block: número de blocos no disco, tamanho dos blocos, ...
- Group descriptors: localização dos bitmaps inodes/blocos e da tabela de i-nodes, ..

Diretórios no Ext2



- inode
- name length
- name

Ext2 e crashes

- Operações como criação e remoção envolvem várias escritas em disco
- Um crash pode ocorrer durante o processo e
  - um bloco foi incluído no inode, mas não está marcado como em uso no bitmap
  - um bloco foi removido do inode, mas não está marcado como livre no bitmap
  - um bloco foi reutilizado, mas o inode original ainda aponta para ele.
  - um inode foi marcado como em uso, mas não há entrada em diretório apontando para ele.
  - ...

`fsck` e recuperação

Testando com `debugfs`

- Crie um sistema de arquivos ext2 com `mkfs.ext2`
- Utilize o `debugfs` para corromper este sistema
- Verifique se o `fsck` encontra os erros

Criando e fazendo verificação inicial

```
$ dd if=/dev/zero of=ext2.dmp bs=1k count=256
$ mkfs.ext2 ext2.dmp
$ fsck ext2.dmp
e2fsck 1.42.13 (17-May-2015)
ext2.dmp: clean, 11/32 files, 23/256 blocks
$ fsck -f ext2.dmp
e2fsck 1.42.13 (17-May-2015)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
ext2.dmp: 11/32 files (0.0% non-contiguous), 23/256 blocks
```

Alterando bitmap dos inodes

```
$ debugfs -w ext2.dmp
  debugfs: write a.txt a.txt
  debugfs:  ls
  2  (12) .    2  (12) ..    11  (20) lost+found    12  (980) a.
  debugfs: freei <12>
```

Como o `fsck` consegue corrigir o erro?

lost+found

```
 debugfs: write a.txt a.txt
 debugfs: modify_inode <12>
      Link count      [1]  2
 debugfs:  ls
  2  (12) .  2  (12) ..  11  (20) lost+found  12  (980) a.txt
 debugfs: rm a.txt
 debugfs: quit
$ fsck.ext2 -f ext2.dmp
```

- Por que o inode <12> seria colocado no diretório de "achados e perdidos"?

Journaling

- Anúncio das ações seguido de *commit record*
- redo log vs undo log
- ext3 adota redo log (delete?)

- teste `mkfs.ext3` e veja o espaço reservado para o *journal*
- Veja o tamanho do log no debugfs com logdump

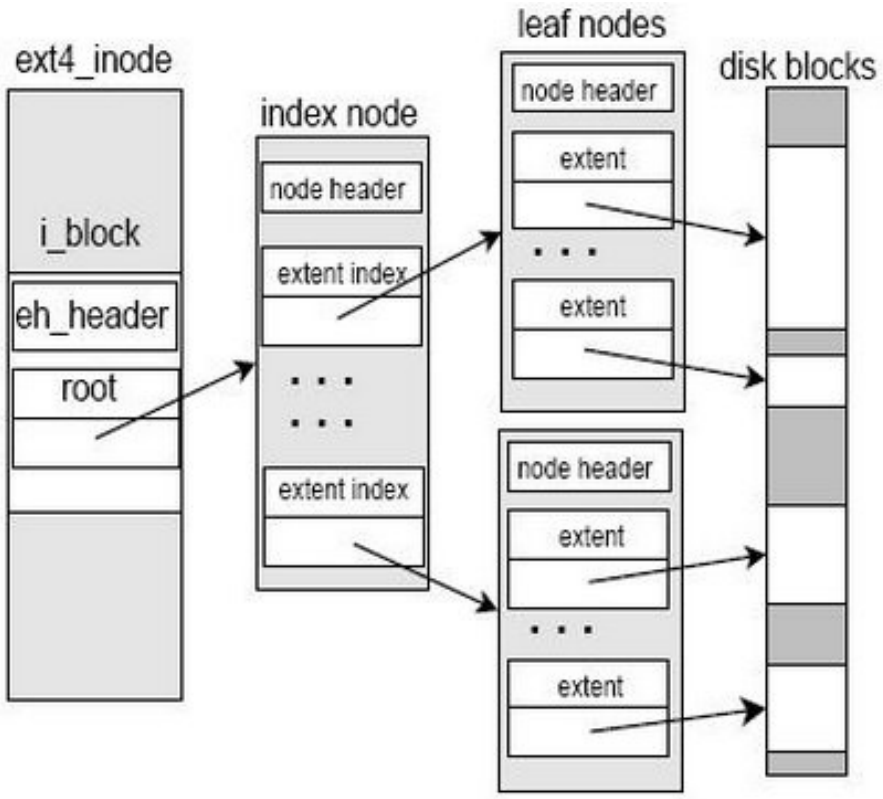
Modos de *journaling*

- **Journal**: Dados + metadados no log
- **Ordered**
  - Apenas metadados
  - Metadados só podem ser gravados no log **após** os dados estarem atualizados
  - Possibilidade de falha em caso de sobrescrita de blocos de dados
- **Writeback**: Sem restrição para ordem de gravação dos dados e metadados

Checkpointing

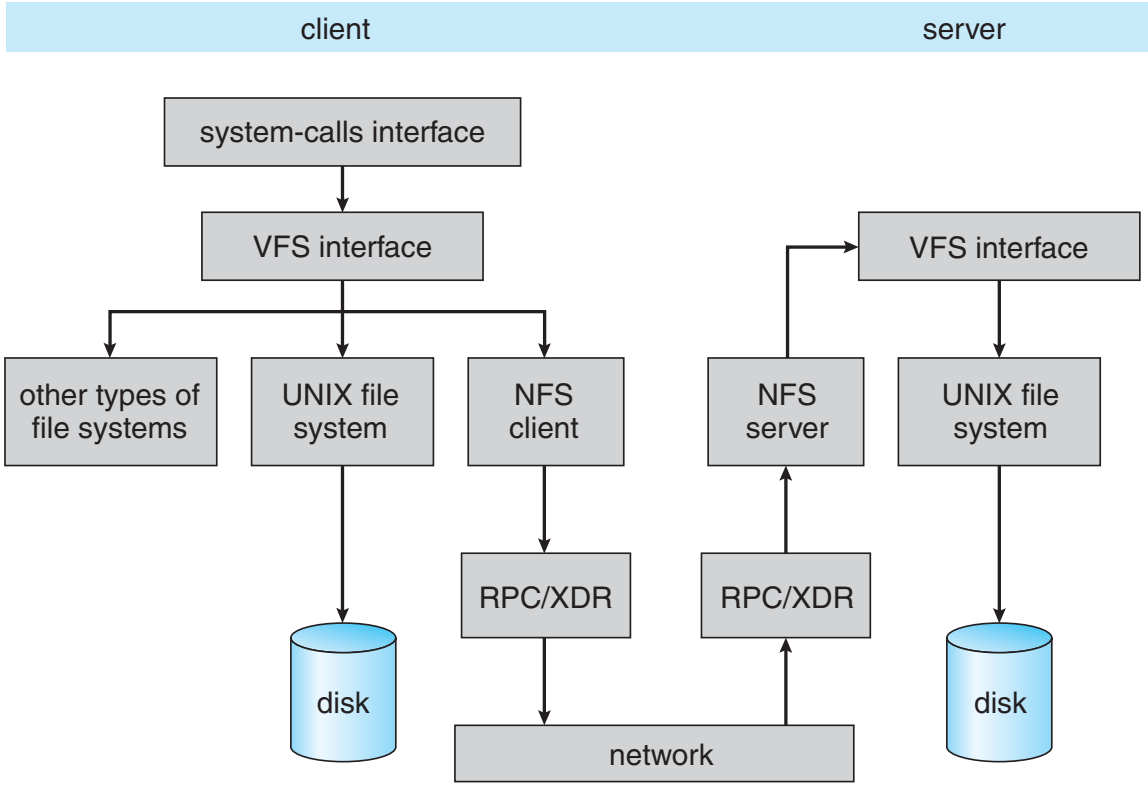
- Coleta de lixo das entradas no log
- Operações que já estão no disco podem ser descartadas
- Espaço pode ser reutilizado
- fsck poderia procurar inconsistências somente onde é necessário

ext4



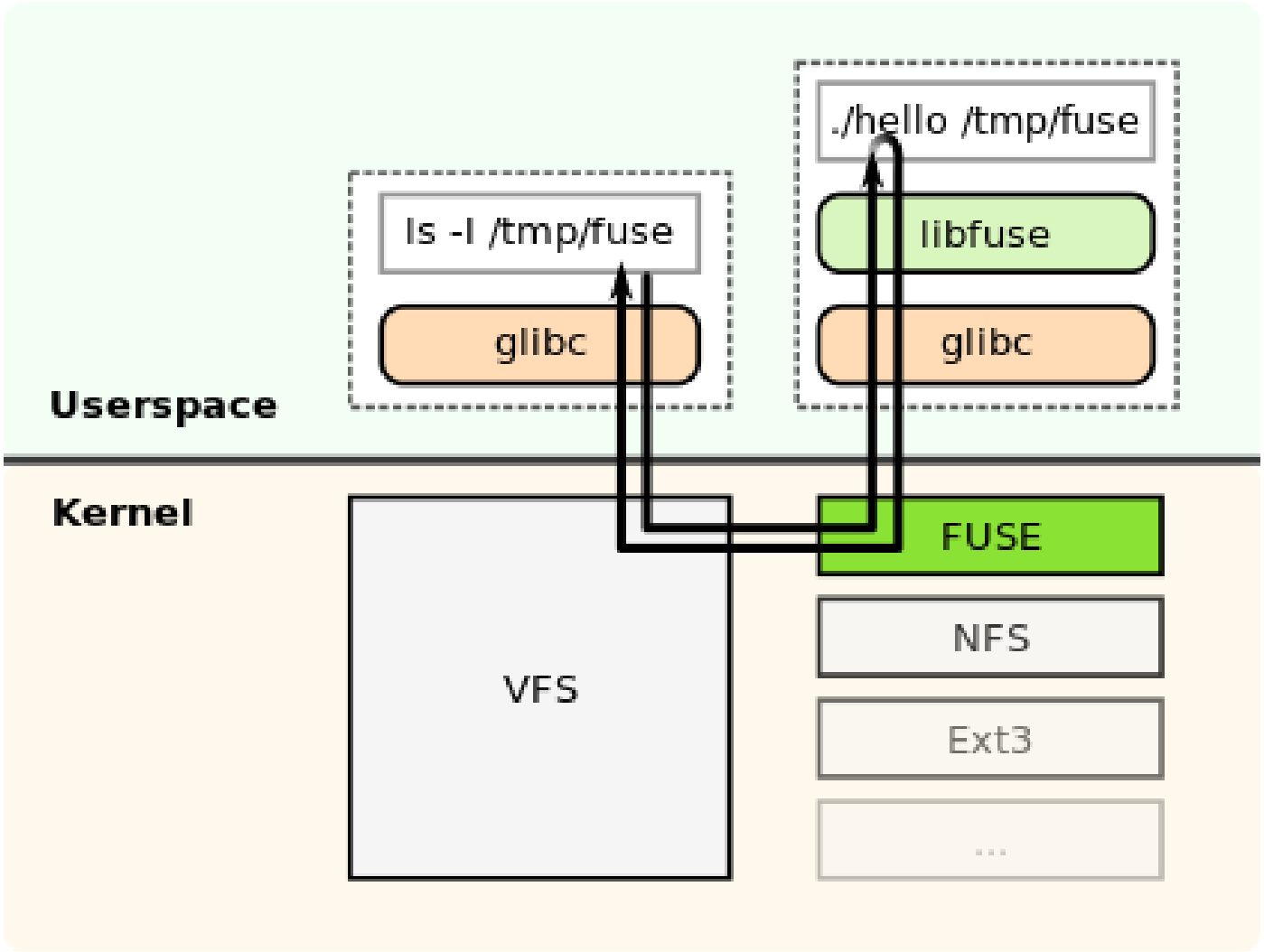
- *journal*
- arquivos maiores
- extents
- estrutura para diretórios: (HTree)

NFS



Silberschatz: Figura 12.15

FUSE File System



https://en.wikipedia.org/wiki/Filesystem_in_Userspace

Referências adicionais

- Anatomy of the Linux virtual file system switch, M. Tim Jones
- Understanding the Linux Kernel, 3rd Edition.
- Bruno Azevedo e Wilson Higashino. Virtual File System. MO806 - 2s2009