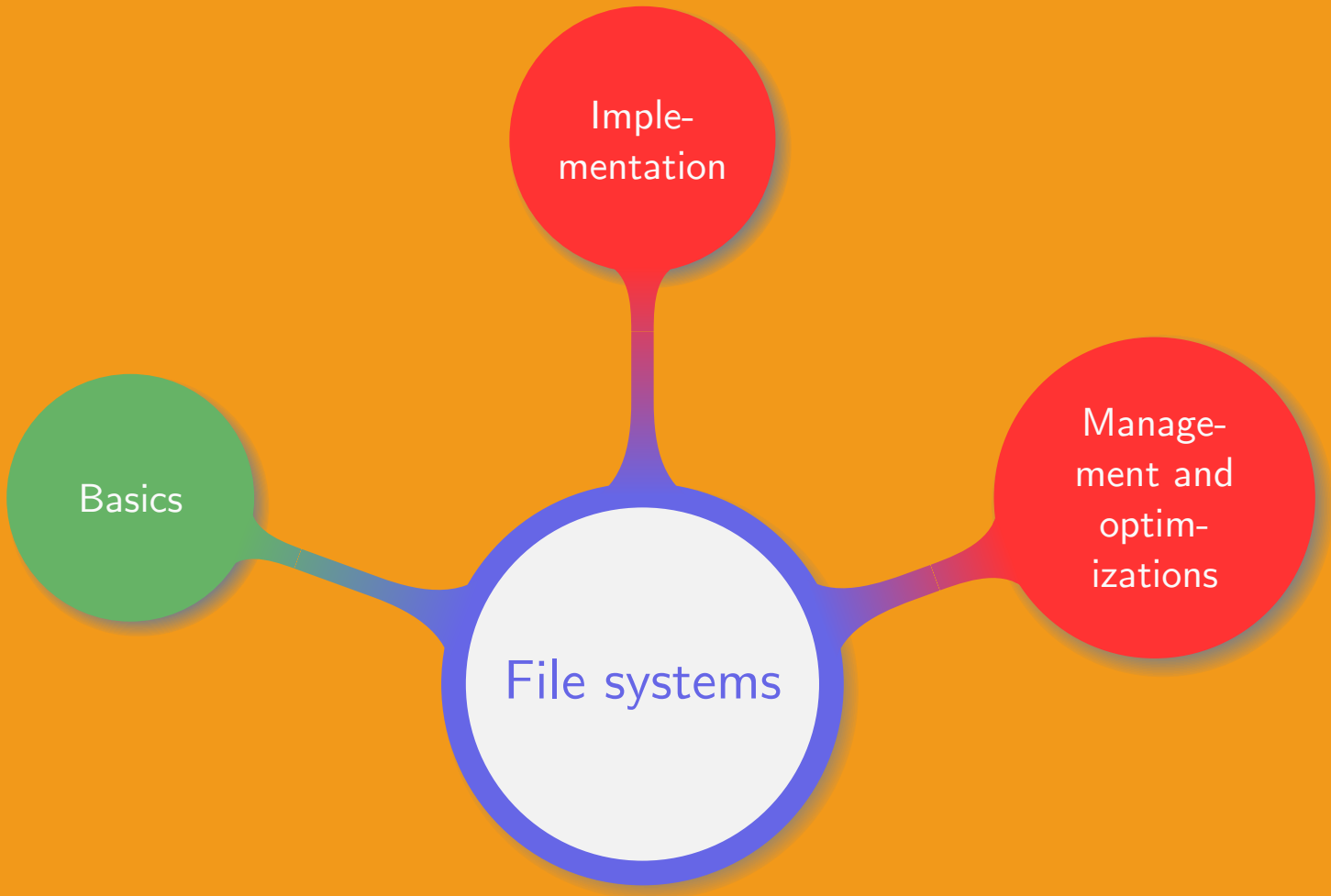
Introduction to Operating Systems

8. File systems

Manuel – Fall 2020

Chapter organisation



Goals

Limitations of virtual memory:

- Small
- Volatile
- Process dependent

Goals

Limitations of virtual memory:

- Small
- Volatile
- Process dependent

Goals that need to be achieved:

- Store large amount of data
- Long term storage
- Information shared among multiple processes

User's view

High level view of a file-system:

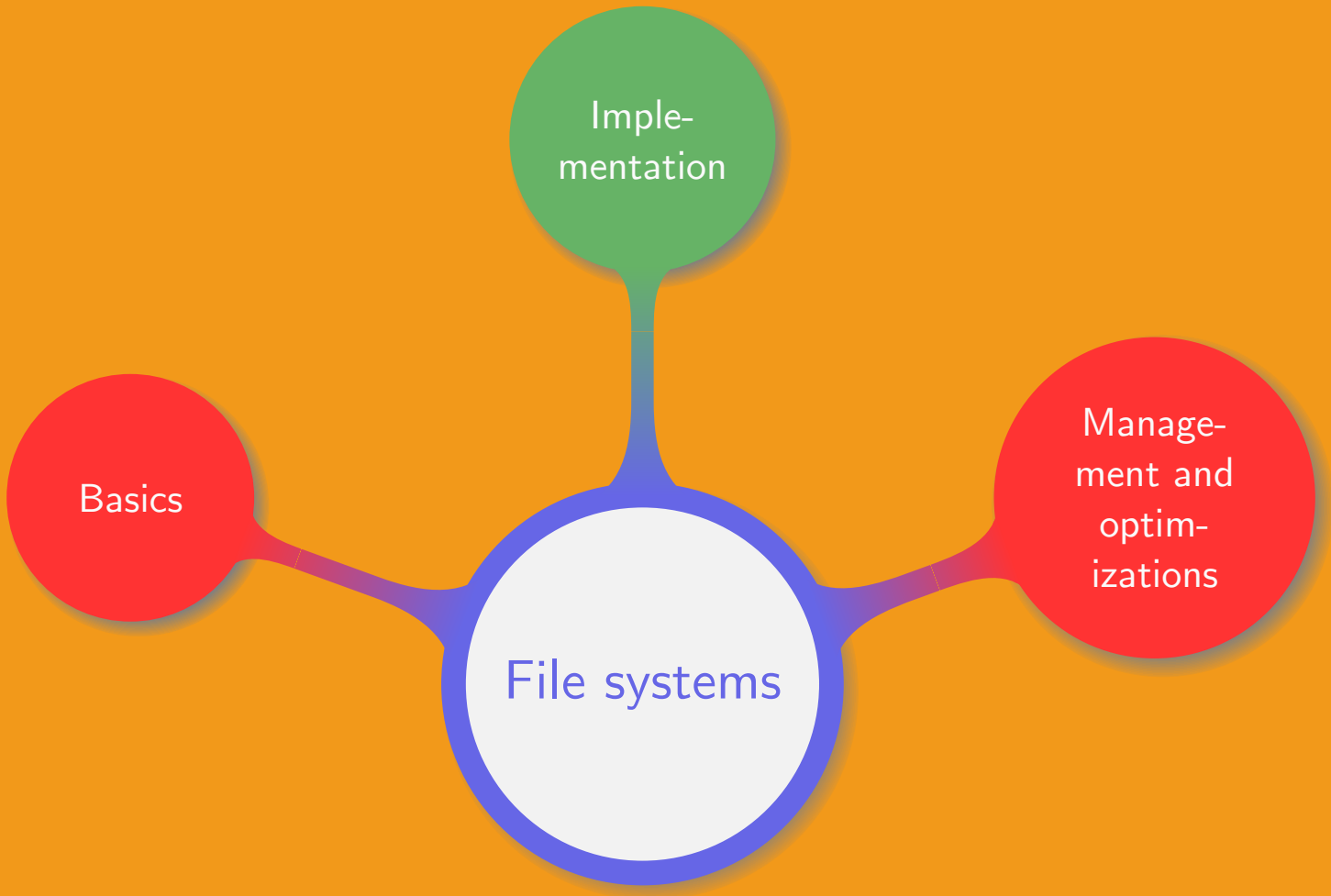
- Small part of the disk memory can be directly accessed using high level abstraction called a *file*
- File name can be case sensitive or insensitive
- File name is a string with (an optional) suffix
- Each file has some attributes containing special information

File organisation

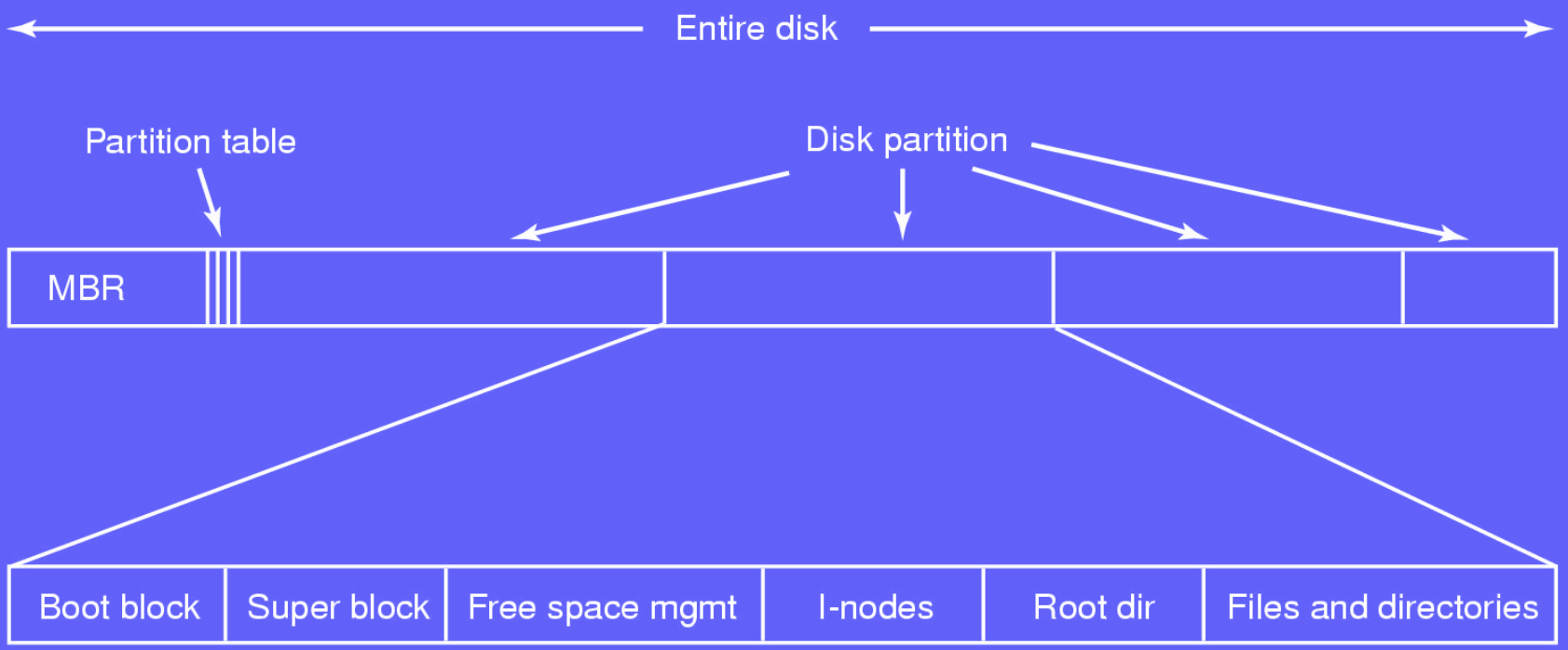
Structure content:

- Files are grouped inside a *directory*
- Directories are organised in a *tree*
- Each file has an *absolute path* from the root of the tree
- Each file has an *relative path* from the current location in the tree

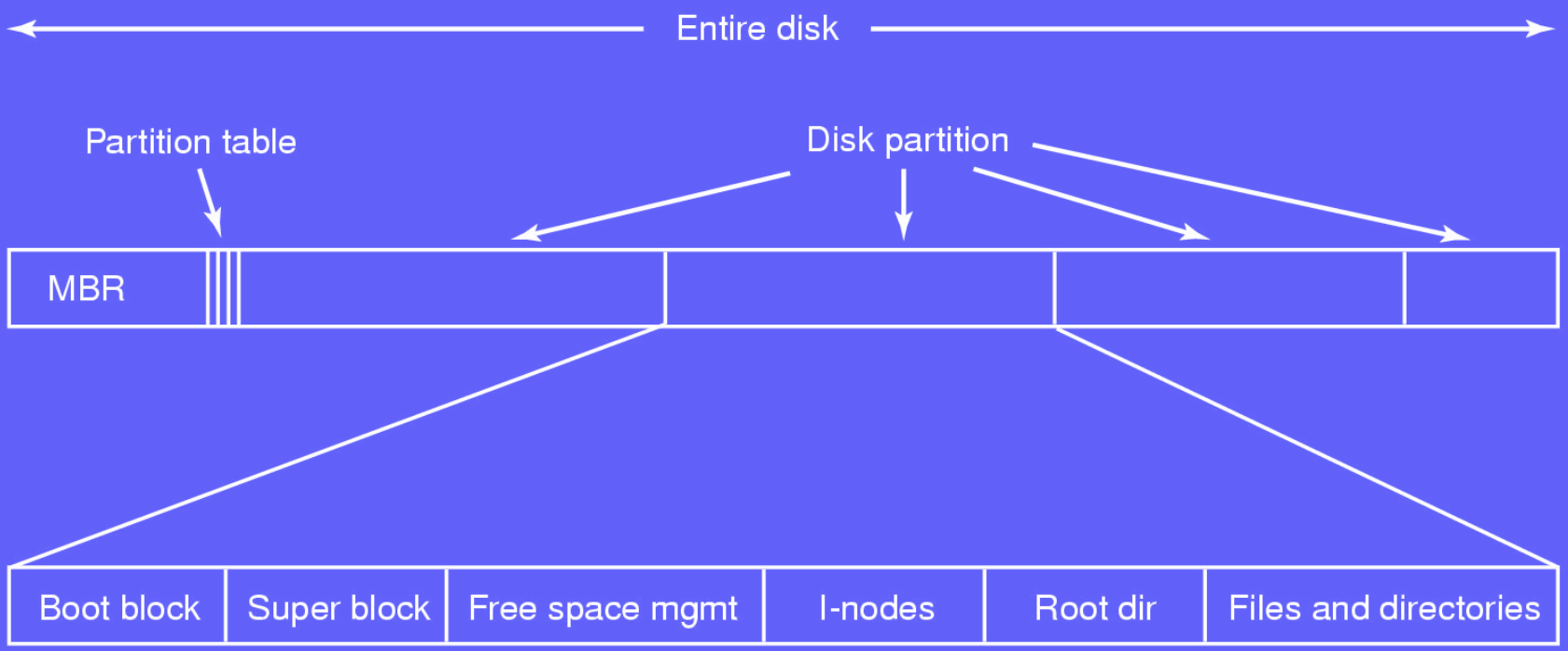
Chapter organisation



Basic disk layout

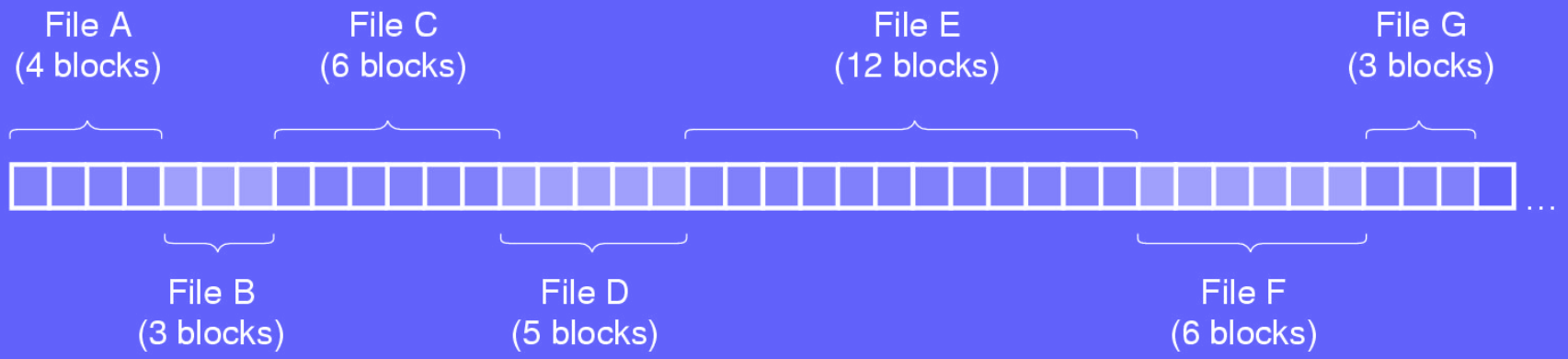


Basic disk layout



How to efficiently match disk blocks and files?

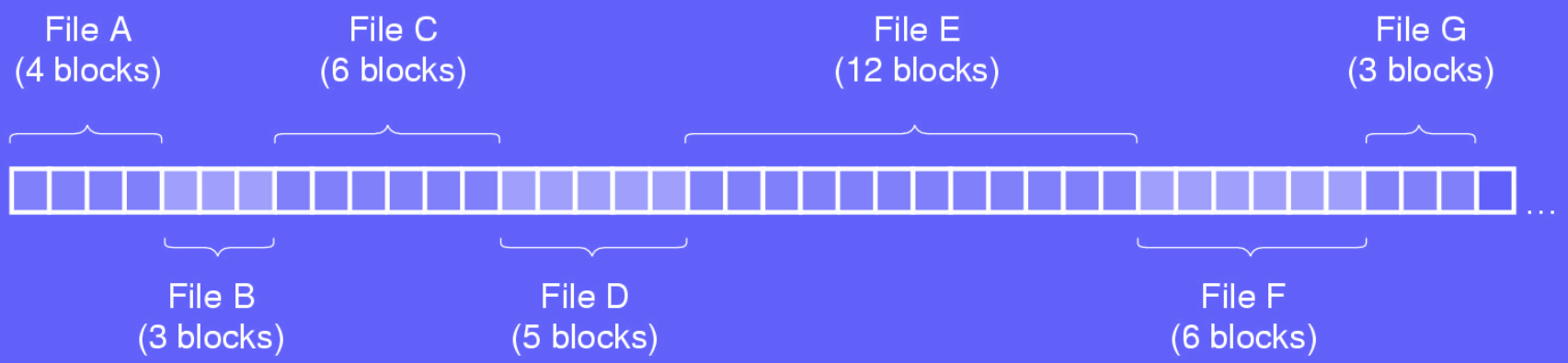
Contiguous allocation



Advantages:

- Simple to implement
- Fast: read a file using a single disk operation

Contiguous allocation

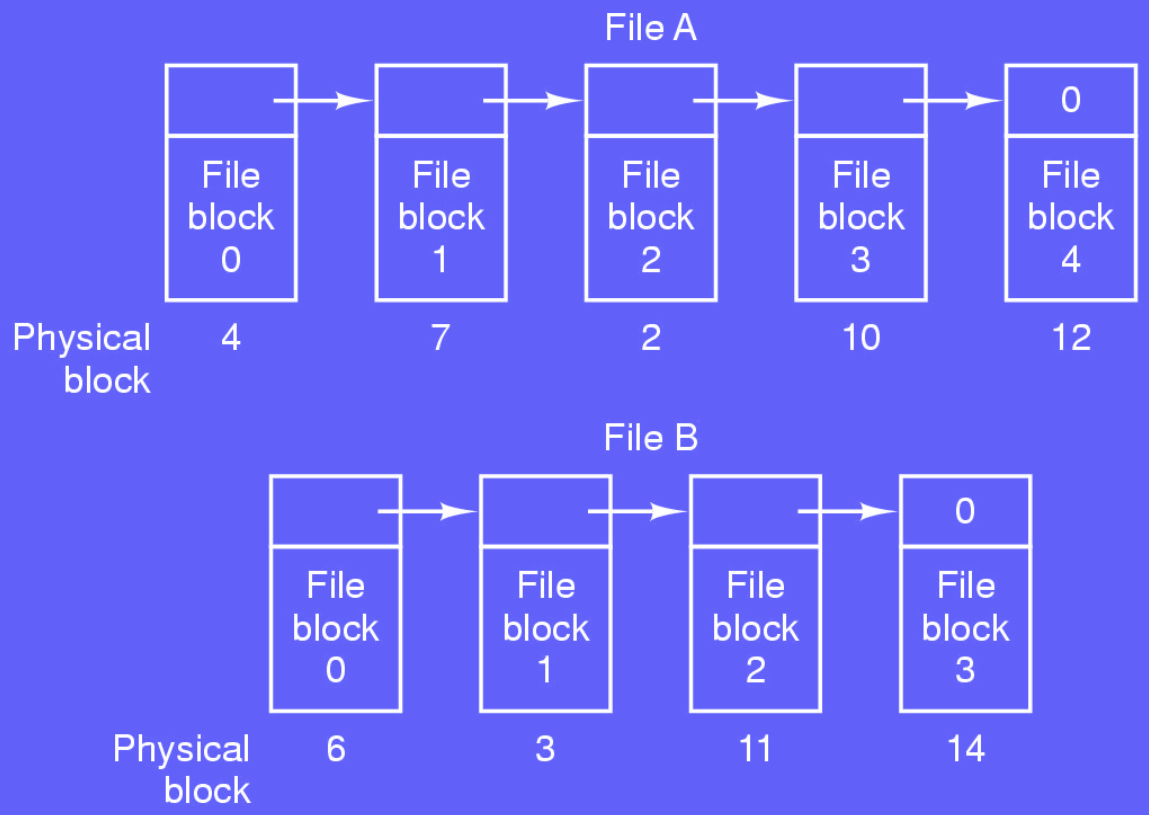


Advantages:

- Simple to implement
- Fast: read a file using a single disk operation

What if files D and F are deleted?

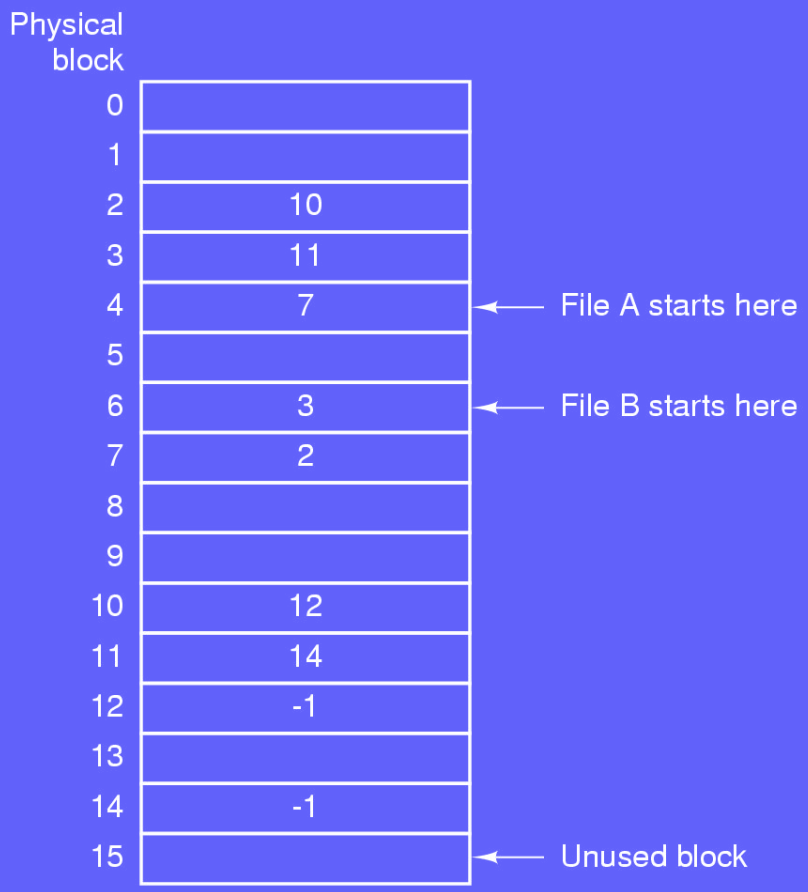
Linked list



Advantage: no fragmentation

How fast is random access?

File Allocation Table



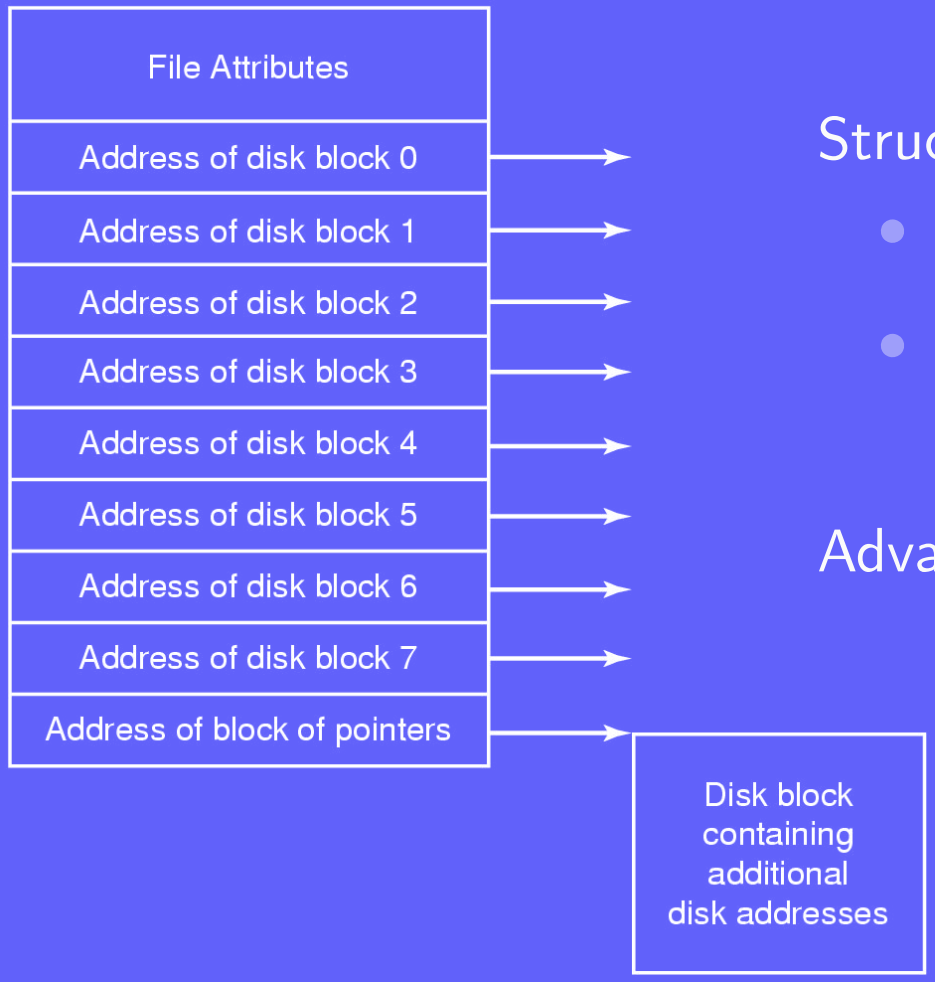
Basic idea:

- Have a pointer for each disk block
- Store all the pointers in a table
- Save the table in the RAM

Advantage: fast random access

What can be said about the memory usage?

Index node

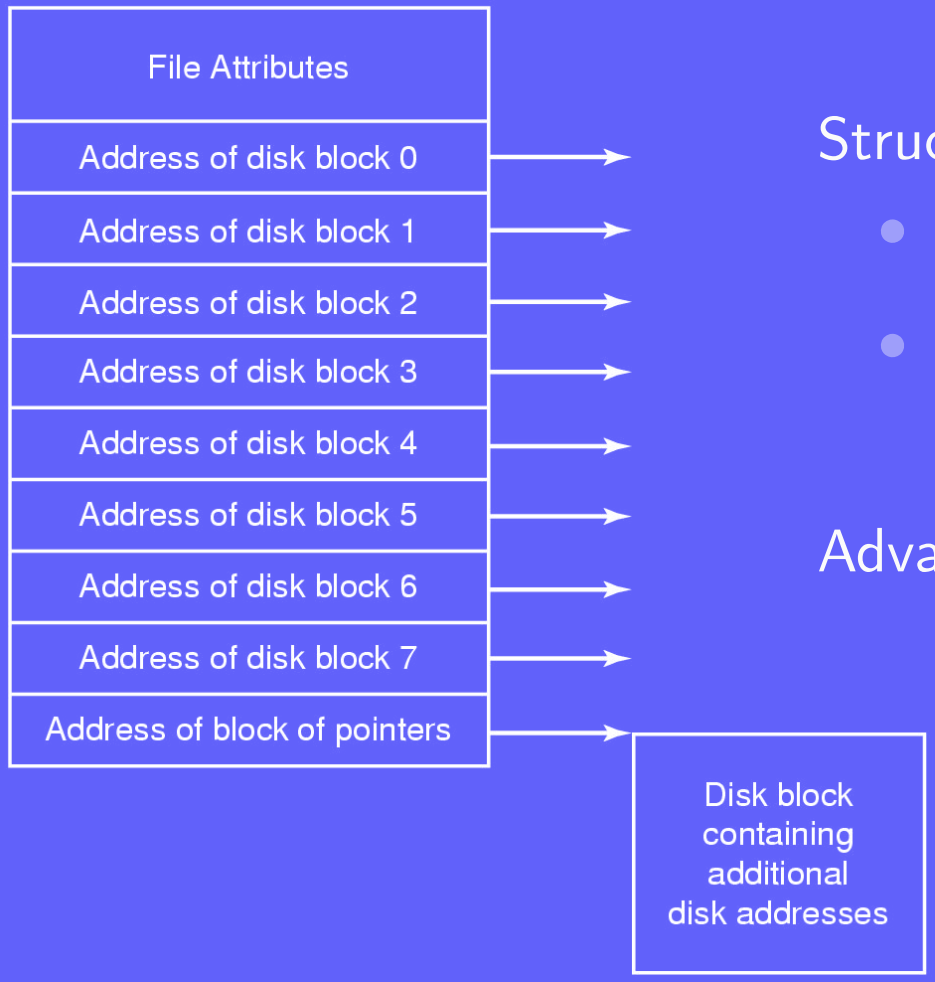


Structure storing:

- The file attributes
- Pointers on the blocks where the file is written

Advantage: fast, requires little memory

Index node



Structure storing:

- The file attributes
- Pointers on the blocks where the file is written

Advantage: fast, requires little memory

What if a large file needs more blocks that can fit in an inode?

Directories

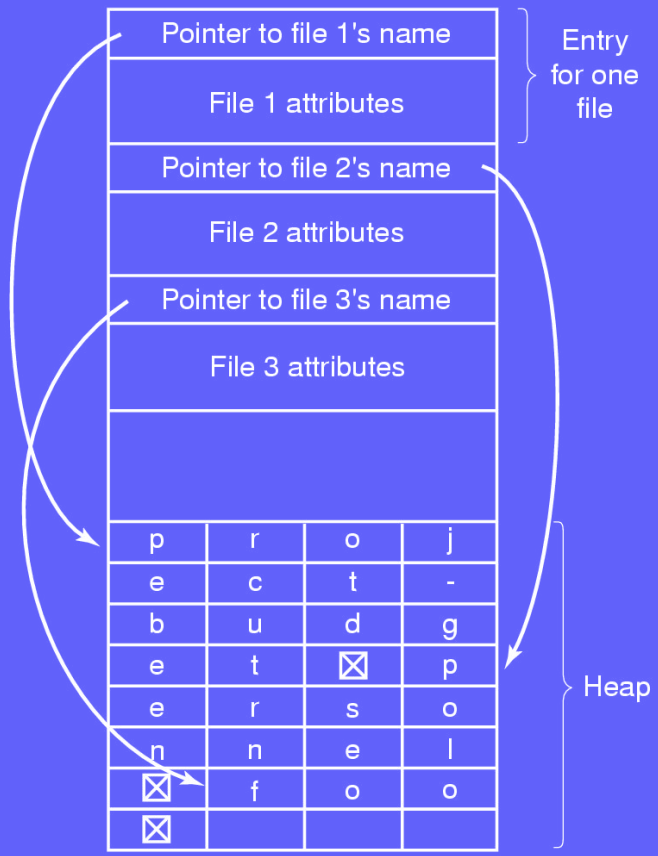
What is the issue with the following strategies?

- Fixed filename size
- Arbitrary long filename size

Directories

What is the issue with the following strategies?

- Fixed filename size
- Arbitrary long filename size



Alternative idea: string pointer

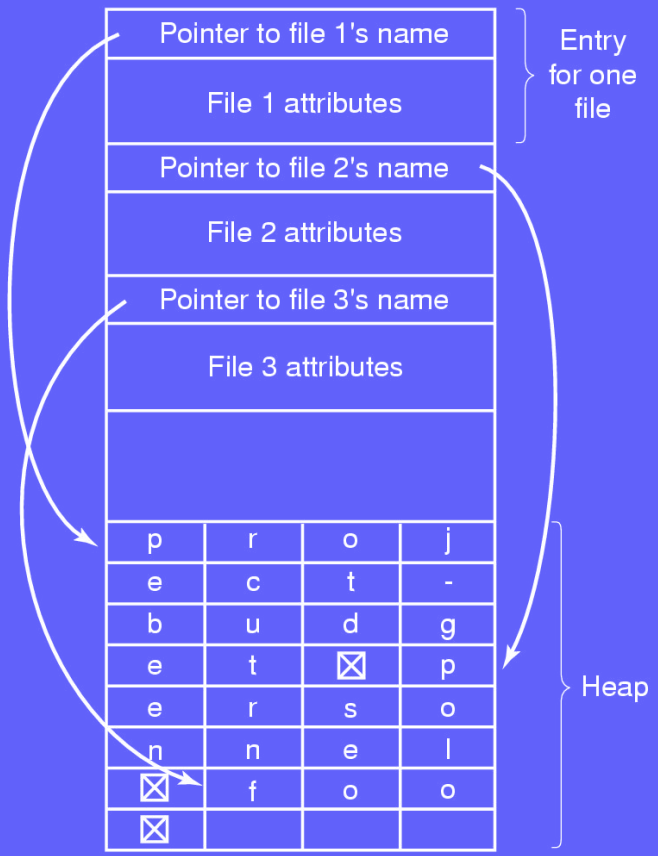
Advantages:

- No space wasted
- Space can be easily reused when a file is removed

Directories

What is the issue with the following strategies?

- Fixed filename size
- Arbitrary long filename size



Alternative idea: string pointer

Advantages:

- No space wasted
- Space can be easily reused when a file is removed

How fast is this strategy?

Directories

Journaling FS

Keep a journal of the operations:

- Log the operation to be performed, run it, and erase the log
- If a crash interrupts an operation, re-run it on next boot

Can any operation be applied more than once?

Journaling FS

Keep a journal of the operations:

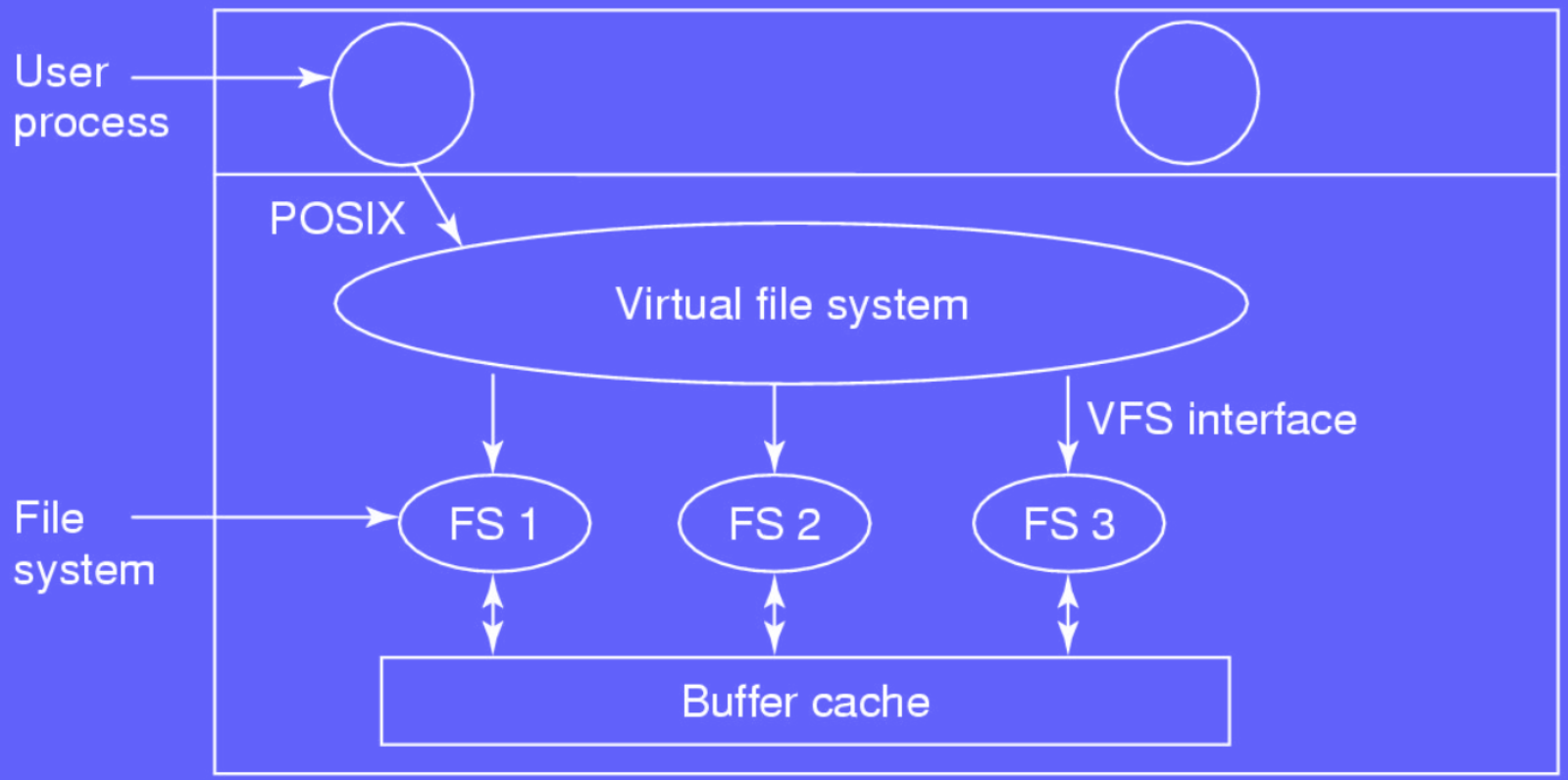
- Log the operation to be performed, run it, and erase the log
- If a crash interrupts an operation, re-run it on next boot

Can any operation be applied more than once?

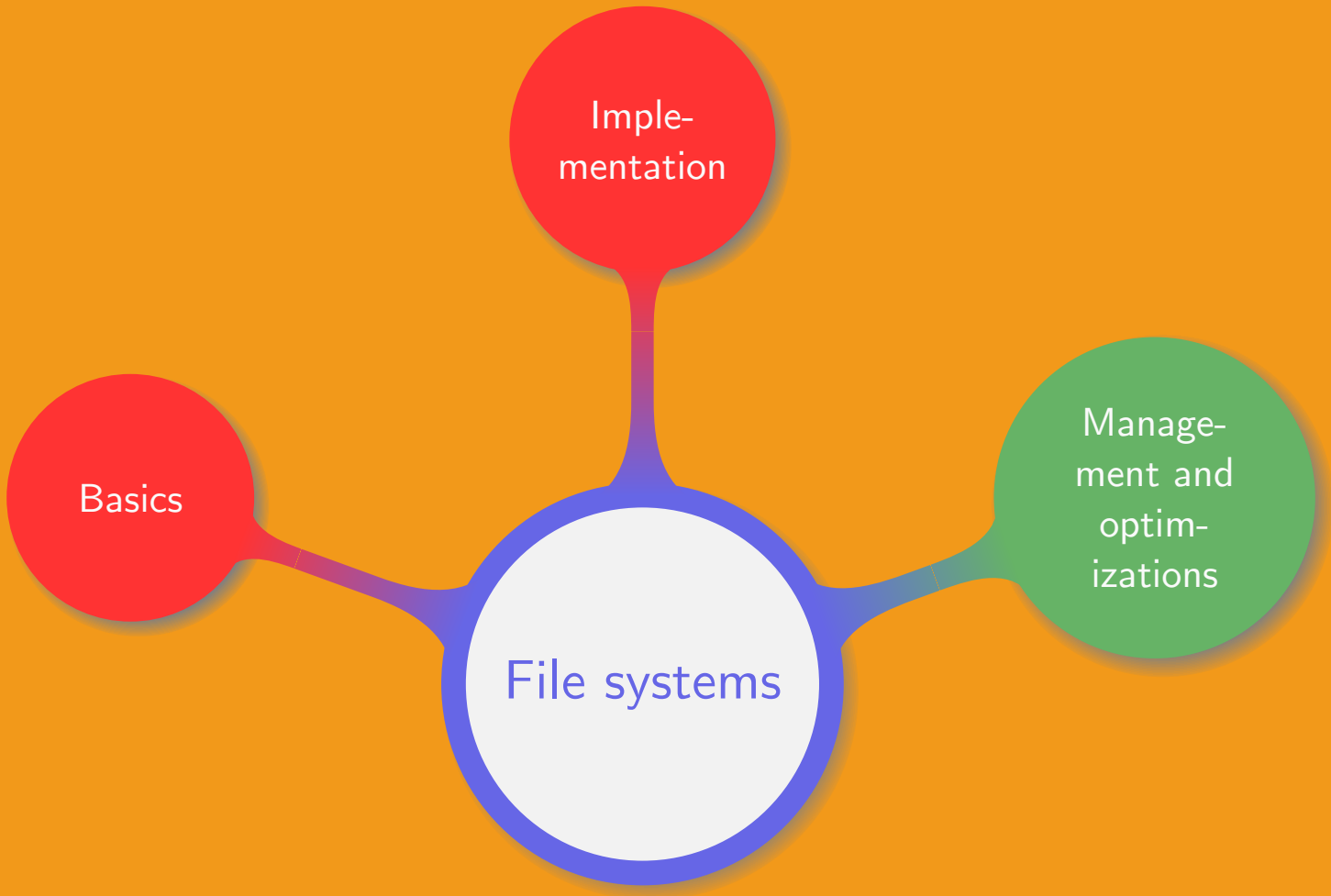
Example. Which of the following operations can be safely applied more than once?

- Remove a file from a directory,
- Release a file inode
- Add a file disk blocks to the list of free blocks

Virtual FS


User process
POSIX
Virtual file system
VFS interface
File system
FS 1
FS 2
FS 3
Buffer cache

Chapter organisation



Block size

Using small blocks:

- Large files use many blocks
- Blocks are not contiguous

A small block size leads to a waste of time

Block size

Using small blocks:

- Large files use many blocks
- Blocks are not contiguous

A small block size leads to a waste of time

Using large blocks:

- Small files do not fill up the blocks
- Many blocks partially empty

A large block size leads to a waste of space

Free blocks

Keeping track of the free blocks:

- Using a linked list: free blocks addresses are stored in a block
  e.g. using 4KB blocks with 64 bits block address, how many free blocks addresses can be stored in a block?
- Using a bitmap: one bit corresponds to one free block
- Using consecutive free blocks: a starting block and the number of free block following it

Which strategy is best?

File system consistency

Checking an inode based FS:

- Using the inodes: list all the blocks used by all the files and compare the complementary to the list of free blocks
- For every inode in every directory increment a counter by 1 and compare those numbers with the counts stored in the inodes

File system consistency

Checking an inode based FS:

- Using the inodes: list all the blocks used by all the files and compare the complementary to the list of free blocks
- For every inode in every directory increment a counter by 1 and compare those numbers with the counts stored in the inodes

Common problems and solutions:

- Block related inconsistency:
  - List of free blocks is missing some blocks: add blocks to list
  - Free blocks appear more than once in list: remove duplicates
  - A block is present in several files: copy block and add it to the files
- File related inconsistency:
  - Count in inode is higher: set link count to accurate value
  - Count in inode is lower: set link count to accurate value

Caching

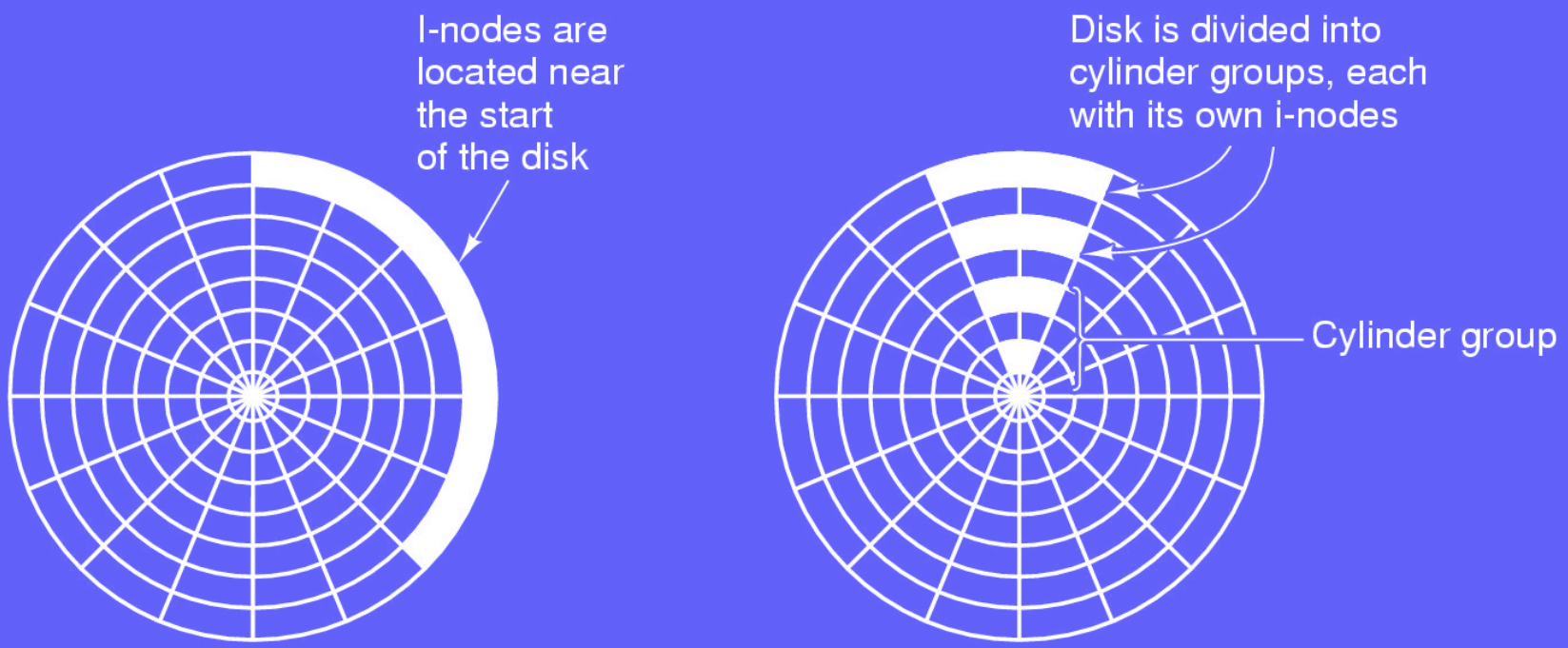
Keep in memory some disk blocks using the LRU algorithm:

- Is a block likely to be reused soon?
- What happens on a crash?

Refined idea:

- Useless to cache inode blocks
- Dangerous to cache blocks essential to file system consistency
- Cache partially full blocks that are being written

Arm motion


I-nodes are
located near
the start
of the disk
Disk is divided into
cylinder groups, each
with its own i-nodes
Cylinder group

Notes

A few extra remarks related to file systems:

- Quotas: assign disk quotas to users
- Fragmentation: how useful is it to defragment a file system?
- Block read ahead: when reading block k assume $k+1$ will soon be needed and ensure its presence in the cache
- Logical volumes: file system over several disks
- Backups: how to efficiently backup a whole file system?
- RAID: Redundant Arrays of Inexpensive Disks

Key points

- What are the three main goals of a file system?
- Describe a basic disk layout
- Explain the structure of an inode
- Mention three challenges in the design of a file system

Thank you!