Sistemas de Operação / Fundamentos de Sistemas Operativos

o *sofs20* sistema de arquivo

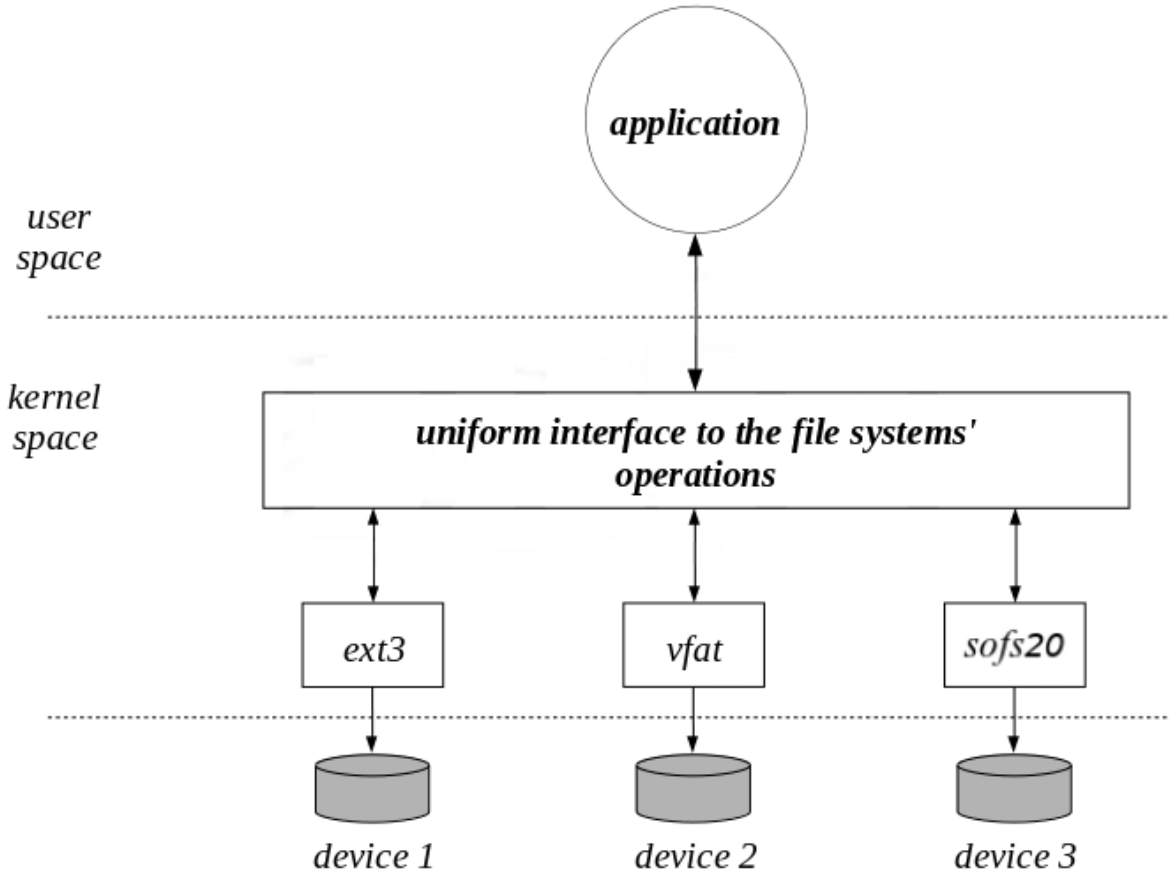
Artur Pereira < artur@ua.pt >

DETI / Universidade de Aveiro

Sumário

O sistema de arquivos FUSE

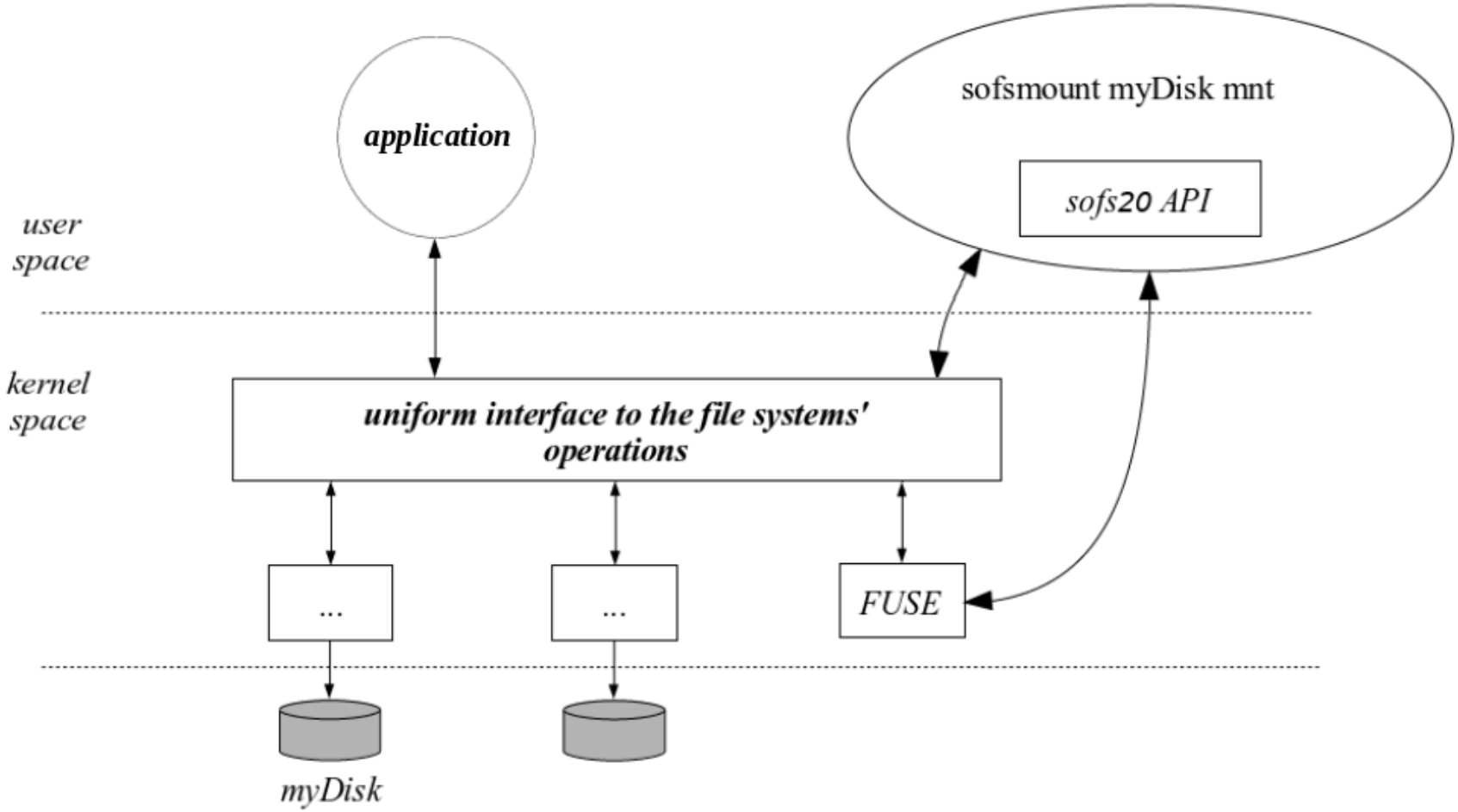
sofs20 como um módulo do kernel



- Problema de segurança: rodando no espaço do kernel
  - Código malicioso ou errado pode danificar o sistema

O sistema de arquivos FUSE

sofs20 como um módulo FUSE

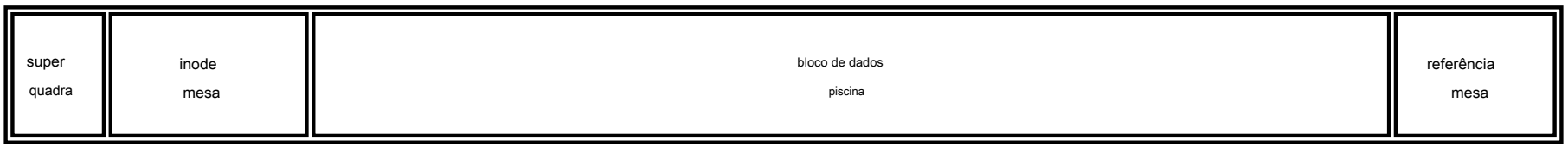


- Seguro: executando no espaço do usuário
  - Código malicioso ou errôneo afeta apenas o usuário

A arquitetura sofs20

Bloco de particionamento

- UMA sofs20 o disco é particionado / estruturado da seguinte maneira:
  - Os inodes são armazenados em um conjunto dedicado de blocos de tamanho fixo (tabela de inode)
  - Os blocos de dados também são armazenados em um conjunto dedicado de blocos de tamanho fixo (dados pool de blocos)
  - Um bloco, chamado superbloco, é usado para metadados gerais
  - A lista de inodes livres é armazenada no superbloco
  - A lista de blocos de dados livres é armazenada no superbloco e em um conjunto de blocos (tabela de referência)
  - Sequências de blocos usados por inodes são armazenadas nos próprios inodes e em blocos de dados alocados para esse fim



A arquitetura sofs20

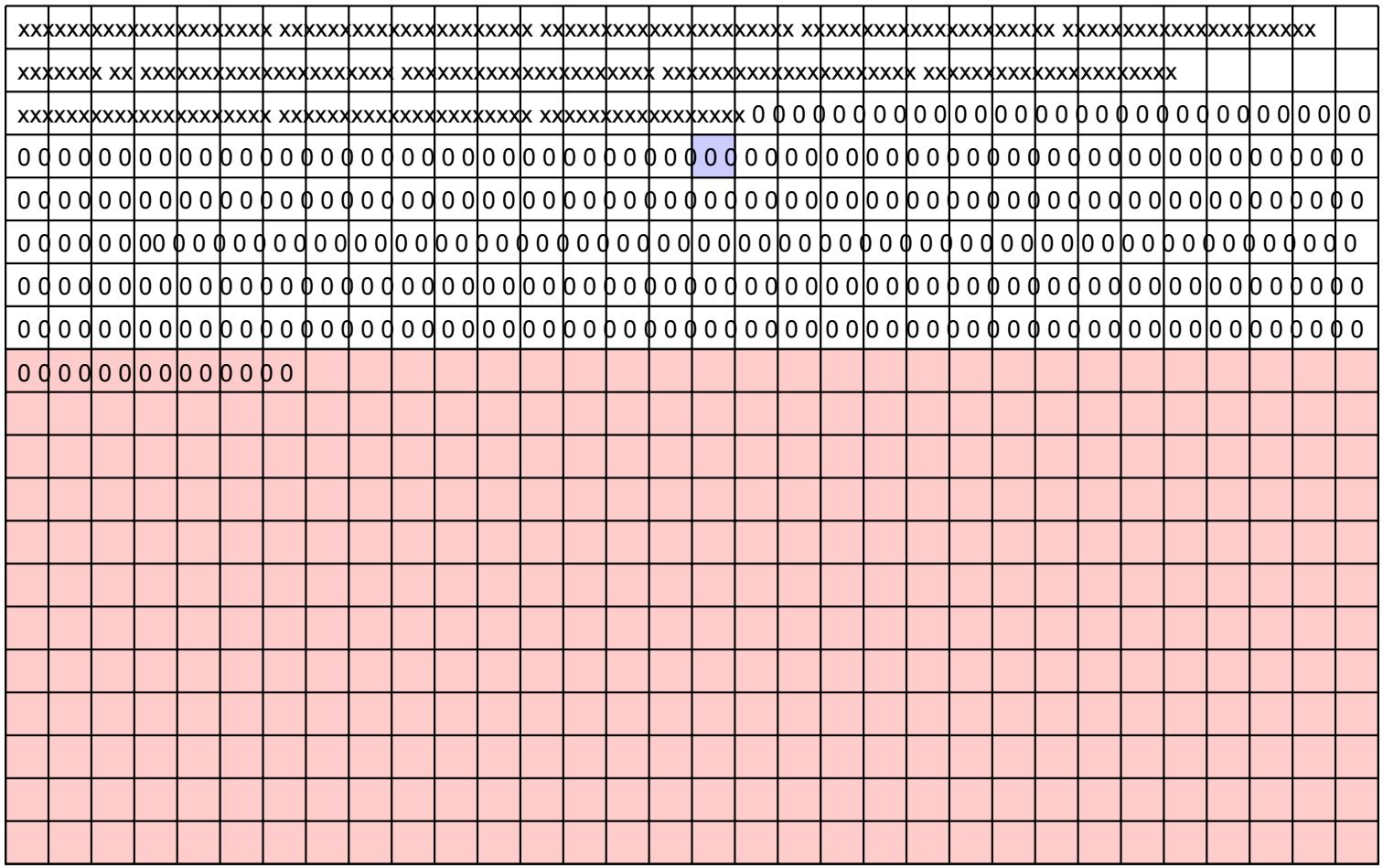
Lista de inodes grátis

- Baseado em um mapa de bits
  - há uma correspondência um a um entre bits no mapa e inodes na tabela de inode, incluindo o inode 0
  - 0 ⇒ inode é grátis; 1 ⇒ inode está em uso
- O bitmap é armazenado no superbloco (ibitmap campo)
  - visto como uma matriz de palavras de 32 bits, com tamanho fixo
  - o inode 0 é representado pelo bit 0 da palavra 0 e assim por diante
  - bits não utilizados são mantidos em 0
- livre operação de integração:
  - limpe o inode e coloque o bit correspondente em 0
- alocar operação de operação:
  - procure um bit em 0, coloque-o em 1 e inicialize o inode correspondente
  - a pesquisa deve começar na posição circular ao lado do último inode alocado (iidx campo)

A arquitetura sofs20

Lista de inodes livres (2)

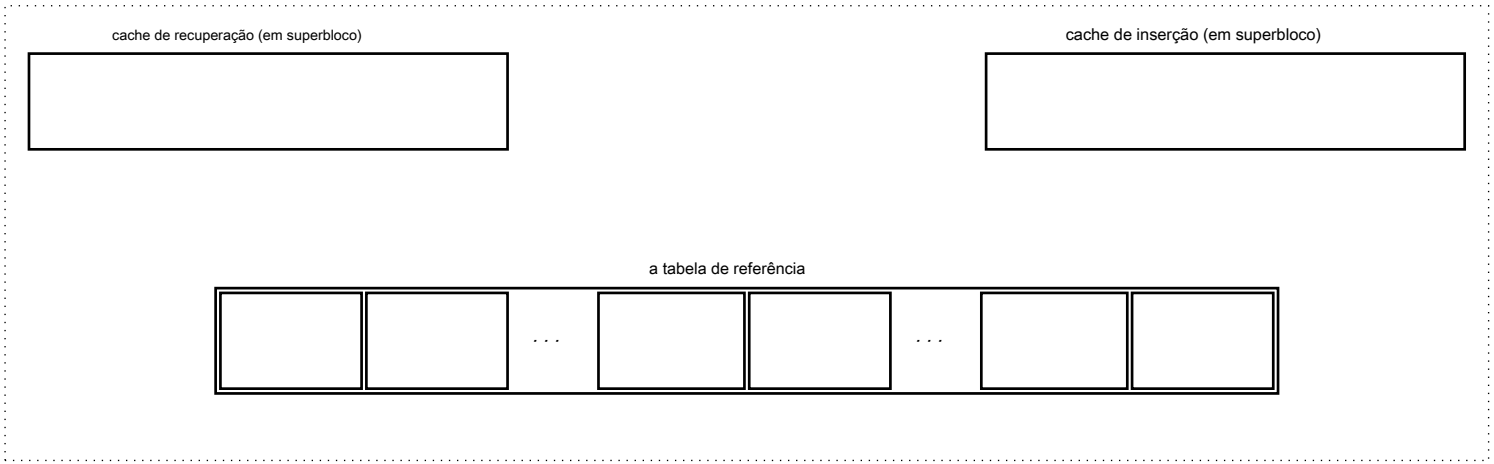
- Um possível estado do mapa de bits



A arquitetura sofs20

Lista de blocos de dados livres

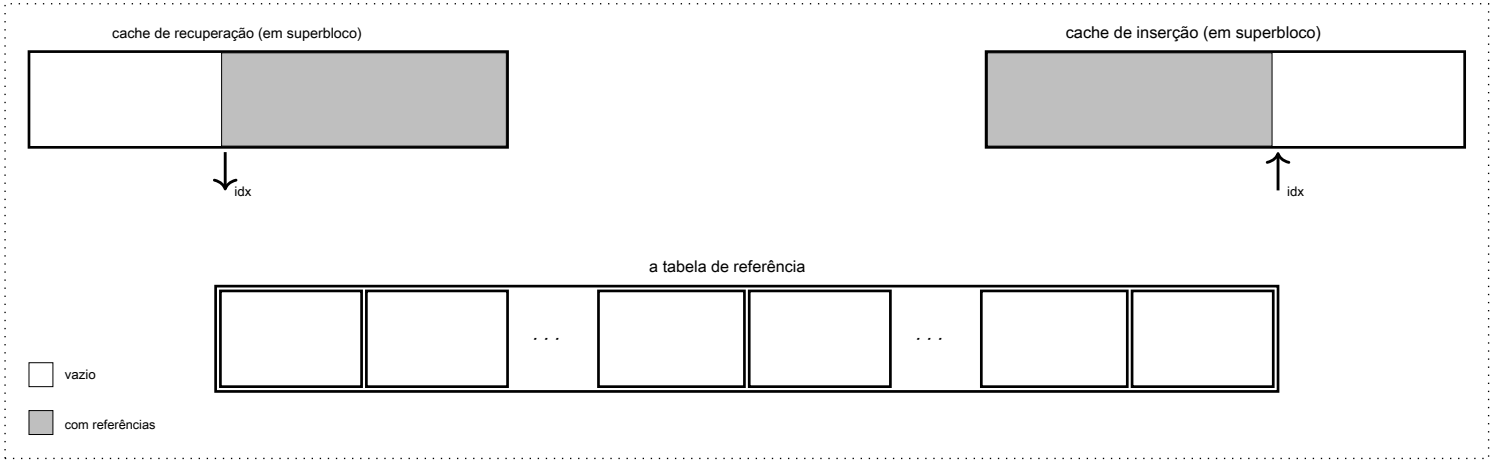
- Baseado em um FIFO
  - o primeiro bloco de dados livre a ser usado é o mais antigo da lista
  - uma referência é uma palavra de 32 bits, sendo 0xFFFFFFFF a referência nula
- A lista é armazenada no superbloco e blocos dedicados
  - a primeira sub-sequência ordenada é armazenada no cache de recuperação , representando as referências mais antigas da lista
  - a última subseqüência ordenada é armazenada no cache de inserção , representando as referências mais recentes na lista
  - as referências encomendadas restantes são armazenadas no tabela de referência



A arquitetura sofs20

Lista de blocos de dados livres

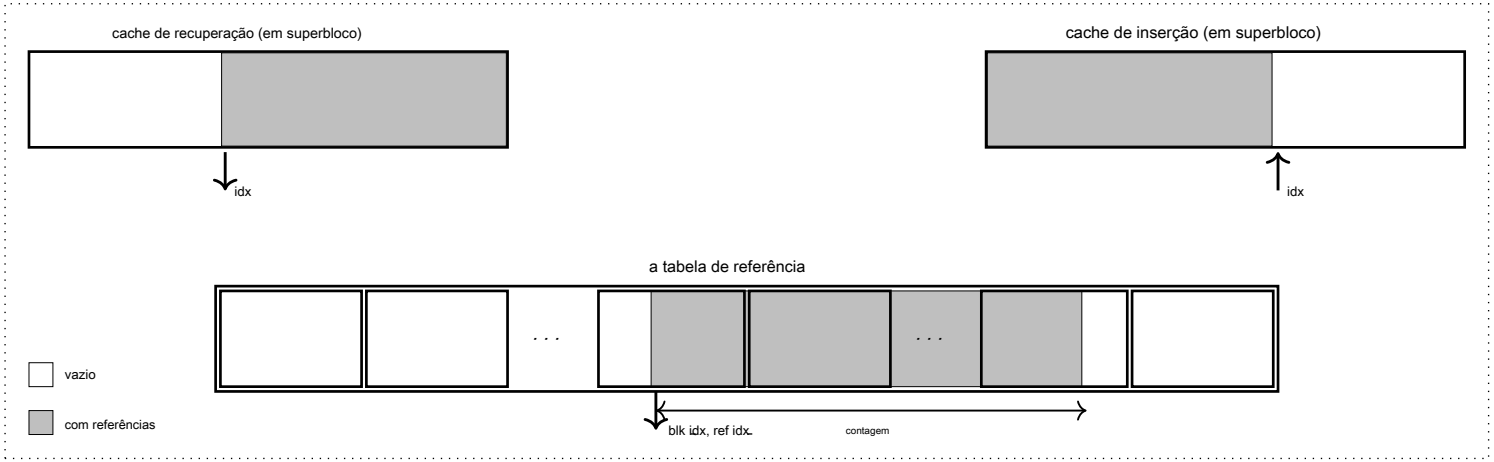
- cache de recuperação
  - este cache pode estar parcialmente vazio, o que significa que algumas referências (necessariamente no início) já foram recuperadas
  - um índice (idx na figura) aponta para a primeira célula com uma referência
- cache de inserção
  - este cache pode estar parcialmente preenchido, o que significa que algumas referências (no início) já foram inseridas
  - um índice (idx na figura) aponta para a primeira célula vazia



A arquitetura sofs20

Lista de blocos de dados livres

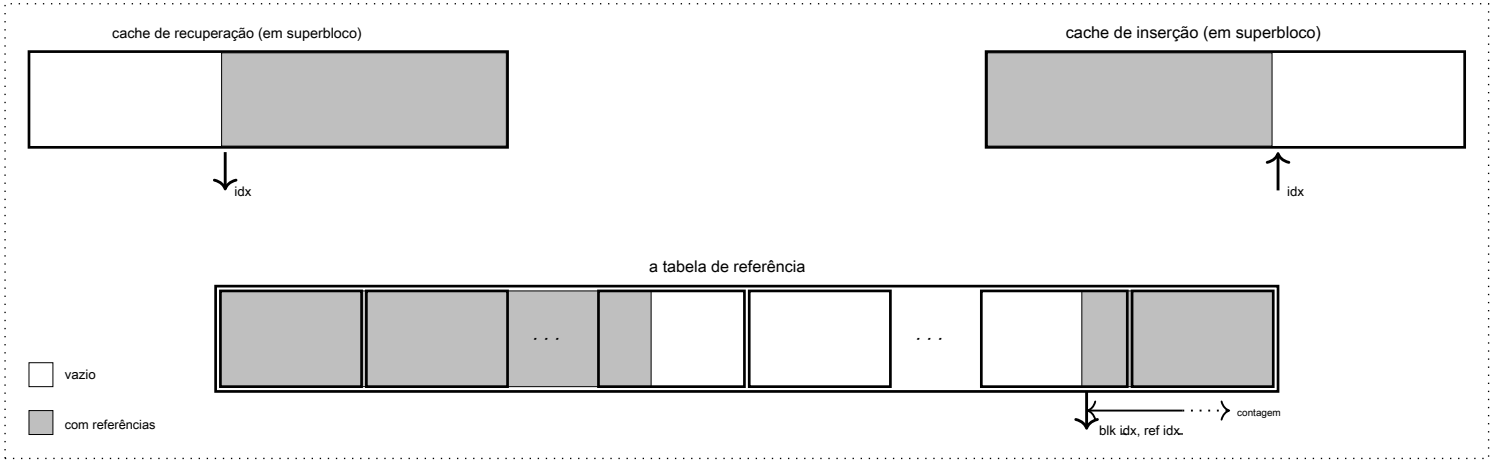
- tabela de referência
  - Dois campos no superbloco (rt start e tamanho rt) delimite a região do disco com a tabela de referência
  - Em geral, apenas parte desta tabela será preenchida, conforme ilustrado na figura (área cinza)
  - Outro campo (reftable), composto por três subcampos (blk idx, ref idx, e contagem), armazena o estado da tabela de referência



A arquitetura sofs20

Lista de blocos de dados livres

- tabela de referência
  - Dois campos no superbloco (rt start e tamanho rt) delimite a região do disco com a tabela de referência
  - Em geral, apenas parte desta tabela será preenchida, conforme ilustrado na figura (área cinza)
  - Outro campo (reftable), composto por três subcampos (blk idx, ref idx, e contagem), armazena o estado da tabela de referência
  - É gerido de forma circular, o que significa que a posição a seguir à última é o índice 0.
    - assim, a região ocupada pode ser semelhante à da próxima figura (área cinza).



A arquitetura sofs20

Sequência de blocos de um arquivo (1)

- Os blocos não são compartilháveis entre os arquivos
  - um bloco em uso b (alonga-se para um único arquivo
- O número de blocos exigidos por) um arquivo para armazenar suas informações é fornecido por

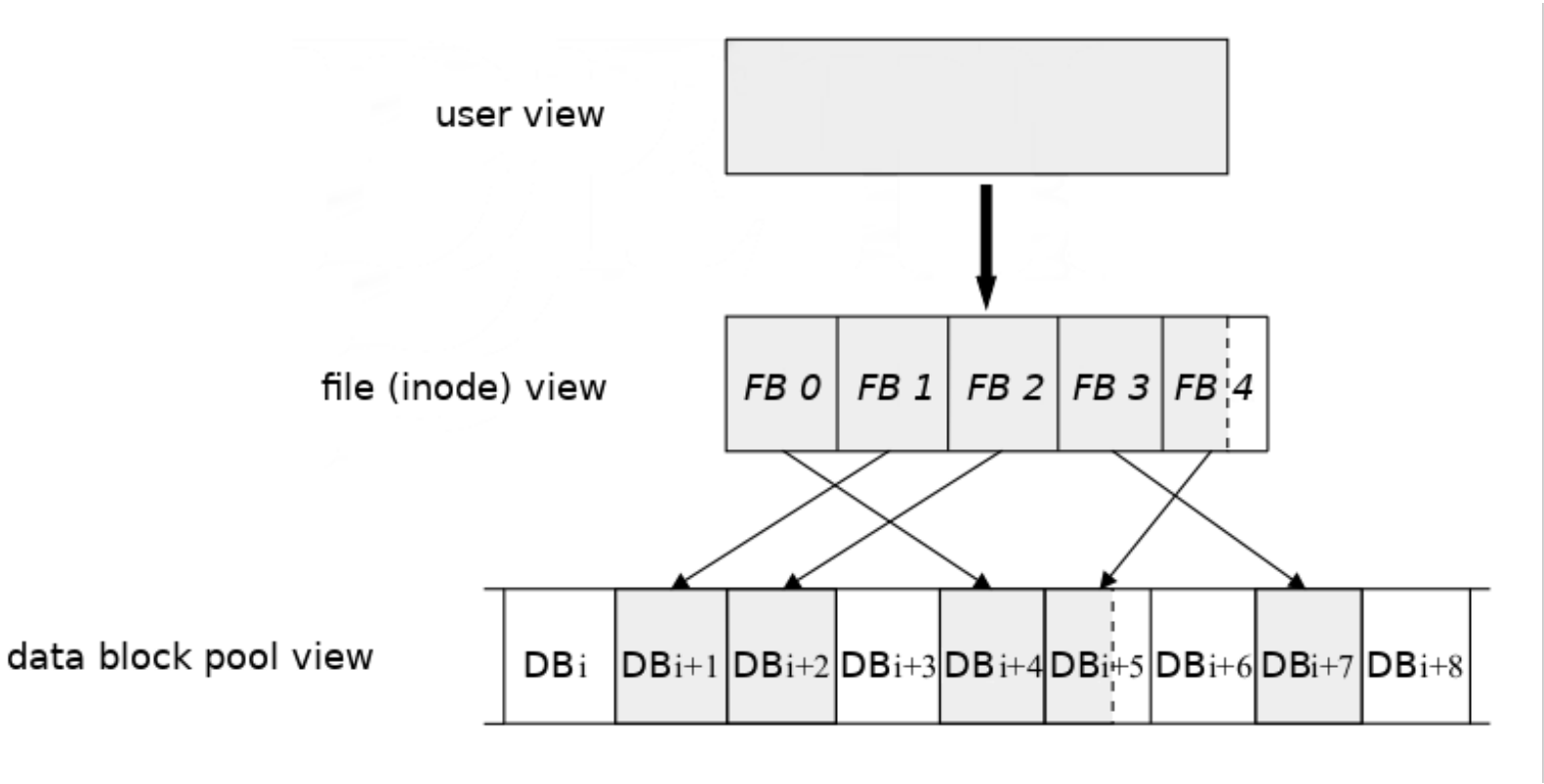
$$N_b = \text{arredondar para cima} \quad \frac{\text{Tamanho}}{\text{Tamanho do bloco}}$$

- N_b pode ser muito grande
  - se o tamanho do bloco for 1024 bytes, um arquivo de 2 GByte precisa de 2 MBlocks
- N_b pode ser muito pequeno
  - um arquivo de 0 bytes não precisa de blocos para dados
- É impraticável que todos os blocos usados por um arquivo sejam contíguos no disco
- O acesso aos dados do arquivo em geral não é sequencial, mas sim aleatório
- portanto uma estrutura de dados flexível, tanto em tamanho quanto em localização, é necessária

A arquitetura sofs20

Sequência de blocos de um arquivo (2)

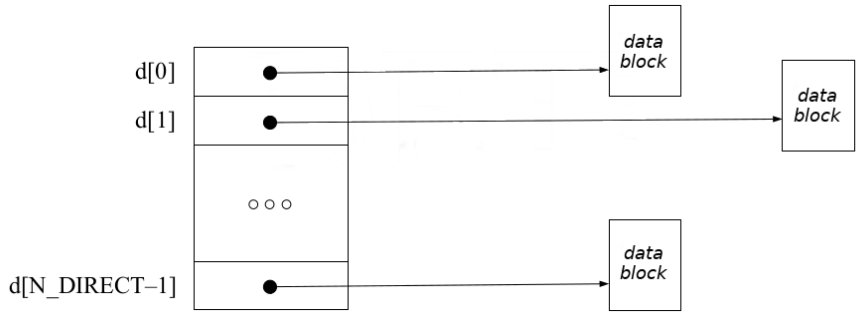
- O programador vê um arquivo como um continuum de bytes
- O inode vê uma sequência de blocos (bloco de arquivo)
- Os blocos de dados são, em geral, espalhados ao longo do pool de blocos de dados



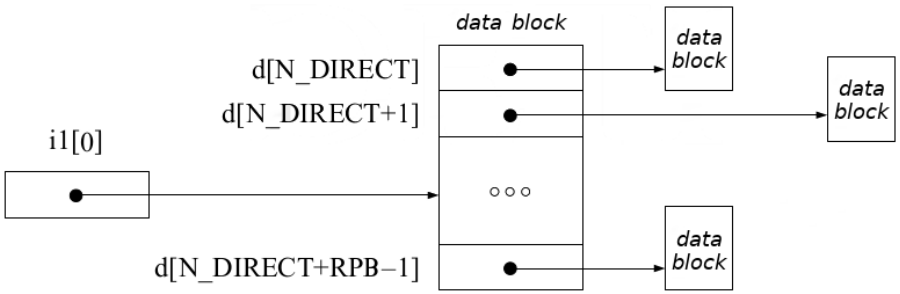
A arquitetura sofs20

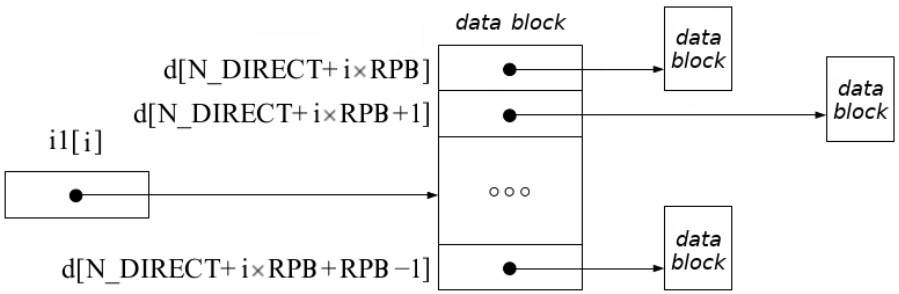
Sequência de blocos de um arquivo (3)

- Como a sequência de (referências a) blocos é armazenada?
- As primeiras referências são armazenadas diretamente no inode



- Então, campo inode i1 [.] aponta para blocos de dados com referências

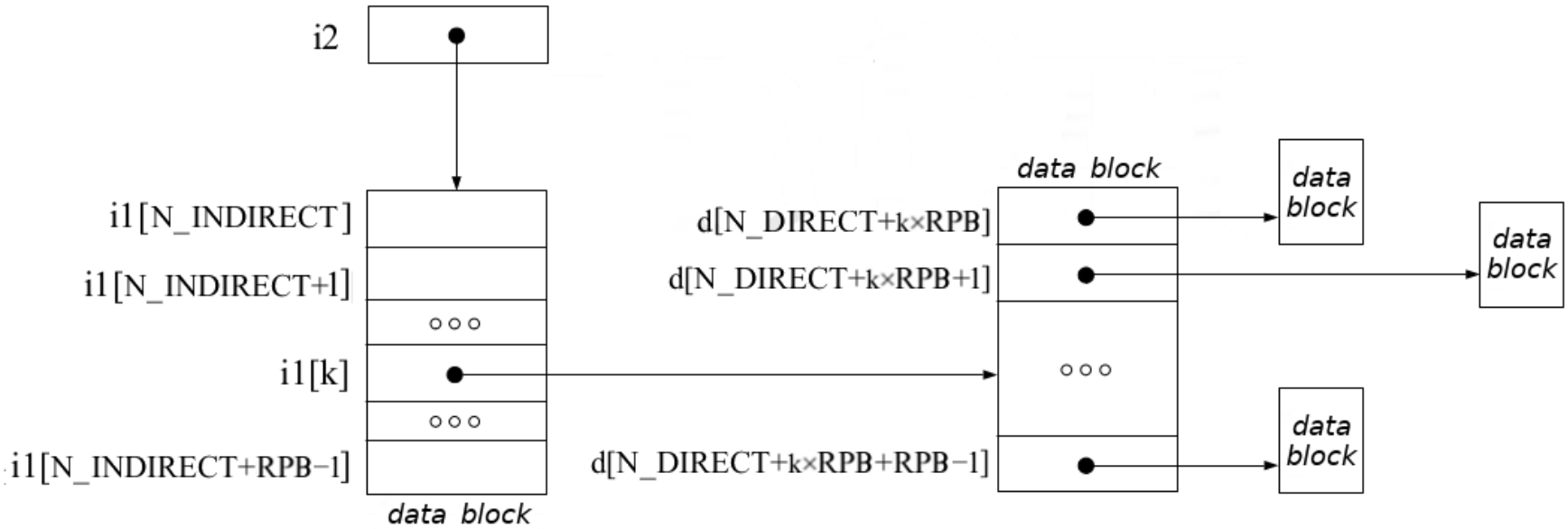




A arquitetura sofs20

Sequência de blocos de um arquivo (4)

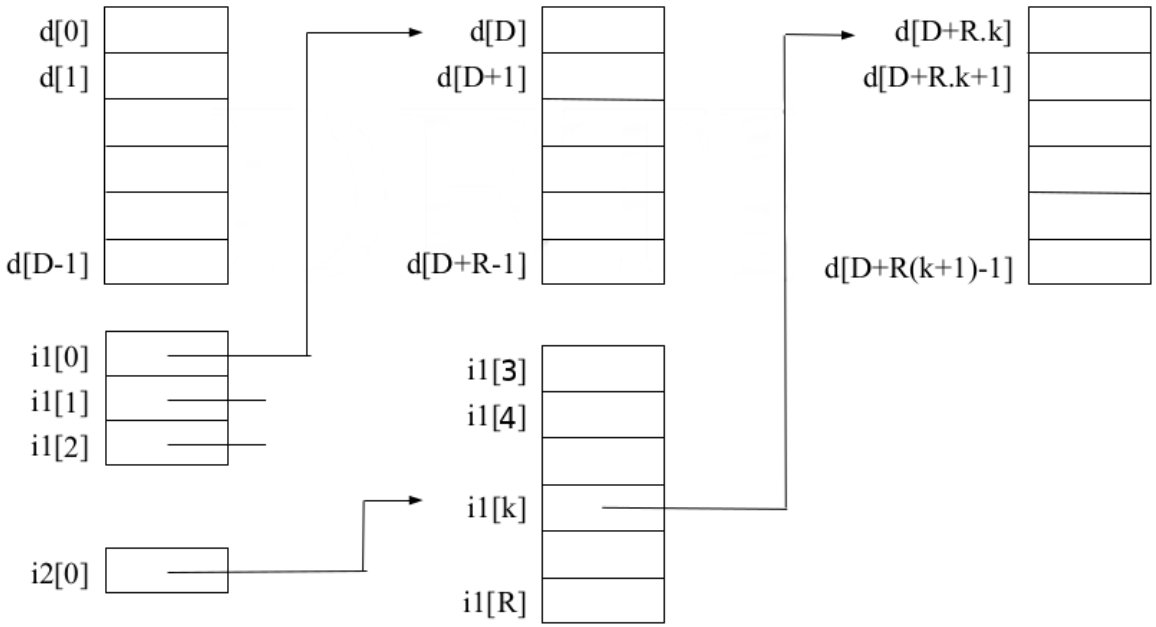
- Finalmente, campo inode i2 apontar para um bloco de dados que estende i1



A arquitetura sofs20

Sequência de blocos de um arquivo (5)

- Colocando todos juntos

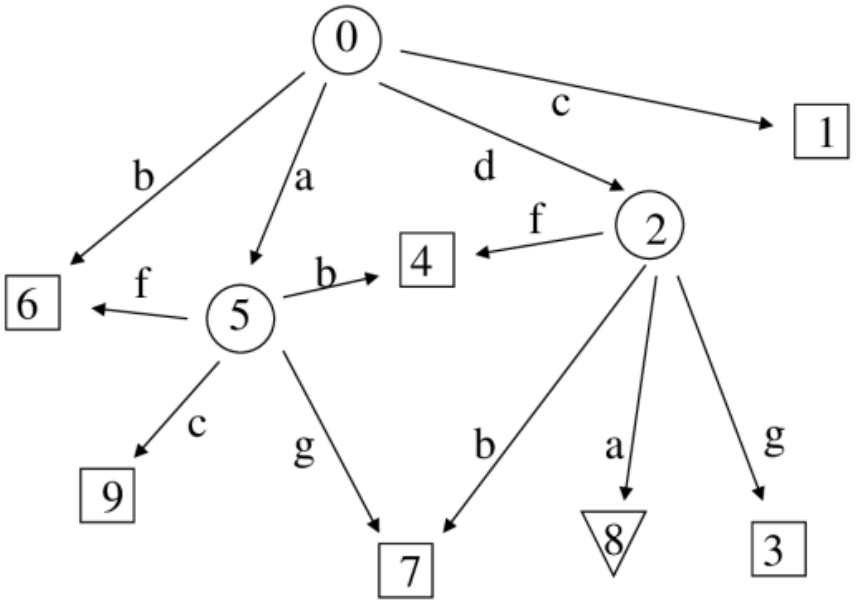


- Um arquivo pode conter "furos"
  - correspondendo a referências nulas cobertas pelo tamanho
  - e representando blocos de zeros

A arquitetura sofs20

Diretórios e entradas de diretório

- Um diretório é apenas uma lista de entradas de diretório
- Uma entrada de diretório é um par que associa um nome a um inode



directory

regular file

shortcut

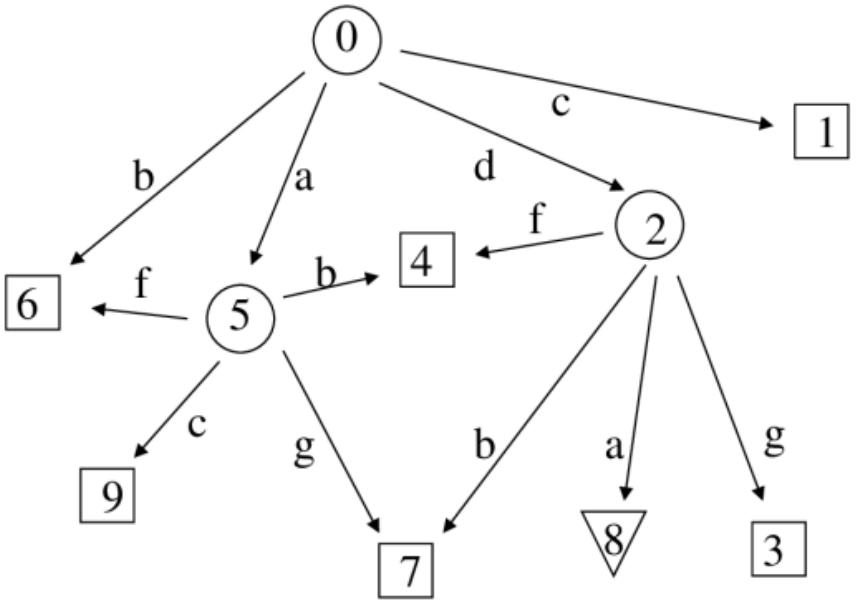
- O conteúdo do diretório "/" (inode 0) é:

nome	inode
.	0
. .	0
c	1
d	2
uma	5
b	6

A arquitetura sofs20

Diretórios e entradas de diretório

- Um diretório é apenas uma lista de entradas de diretório
- Uma entrada de diretório é um par que associa um nome a um inode



directory

regular file

shortcut

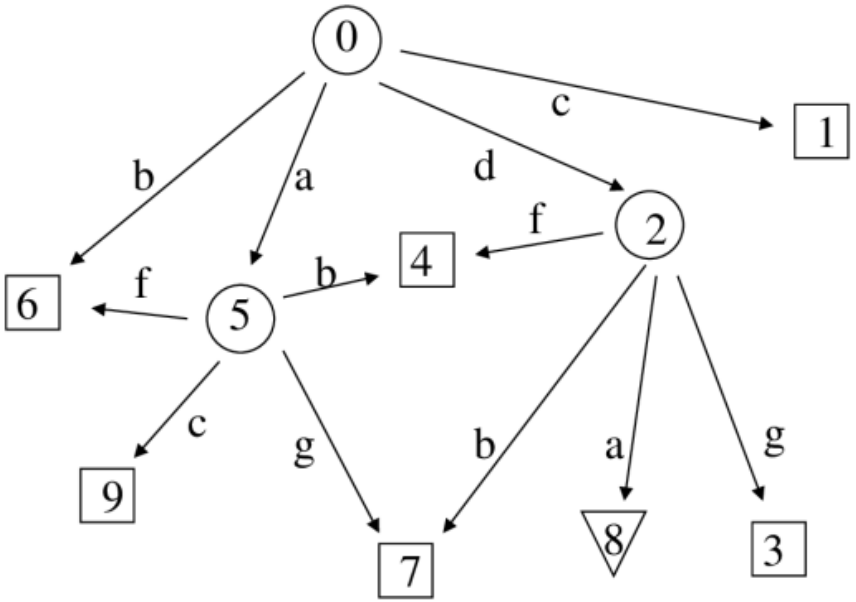
- O conteúdo do diretório "/ a /" (inode 5) é:

nome	inode
.	5
. .	0
c	9
b	4
f	6
g	7

A arquitetura sofs20

Diretórios e entradas de diretório

- Um diretório é apenas uma lista de entradas de diretório
- Uma entrada de diretório é um par que associa um nome a um inode



- O conteúdo do diretório "/ d /" (inode 2) é:

nome	inode
.	2
. .	0
uma	8
g	3
f	4
b	7

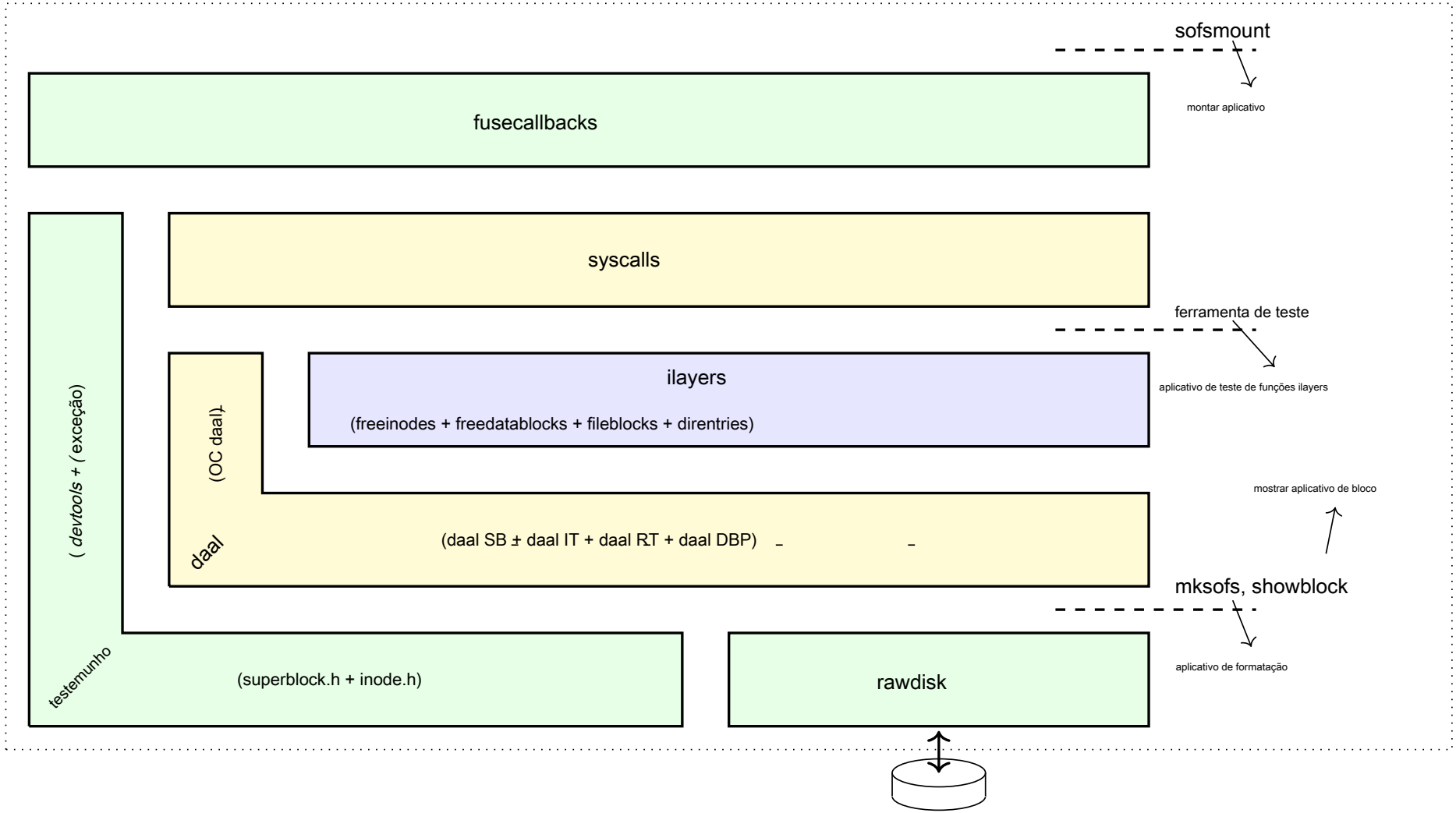
directory

regular file

shortcut

A estrutura de código sofs20

Camadas e ferramentas da biblioteca



A estrutura de código sofs20

Ferramentas

- Código preparado para usar a ferramenta de construção cmake
  - Precisa preparar cmake
  - Pode escolher entre faço e ninja

- Código preparado para usar a ferramenta de documentação doxygen
  - Con fi gurado para usar somente. h arquivos
  - Con fi gurado para gerar apenas páginas html

- ferramentas sofs20 :
  - showblock - mostra um ou mais blocos de um disco sofs20
  - ferramenta de teste - funções de chamada das camadas intermediárias

A ferramenta de formatação

mksofs

- Objetivo :
  - Preencha os blocos de um disco bruto para torná-lo um sistema de arquivos sofs20

- Estado de um disco recém-formatado :
  - Inode 0 é usado pelo diretório raiz
  - Os dados do diretório raiz são armazenados no bloco de dados número 0
  - Um conjunto de outras regras também deve ser observado
    - eles são indicados na documentação

- Aproximação :
  - O código foi decomposto em 6 funções auxiliares
  - A fonte do código principal é fornecida