Sistemas de Operação / Fundamentos de Sistemas Operativos

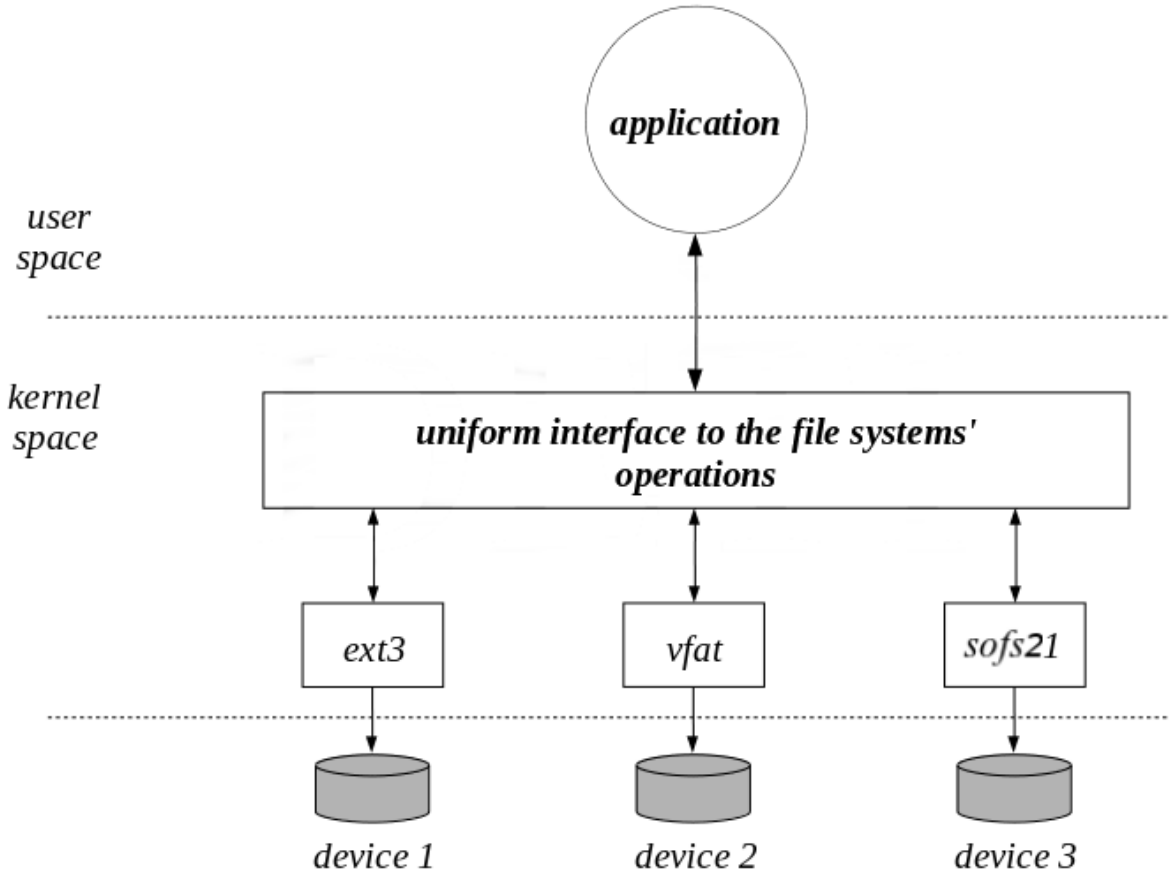
The *sofs21* file system

Artur Pereira <artur@ua.pt>

DETI / Universidade de Aveiro

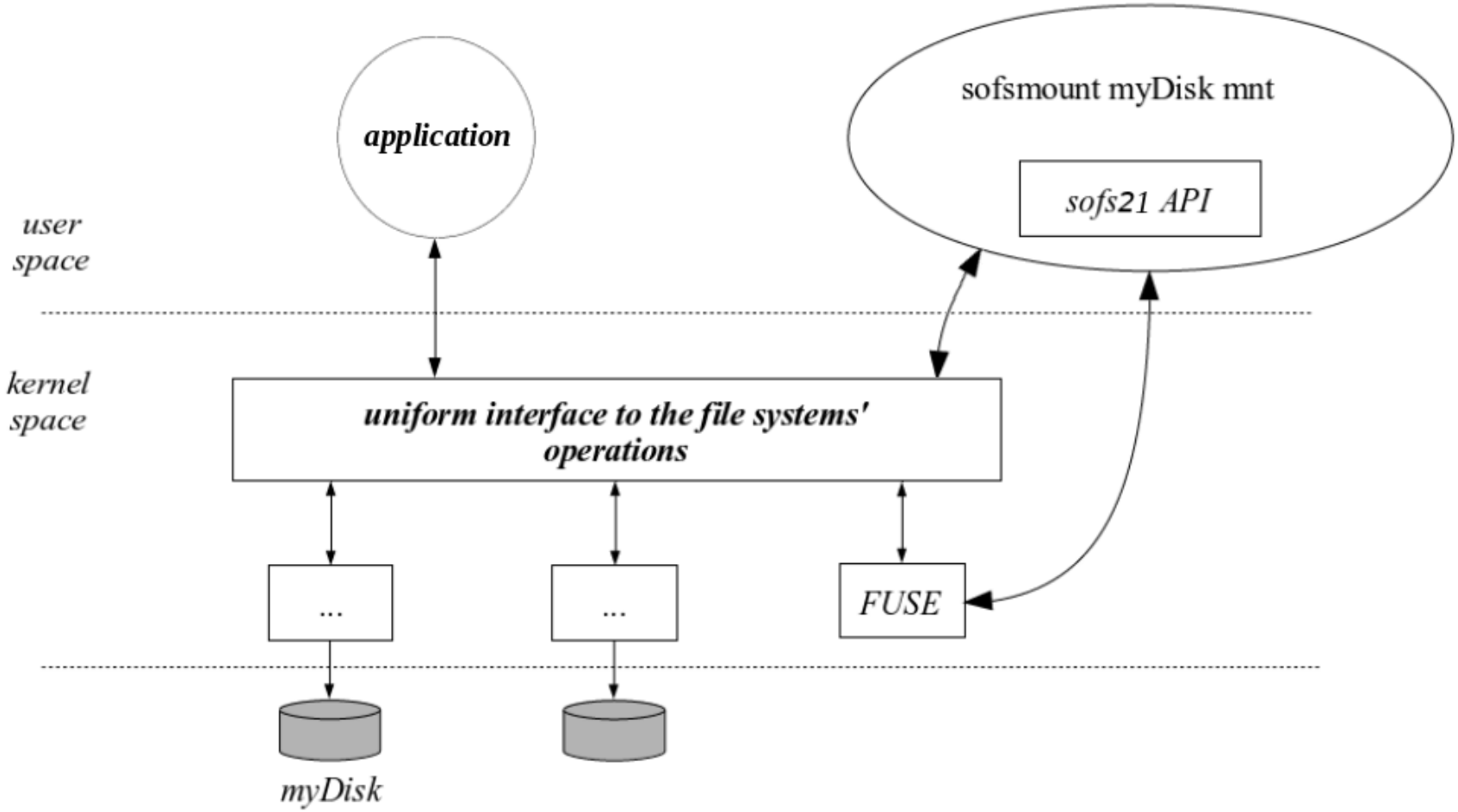
Outline

The FUSE file system
sofs21 as a kernel module



- Safety issue: running in kernel space
  - Malicious or erroneous code can damage the system

The FUSE file system
sofs21 as a FUSE module



- Safe: running in user space
  - Malicious or erroneous code only affects the user

The sofs21 architecture
Block partitioning

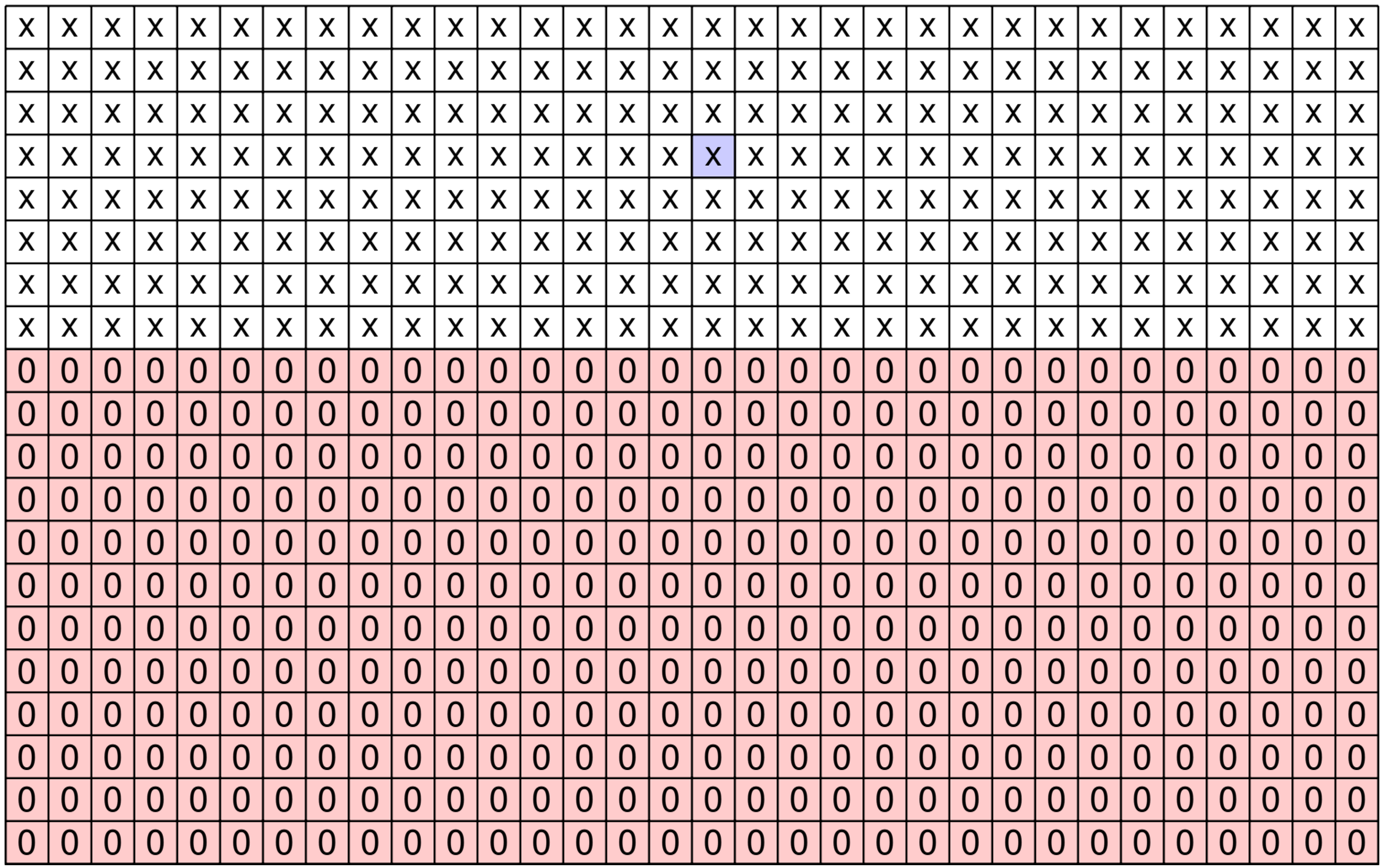
- A sofs21 disk is partitioned/structured as follows:
  - A block, named `superblock`, is used for general metadata
  - Inodes are stored in a fized-size dedicated set of blocks (`inode table`)
  - Data blocks are also stored in a fized-size dedicated set of blocks (`data block pool`)
  - List of free inodes is stored in the superblock
  - List of free data blocks is stored in the superblock and in a set of dedicated blocks (`bitmap table`)
  - References of blocks used by inodes are stored in the inodes themselves and in data blocks allocated for that purpose

The sofs21 architecture
List of free inodes

- Based on a bitmap
  - there is a one-to-one correspondence between bits in the map and inodes in the inode table, including inode 0
  - 1 $\Rightarrow$ inode is free; 0 $\Rightarrow$ inode is in-use
- The bitmap is stored in the superblock (`ibitmap` field)
  - seen as an array of 32-bit words, with fixed size
  - inode 0 is represented by bit 0 of word 0, and so on
  - unused bits are kept at 0
- freeing operation:
  - clean the inode and put the corresponding bit at 1
- allocating operation:
  - search for a bit at 1, put it at 0, and initialize the corresponding inode
  - the search must start in the position circularly next to the last allocated inode (`iidx` field)

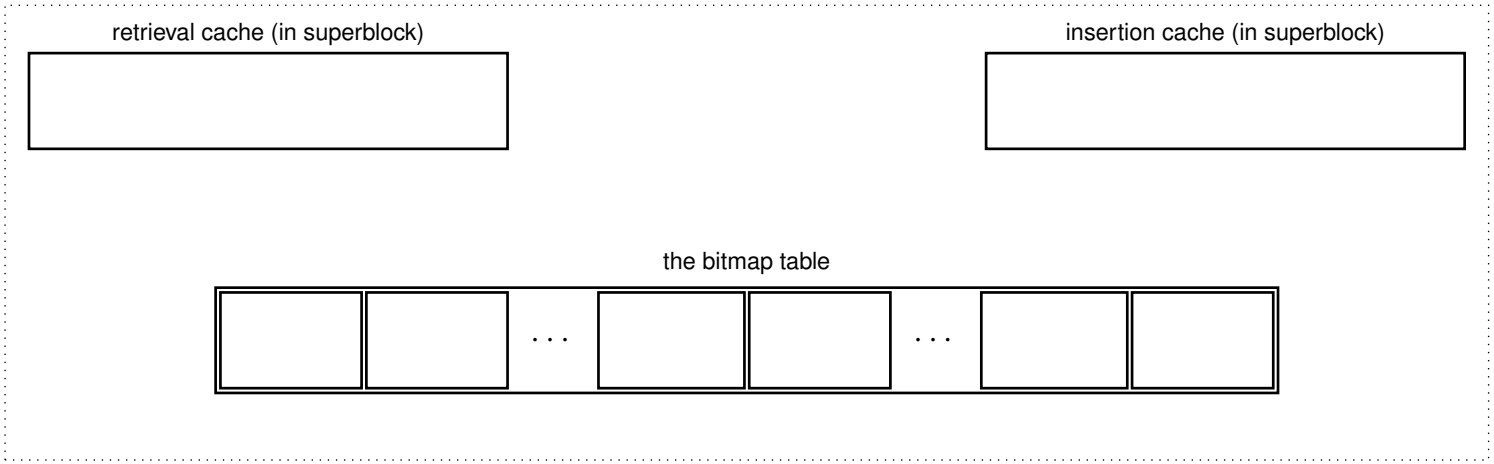
The sofs21 architecture
List of free inodes (2)

- A possible state of the bitmap



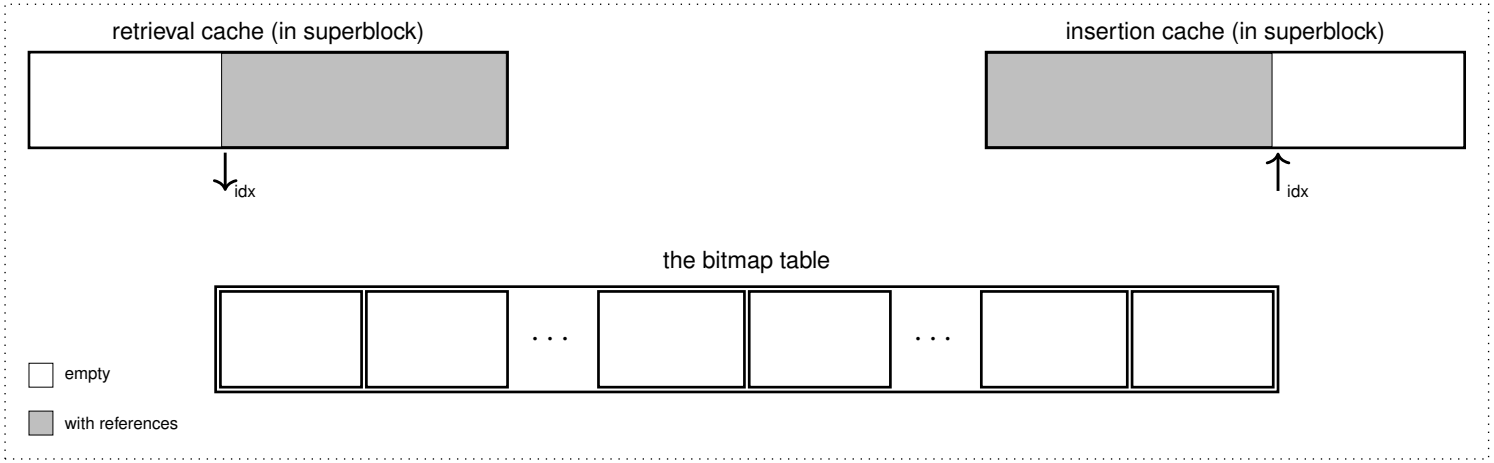
The sofs21 architecture
List of free data blocks

- Based on a bitmap and on two caches
  - the bitmap is stored in dedicated blocks, each seen as an array of 32-bit words
  - a first (ordered) sub-sequence of references is stored in the retrieval cache, representing the next data blocks to be allocated
  - a last (ordered) sub-sequence is stored in the insertion cache, representing the most recently freed data blocks
  - the remaining free data blocks are stated in the bitmap table



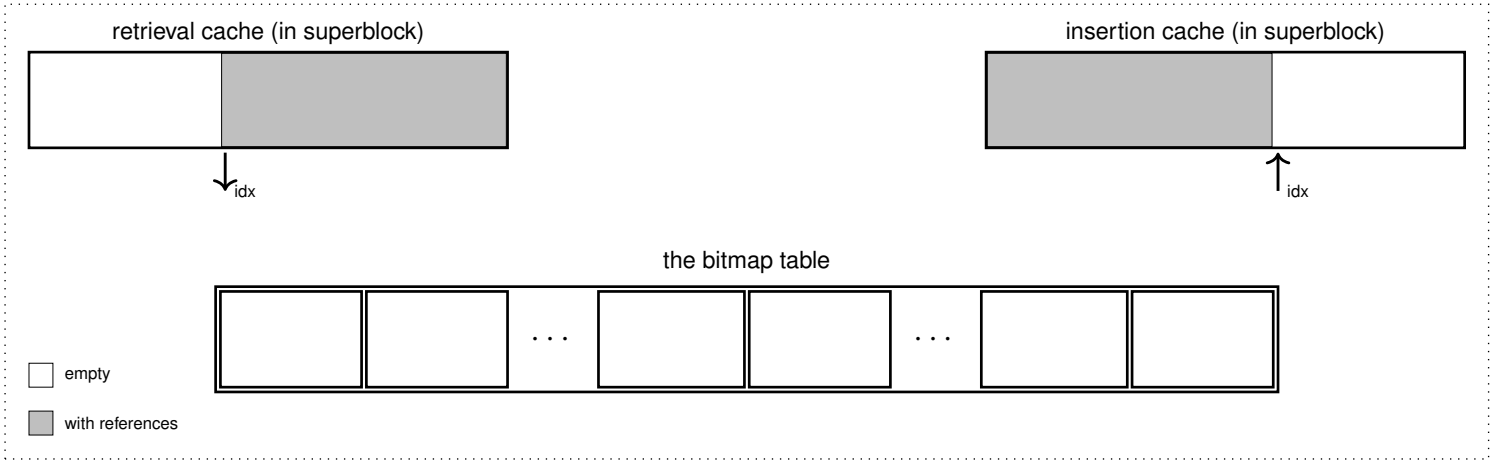
The sofs21 architecture
List of free data blocks

- retrieval cache
  - this cache may be partially empty, meaning that some references (necessarily at the begining) were already retrieved
  - an index (idx in the figure) points to the first cell with a reference
- insertion cache
  - this cache may be partially filled, meaning that some references (at the begining) were already inserted
  - an index (idx in the figure) points to the first empty cell



The sofs21 architecture
List of free data blocks

- bitmap table
  - Two fields in the superblock (`rbm_start` and `rbm_size`) delimit the region of the disk with the bitmap table
  - Another field (`rbm_idx`) states where a new transference should start from
    - thus creating a kind of circularity in the use of blocks



The sofs21 architecture
Sequence of blocks of a file (1)

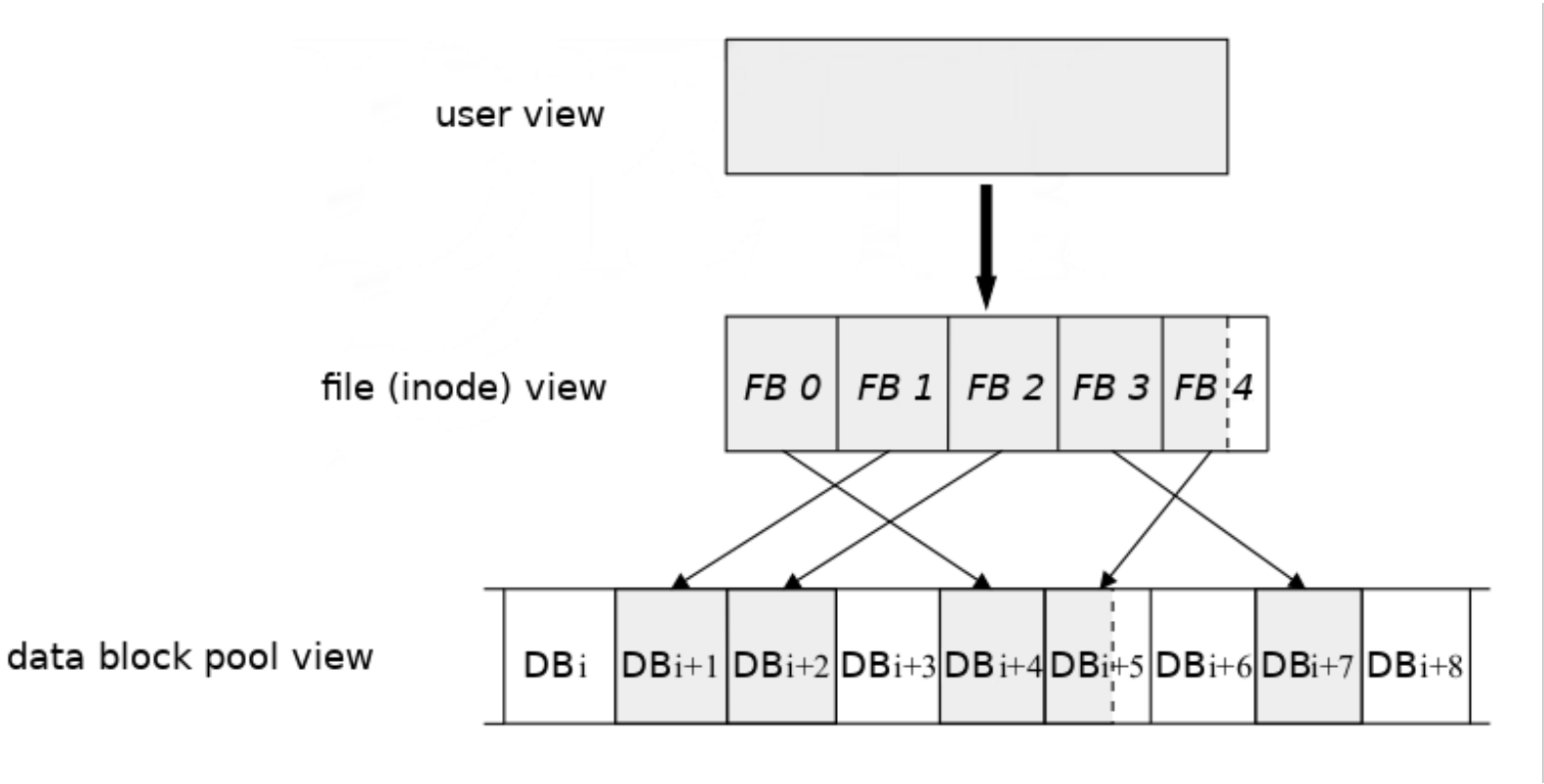
- Blocks are not shareable among files
  - an in-use block belongs to a single file
- The number of blocks required by a file to store its information is given by

$$N_b = \text{roundup}\left(\frac{\text{size}}{\texttt{BlockSize}}\right)$$

- N_b can be very big
  - if block size is 1024 bytes, a 2 GByte file needs 2 MBlocks
- N_b can be very small
  - a 0 bytes file needs no blocks for data
- It is impractical that all the blocks used by a file are contiguous in disk
- The access to the file data is in general not sequencial, but instead random
- Thus a flexible data structure, both in size and location, is required

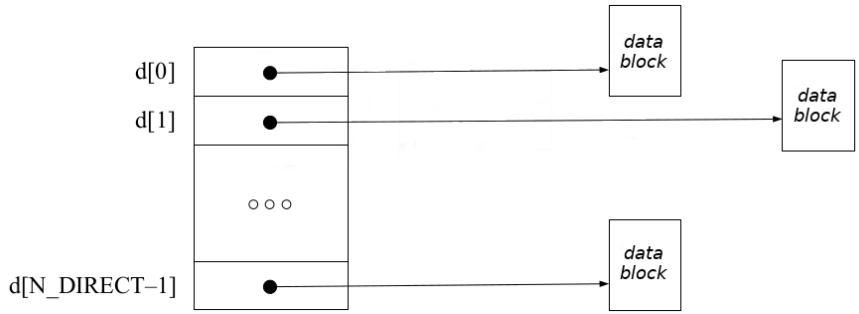
The sofs21 architecture
Sequence of blocks of a file (2)

- The programmer views a file as a continuum of bytes
- The inode views a sequence of blocks (`file block`)
- The data blocks are, in general, scattered along the data block pool

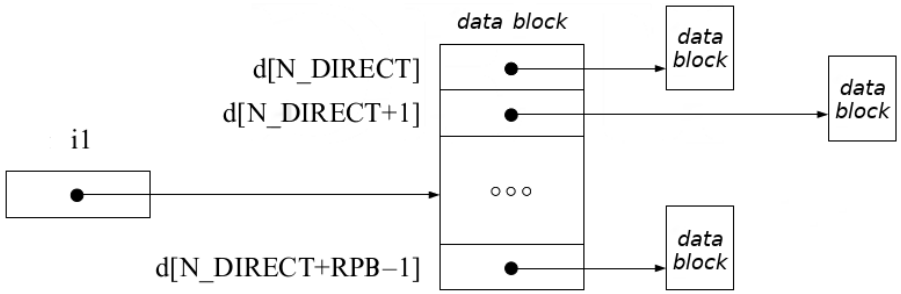


The sofs21 architecture
Sequence of blocks of a file (3)

- How is the sequence of (references to) data blocks stored?
- The first references are directly stored in the inode

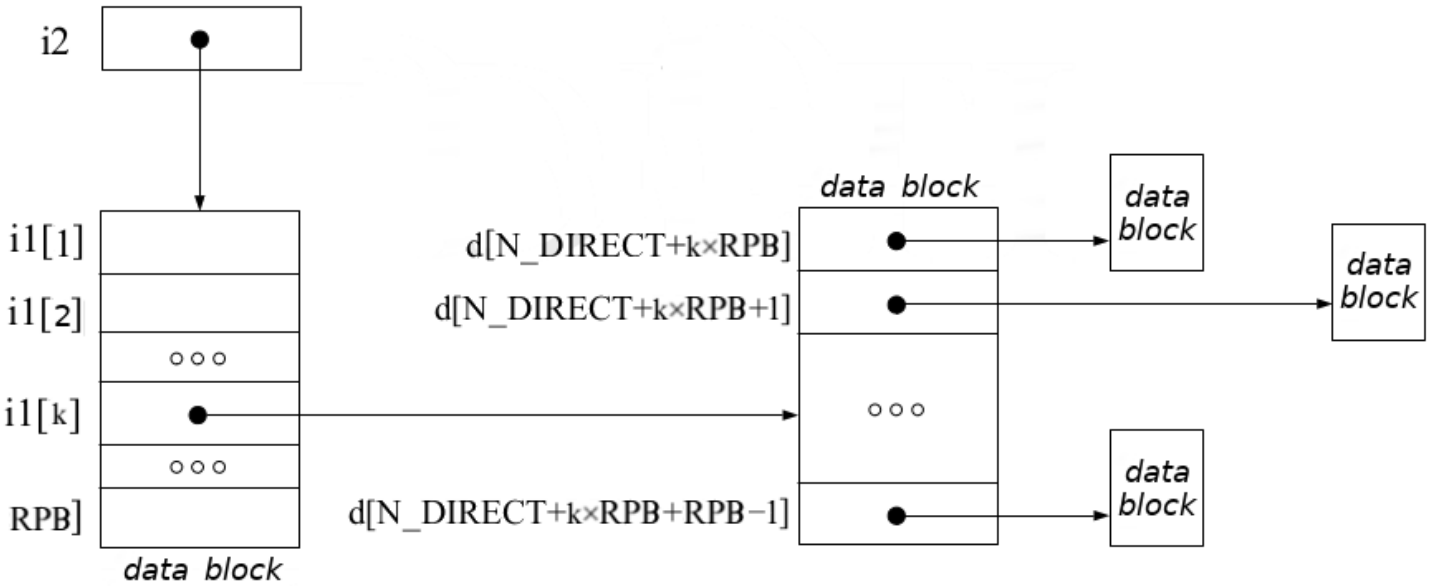


- Then, inode field `i1` points to a data block with references



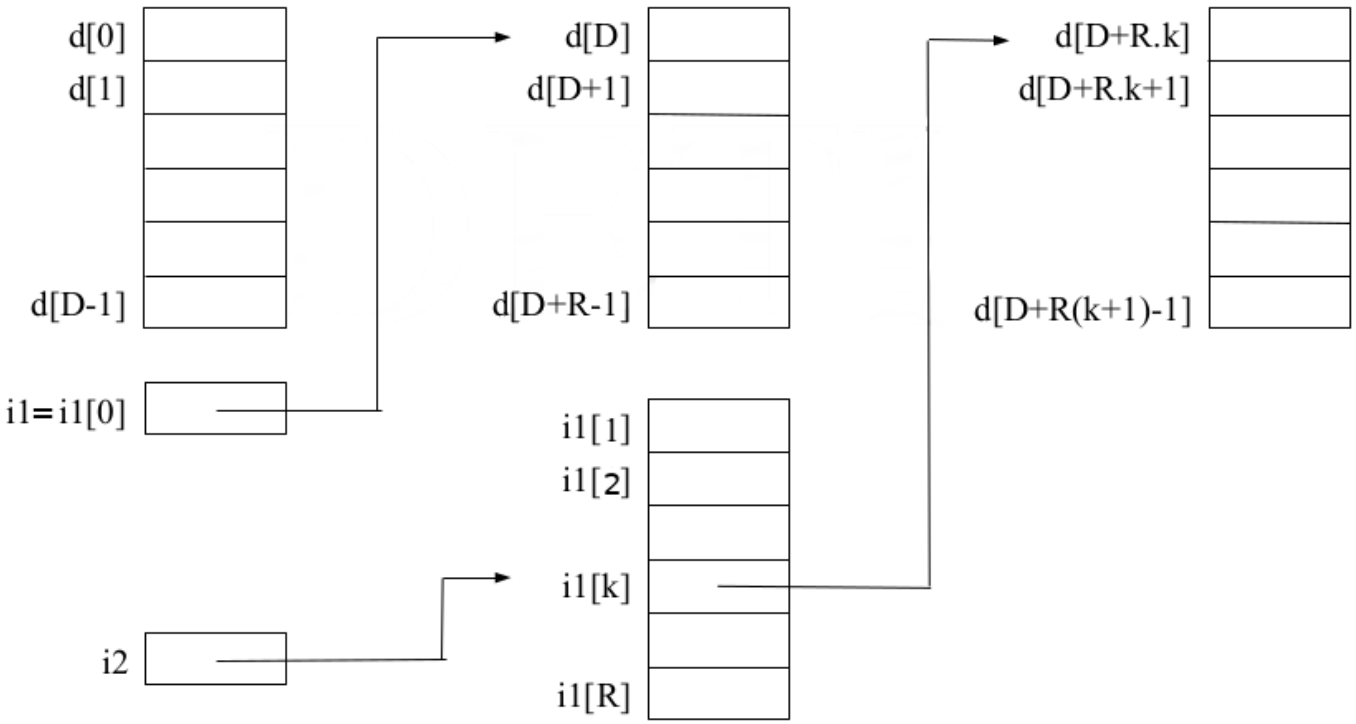
The sofs21 architecture
Sequence of blocks of a file (4)

- Finally, inode field `i2` point to a data block that extends `i1`



The sofs21 architecture
Sequence of blocks of a file (5)

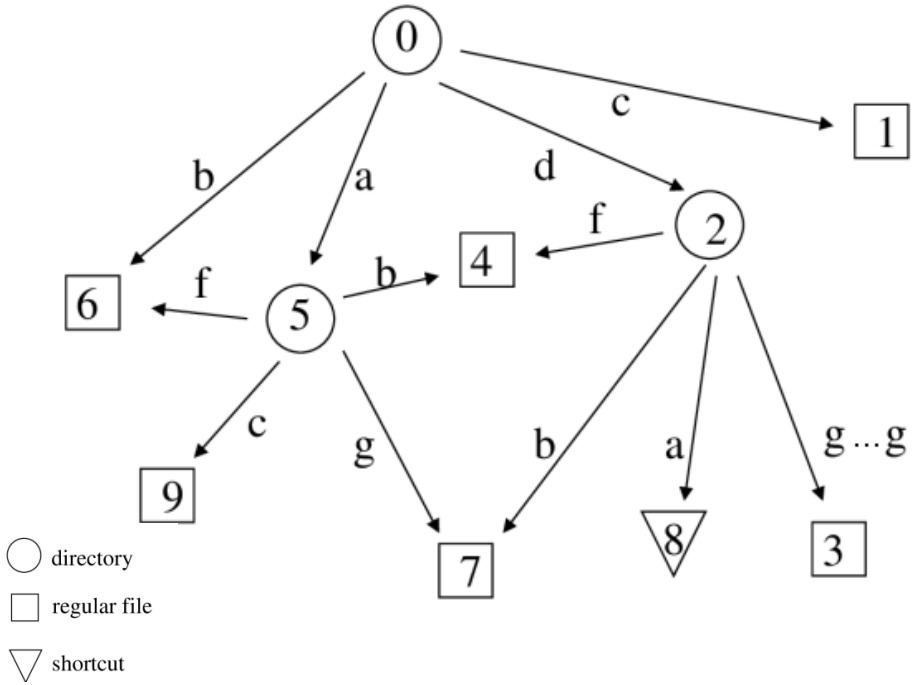
- Puting all together



- A file can contain “holes”
  - corresponding to null references covered by the size
  - and representing streams of zeros

The sofs21 architecture
Directories and directory entries

- A directory is:
  - (functionally) a list of directory entries
  - (structurally) a list of directory slots
- A directory entry is a pair that associates a name to an inode
- A directory slot is the fixed-size partition of a directory
- A directory entry can occupy 1 or more directory slots

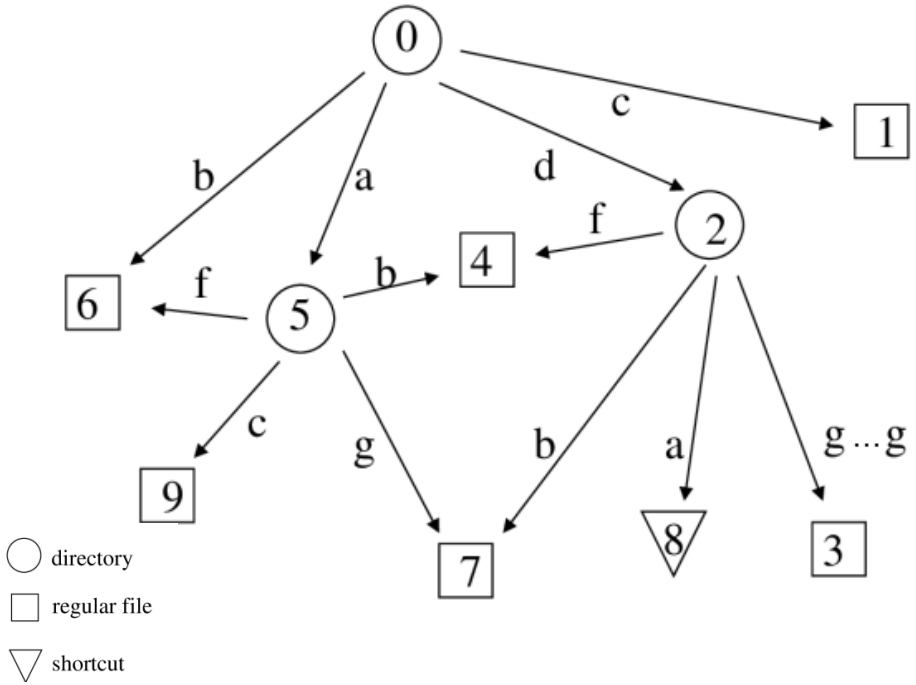


- The contents of directory "/" (inode 0) could be:

name	inode
.	0
..	0
c	1
d	2
a	5
b	6

The sofs21 architecture
Directories and directory entries

- A directory is:
  - (functionally) a list of directory entries
  - (structurally) a list of directory slots
- A directory entry is a pair that associates a name to an inode
- A directory slot is the fixed-size partition of a directory
- A directory entry can occupy 1 or more directory slots

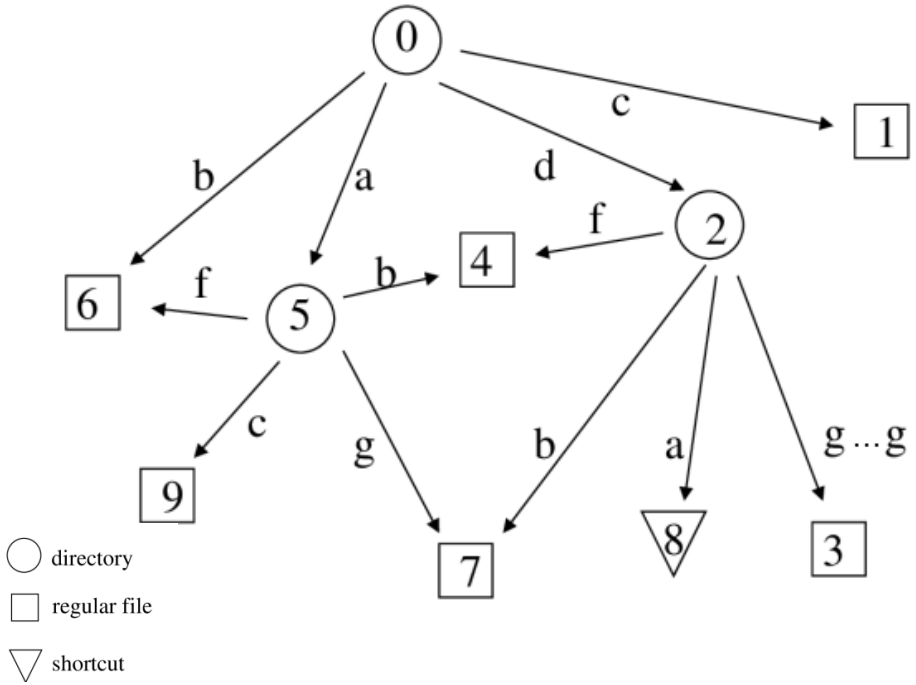


- The contents of directory "/a/" (inode 5) could be:

name	inode
.	5
..	0
c	9
b	4
	(nil)
f	6
g	7

The sofs21 architecture
Directories and directory entries

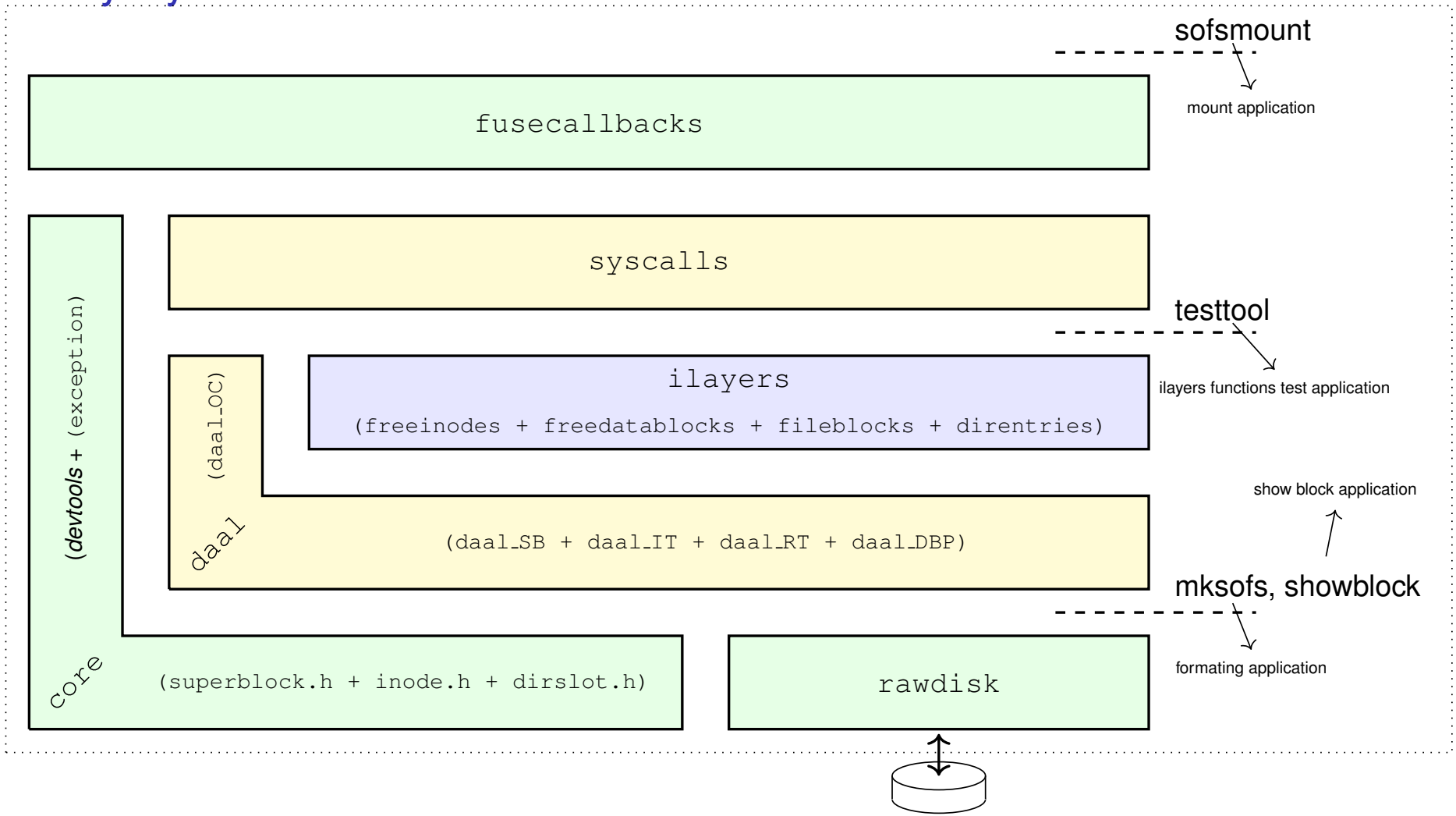
- A directory is:
  - (functionally) a list of directory entries
  - (structurally) a list of directory slots
- A directory entry is a pair that associates a name to an inode
- A directory slot is the fixed-size partition of a directory
- A directory entry can occupy 1 or more directory slots



- The contents of directory "/d/" (inode 2) could be:

name	inode
.	2
..	0
a	8
g...	3
...g	3
f	4
b	7

The sofs21 code structure
Library layers and tools



The sofs21 code structure
Development tools

- Code prepared to use the building tool cmake
  - Need to prepare cmake
  - Can choose between make and ninja

- Code prepared to use the documentation tool doxygen
  - Configured to use only `.h` files
  - Configured to generate only html pages

- sofs21 specific available tools:
  - showblock – show one or more blocks of a sofs21 disk
  - testtool – call functions of the intermediate layers

The formatting tool

`mksofs`

- Purpose:
  - Fill in the blocks of a raw disk to make it be a sofs21 file system

- State of a newly formatted disk:
  - Inode 0 is used by the root directory
  - Data of the root directory is stored in data block number 0
  - A set of other rules have also to be observed
    - they are stated in the documentation

- Approach:
  - Code was decomposed in 6 auxiliary functions
  - Source of the main code is given

The formating tool
Testing

- Activating and using the bash basic functions can help in the test

- In a newly formatted disk, what is the state of the:
  - list of free inodes
  - list of free data blocks
  - inode table
  - root directory
  - free data blocks

- State of inodes after formatting:
  - inode 0 is in use, while all other inodes are free
  - not used bits of the inode bitmap are stated as not free
  - thus, in the bitmap, the LSB of word 0 and not used bits must be put at 0

The formating tool
Testing (2)

- Free data blocks after formatting:
  - insertion cache is empty
  - retrieval cache is empty
  - data block 0 is in use, while all other inodes are free
  - not used bits of the data block bitmap are stated as not free
  - thus, in the bitmap, the LSB of word 0 and not used bits must be put at 0

- State of the inode table:
  - inode 0 is in use as a directory
    - it uses data block 0 at position 0 (d[0]=0)
    - size = `BlockSize` = 1024
  - all other inodes are free

The formating tool
Testing (3)

- State of the root directory:
  - 2 entries used
    - first: name = "." inode = 0
    - second: name = ".." inode = 0
  - remainder of the block must be clean
    - `nameBuffer` filled with the null character
    - inode reference filled with `NullInodeReference`

- State of the free data blocks:
  - If option -z is used, they must contain zeros
  - Otherwise, they are untouched