Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 3: File systems and I/O

Seminar 8

Minix file system

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

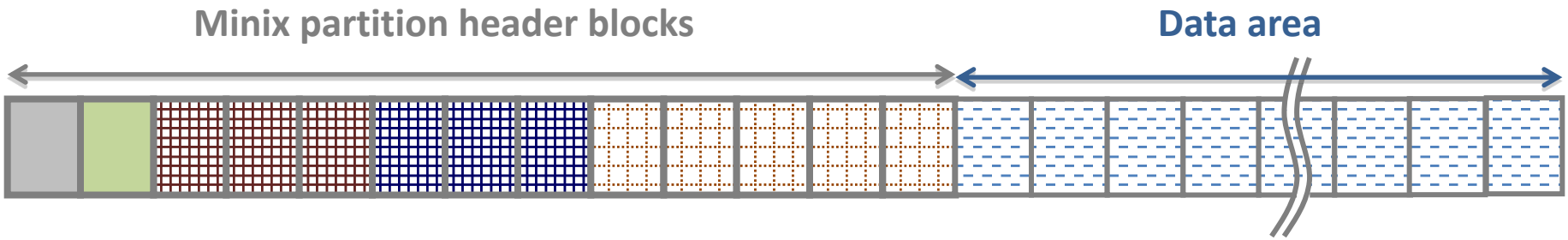
Minix file system

- **Goals**
  - To know the structure of a Minix partition
  - To know how the OS manage i-nodes to keep file information
  - To understand the bit map concept to manage free and busy space
  - To be able to locate a particular file within a directory structure from its absolute path
- **Bibliography**
  - "Operating Systems Design and Implementation" (3rd Edition), Andrew S. Tanenbaum, Prentice Hall 2006 Section 5.6

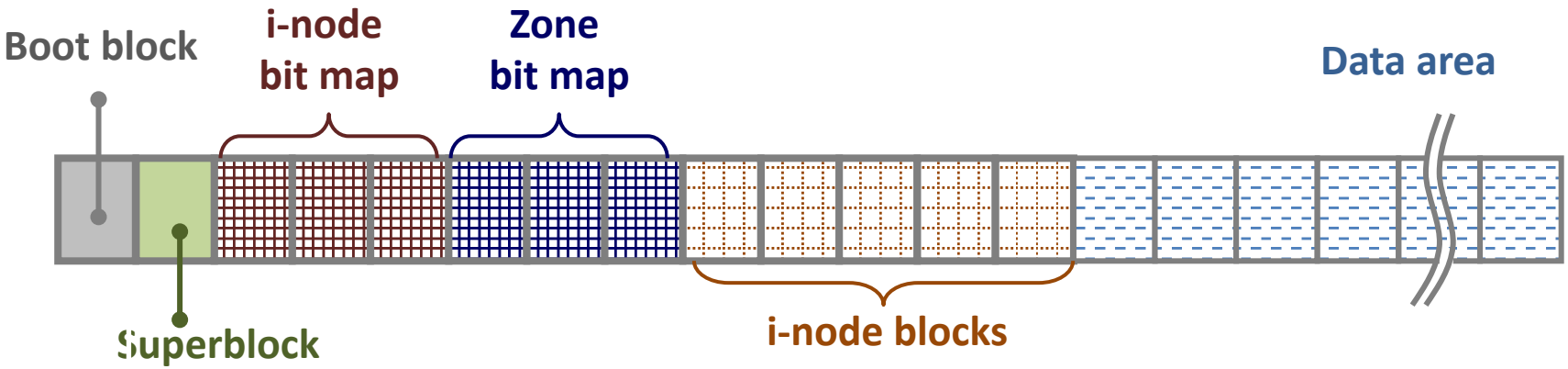
- **Partition structure**
- i-node structure
- Directory entry
- Standard sizes
- Exercises

Partition structure

- **A Minix partition** is built upon a set of fixed size blocks (i.e. 1KByte)
  - A partition structure contains:
    - A **header** made up of block groups intended to store the data structures that sustain the file system
    - **Data area** made up of blocks intended to store file data



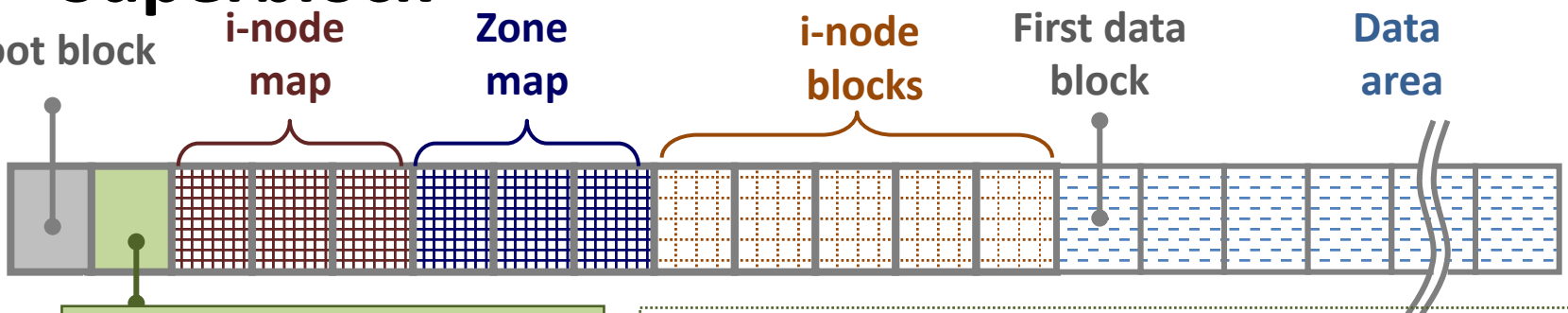
- **Header blocks**



  - **Boot block:** contains the boot program that loads the operating system and transfers the control to it
  - **Superblock:** is a data structure with the file system description that indicates the size and location of every element
  - **i-node bit map**: bit vector to manage free and allocated i-nodes. It contains one bit per i-node
  - **Zone bit map**: bit vector to manage free and allocated zones. It contains one bit per zone
  - **i-node blocks:** contains the i-node data structures. The i-node number depends on the partition size. i-node 0 is not used

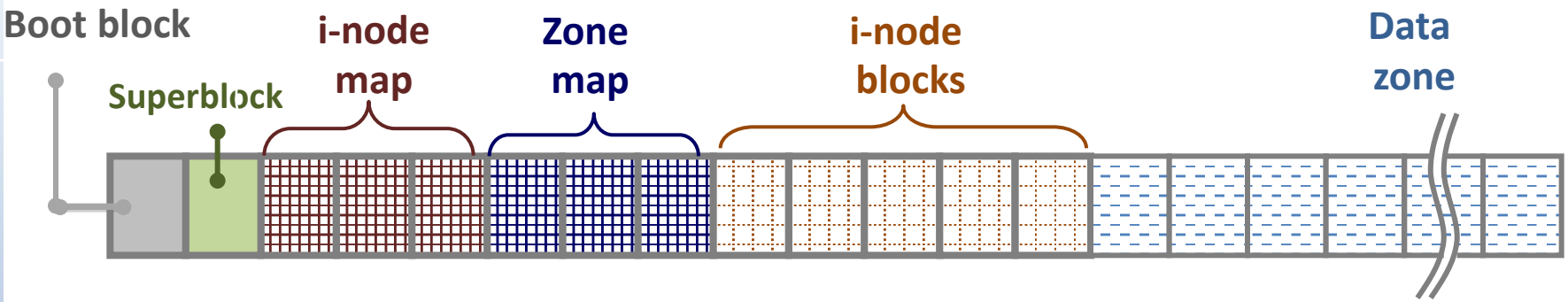
Partition structure

- **Superblock**



	Superblock	
00	Number of i-nodes	It is fixed by the user at partition creation or it takes default values
02	Number of data zones	It is fixed at partition creation, it can be less than partition size
04	Number of i-node map blocks	[i-nodes number / Block bits number]
06	Number of zone map blocks	[Data zones number / Block bits number]
08	Block number of the first data zone	2 (boot and superblock) + number of i-node map blocks + number of zone map blocks + number of i-node blocks
0A	N (1 zone= 2^N * 1024)	1 zone = 2^N blocks N value is stored into the superblock
0C	Maximum file size	Maximum file size in bytes
10	Magic number	Numerical value that guaranties that the partition contains a MINIX file system

- **i-node/zone bit maps**



i-node map

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Zone map

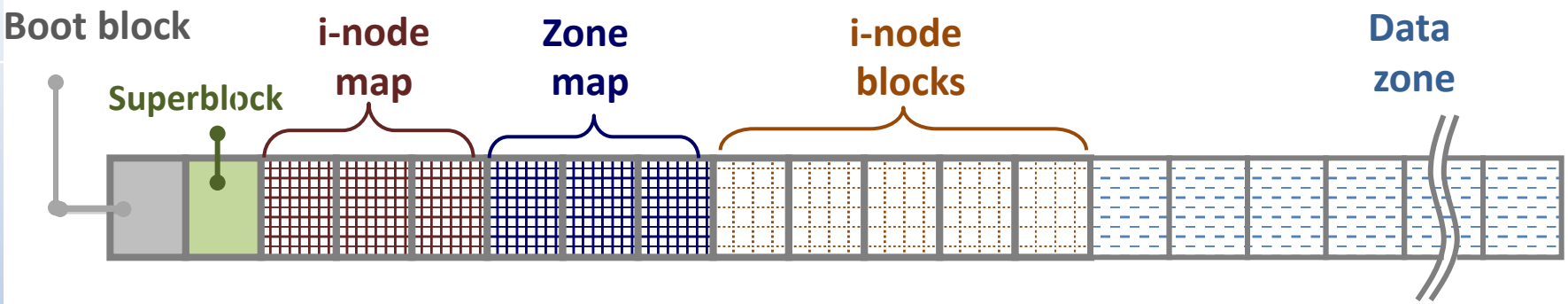
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0
1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

0 bit -> allocated
1 bit -> free

Every **i-node** is **represented** by **one bit** equal to 0 or 1 if the i-node is allocated or free, respectively

Every **data zone** is **represented** by **one bit** equal to 0 or 1 it the zone is allocated or free, respectively

- **Data zone**



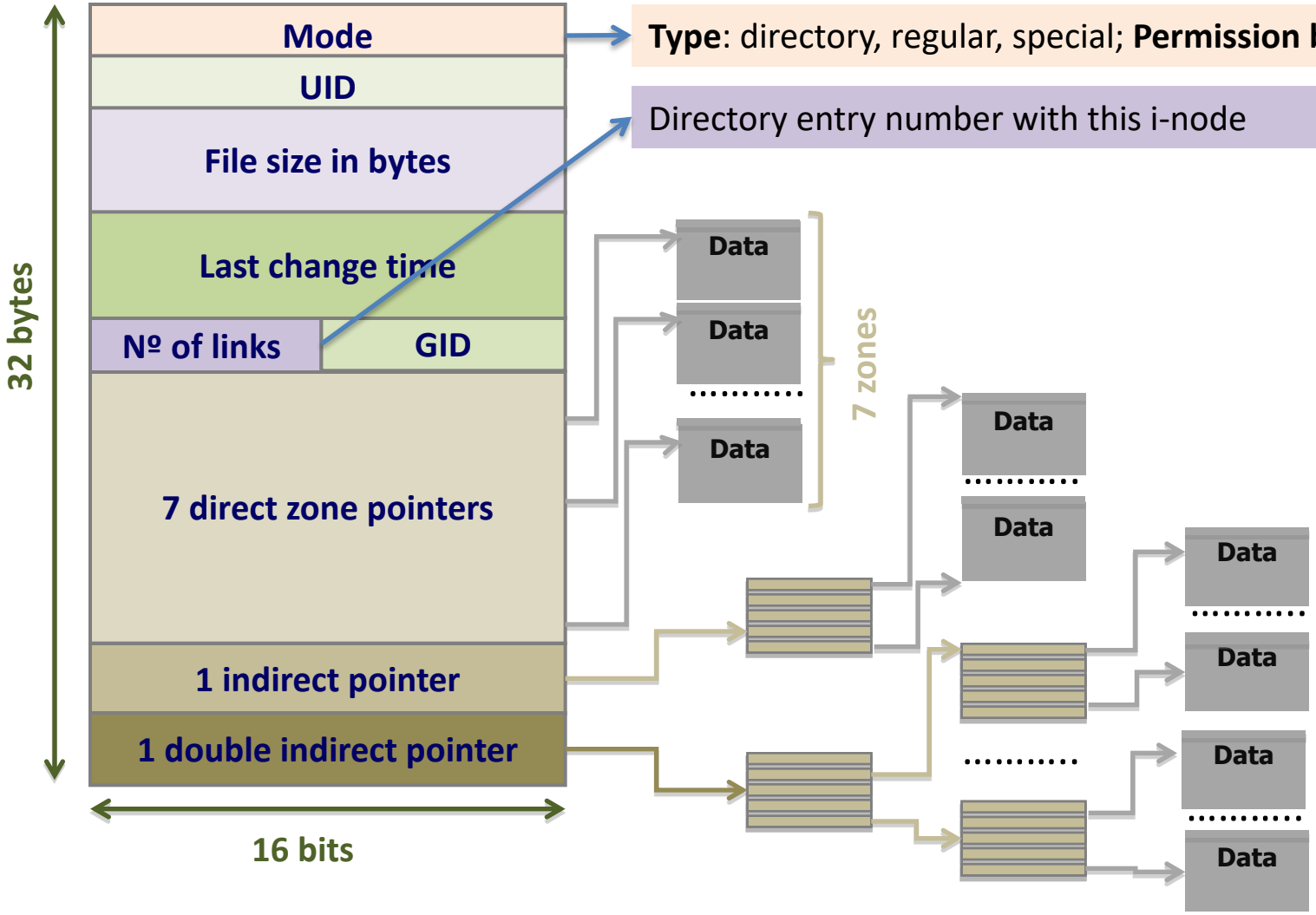
- Block set used to store regular file content, directory entries and block references
  - To be able to address big partitions, MINIX file system allows grouping blocks into zones
  - Zone is the file allocation unit
    - **1 zone = 2^N blocks** → by default 1 zone = 2^0 blocks = 1block
    - The first data block (referenced inside the superblock) is adjusted to a zone starting block

- Partition structure
- **i-node structure**
- Directory entry
- Standard sizes
- Exercises

i-node structure

– Data structure that contains all file attributes except its name

- Every file has an associated i-node
- It controls indexed allocation by means of direct, indirect and double indirect pointers



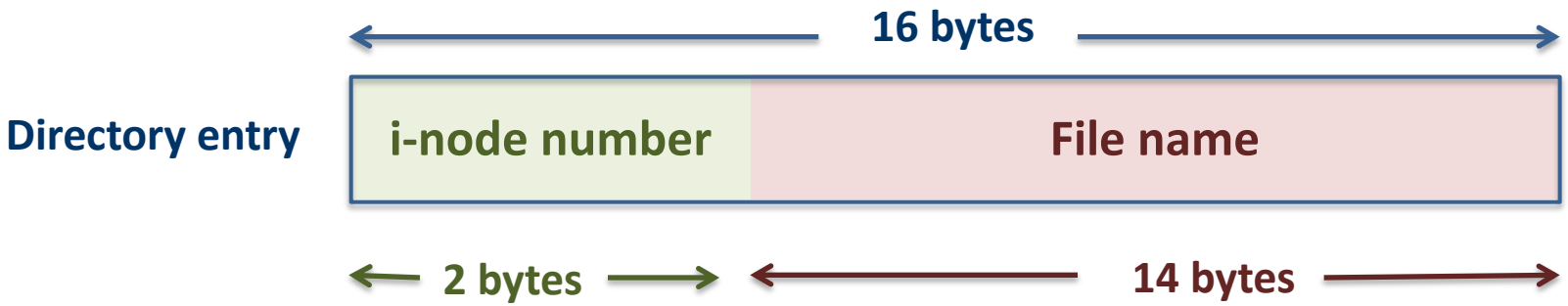
i-node structure

- Performance analysis
  - **Efficient random access:** The maximum number of disk accesses is limited to 4
    - The indirect pointers are only used with big or very big files (commonly few)
    - Small file (common case) access is very efficient
  - **Reliable and elegant design**: every file has its own separated data structure

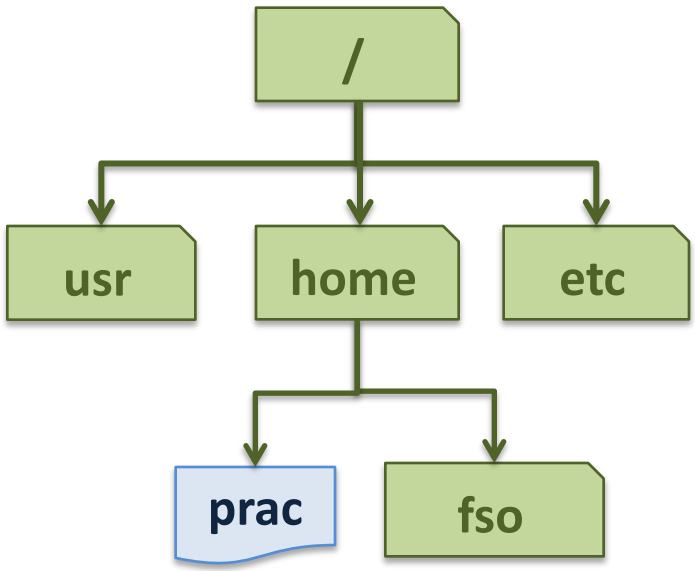
- Partition structure
- i-node structure
- **Directory entry**
- Standard sizes
- Exercises

Directory entry

- **Minix directories**
  - Directory structure as a directed acyclic graph (DAG)
  - Directories are files which content is interpreted as registers → directory entries (also named links)
- **Minix directory entry or link**
  - It has a 16 byte size
    - 2 bytes for the i-node
    - 14 bytes for the file name



- **Directory entry**
  - When a directory is created, the **entries '.' and '..'** are automatically created
  - **i-node 1 describes the root directory**
  - When a directory entry is removed it is marked with i-node 0



1	.
1	..
3	usr
4	home
5	etc

4	.
1	..
35	prac
84	fso

84	.
4	..

3	.
1	..

5	.
1	..

Content

- Partition structure
- i-node structure
- Directory entry
- **Standard sizes**
- Exercises

Standard sizes

- Default sizes for Minix elements
  - 1 Zone =2^0 blocks = 1024 bytes
  - 1 pointer to zone or block = 2 bytes = 16 bits
  - 1 directory entry = 16 bytes
  - 1 i-node = 32 bytes

Content

- Partition structure
- i-node structure
- Directory entry
- Standard sizes
- **Exercises**

- **Exercise 1: Maximum size of a Minix file**

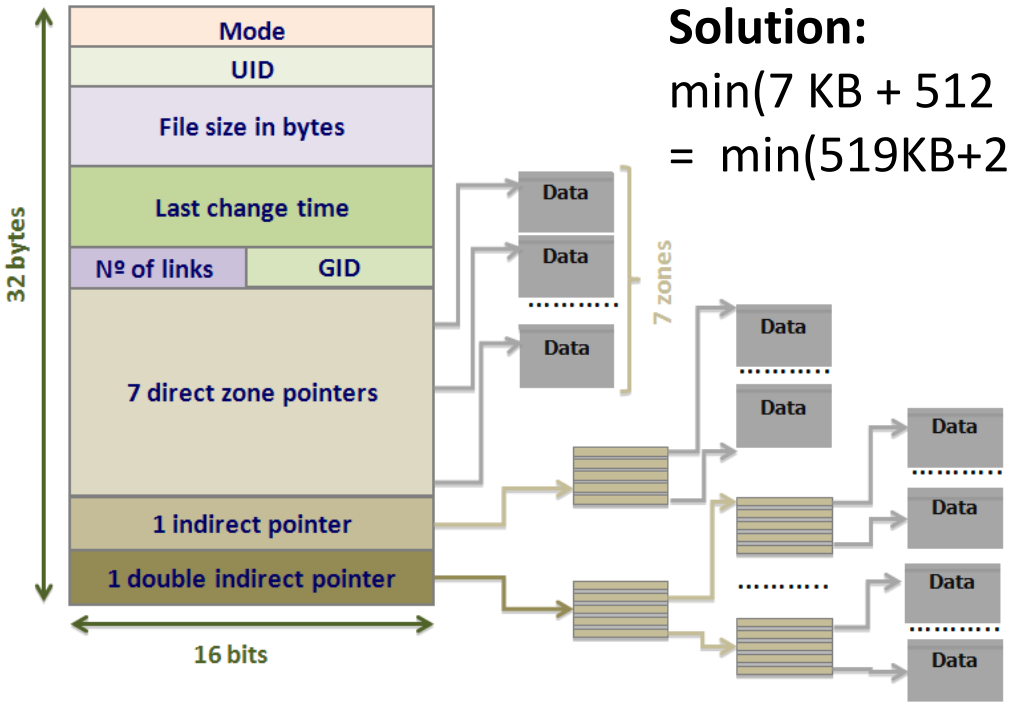
  Obtain the maximum theoretical size of a Minix file, ignoring the device size where the file is stored. Specify the addressed blocks by every pointer type. The Minix parameters considered are:

  16 bit data zone pointers

  1Kbyte block size

  1 zone = 1 block

  i-node structure: 7 direct, 1 indirect and 1 double indirect pointers



Solution:

min(7 KB + 512 KB + 512*512*1KB, 64K*1KB)

= min(519KB+256K*1KB, 64K*1KB) = 64MB

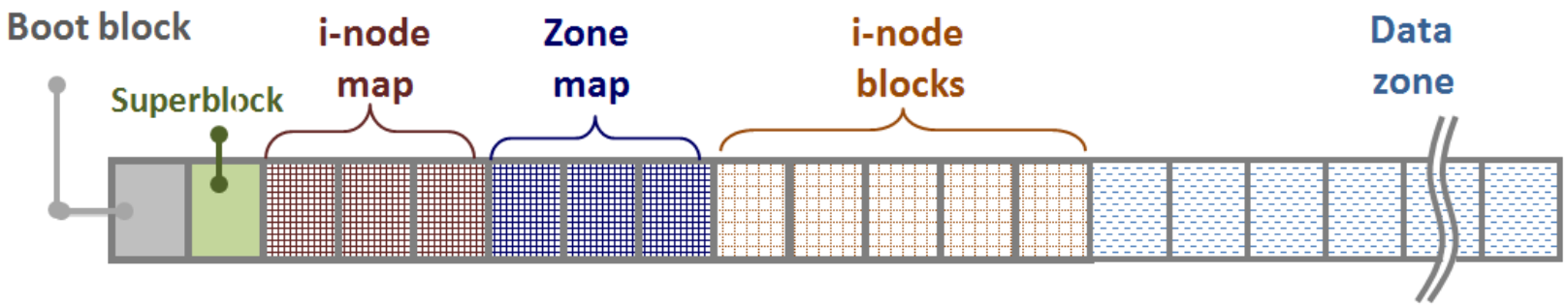
- **Exercise 2: Minix partition structure**

  Given a Minix partition in a 20 Mbyte Disk with the following parameters:

  - 16 bit data zone pointers
  - 1 Kbyte blocks and 1 zone = 1 block
  - 32 byte i-node size
  - Maximum number of i-nodes: 512

  a) Specify all the file system data structures and the blocks number that every one requires

  b) In case of zone bit map fault think about how to reconstruct it from the information available in the other (error free) file system structures



- **Exercise 3: Looking for a file inside a directory**

  Consider a Minix file system created with standard sizes which actual directory entries are the following:

1	.
1	..
3	usr
4	home
5	etc

4	.
1	..
35	prac
84	fso

3	.
1	..
60	ut12
61	sut12

84	.
4	..
90	map
91	listado

5	.
1	..

90	.
84	..

a) Draw the directory and file tree that corresponds to this system

b) Describe **what i-node numbers and how many blocks** are accessed to read the first 128 bytes in file **/home/fso/listado**

c) What information should we get when reading the first 32 bytes in file **/home/fso/map**