!"#$%&’"()*%+,(-./0’1(!"1*%*/02$34%(

-+6:’&0(;<(8*7’(/./0’1/(=+%"$1’%0$(7/

5$674(!6$"(8*"$794(

!"#$%&$"’(&)*+"*,(-"(."%/$*,(0)%’123$*4*56"3&%7(.3$*

58&"*&"’$*8"*#9:6.3$*:$;)*<.3"(3.$=*

>%"$2?"*>)’’)(8*@ABC>BDE*FGH

Index (Getting started)

- **Introduction:**
  - Devices.
  - Basic aspects about File Systems.
  - Partitions, Mount/Umount.
- **FAT File System.**
- **EXT File System:**
  - I-nodes and blocks.
  - Block groups (ext2).
  - Journaling (ext3).
  - Extents and B-Trees (ext4).
- **Virtual File System.**
- **Administration.**

Introduction

- Previous concept: **Device:**
  - Name assigned to an I/O device, physical (disk, tape, sound card…) or logic (terminal, network port…).
  - **Device file:** file for app-HW interactions (through the kernel):
    - Consistent way to access different devices (same group of commands):
      - $ cat /dev/dsp > my_recording [talk] $cat my_recording > /dev/dsp.
      - cat /dev/input/mouse0.
    - Every device file can be located in directory **/dev:**
      - Standard devices [stdin, stdout, stderr] and Memory: [mem] (and virtual memory: kmem).
      - Specials: [null] (garbage), [zero] (zero generator), [random] (random number generator)…
      - Virtual terminals [ttyX] , Parallel and serial ports [lpX, ttySX], Optical devices [cdrom].
      - **IDE devices [hdXX], USB/SCSI/SATA devices [sdXX], RAID devices [mdX]** (or mapper/XXXX).
  - **Device driver:** kernel routines which define how to perform communication between kernel and HW (Interruptions, DMA…).

Introduction

- **Disk Partition:**
  - Logical storage unit which allows treating a single physical device as multiple ones, allowing a different File system on each partition.
  - High utility for administration tasks:
    - Protects directories that tend to grow indefinitely:
      - /var/spool against “mailbombing”.
      - /tmp against careless users/apps.
    - Divides software and users:
      - Easier upgrading, avoid users blocking the system.
  - In recent kernels, the system creates alias for each partition:
    - Can be employed whenever needed (Loader configuration, mounting, etc.).
    - In /dev/disk/{disk-by-uuiid}, links to the corresponding /dev/sdXX.
    - Persistent device naming: avoids changes in device naming at each boot.

Introduction

- **Disk Partition, mount/umount:**
  - **Mounting** process provides access to the content of a disk from the file system (making use of device file).
  - Can be done for any storage device (USB, CDROM, tape…).
  - At least one partition (system) is mounted during booting process.
  - Command **mount:** mount a file system:
    - Syntax: mount -<options> [file-dev] [mnt-point]:
      - Option –r: mounting in read-only mode.
      - Option –t: kind of file system mounted.
      - Example: mount –t ext3 /dev/hdc1 /home/.
  - **Umount** process disconnects the device from the rest of the system.
  - Command **umount** (syntax: umount [mnt-point]):
    - Doing this requires that no process is making use of the file system to umount.
    - Command fuser shows the processes making use of it.

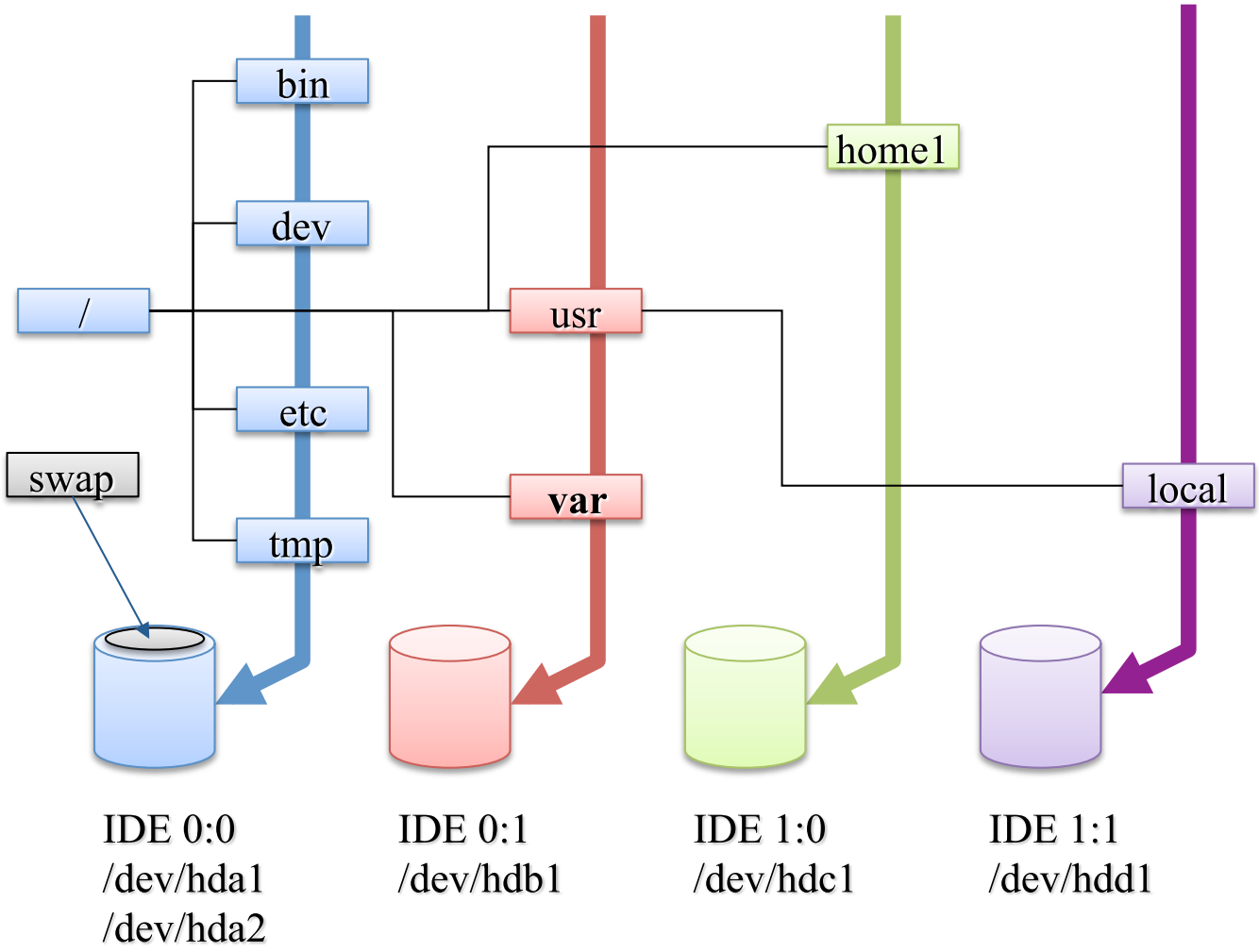
Introduction

```
# /etc/fstab: static file system information.
#
#<file sys>  <mount point>   <type>      <options>          <dump>
<pass>
proc         /proc          proc        defaults           0       0
/dev/sda1    /              ext3        errors=remount-ro  0       1
/dev/sda5    none           swap        sw                 0       0
/dev/hdc     /media/cdrom0  udf,iso9660 user,noauto        0       0
/dev/fd0     /media/floppy0 auto        rw,user,noauto     0       0
```

- **Automatic mount/umount:**
  - Systems to mount/umount are read from file **/etc/fstab**.
  - Done automatically during boot process (can also be performed at a different moment with command “mount –a”).
  - File /etc/fstab:
    - **<file sys>:** device file.
    - **<mount point>:** mount point (directory).
    - **<type>:** type of file system (ext3, ext4, vfat, xfs...).
    - **<options>:** read or read/write mode (ro/rw), SUID/SGID support (suid/nosuid), allow user mounting (user/nouser), allow binary execution (exec/noexec)…
    - **<dump>:** dump frequency (backup utility, obsolete).
    - **<pass>:** order to run fsck on the device. Run at boot time if an illegal umount is performed for that device (power button).

Introduction

- **Example: File system in multiple partitions/disks.**

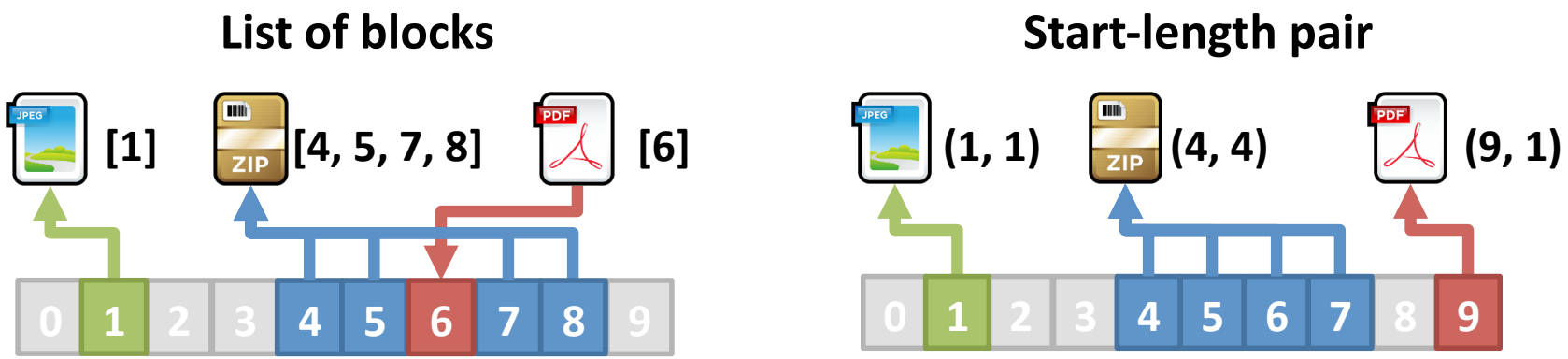


Introduction

- **The File System:**
  - What are the main requirements for a file system?:
    - Labeled files (with name).
    - File organization as a linked hierarchy (tree-like) of directories.
    - Meta-data for every file (generation time, permission, etc.).
  - How is this implemented? (File Storage):
    - Disk performs sequential storage (blocks), does not know about hierarchies.

Introduction

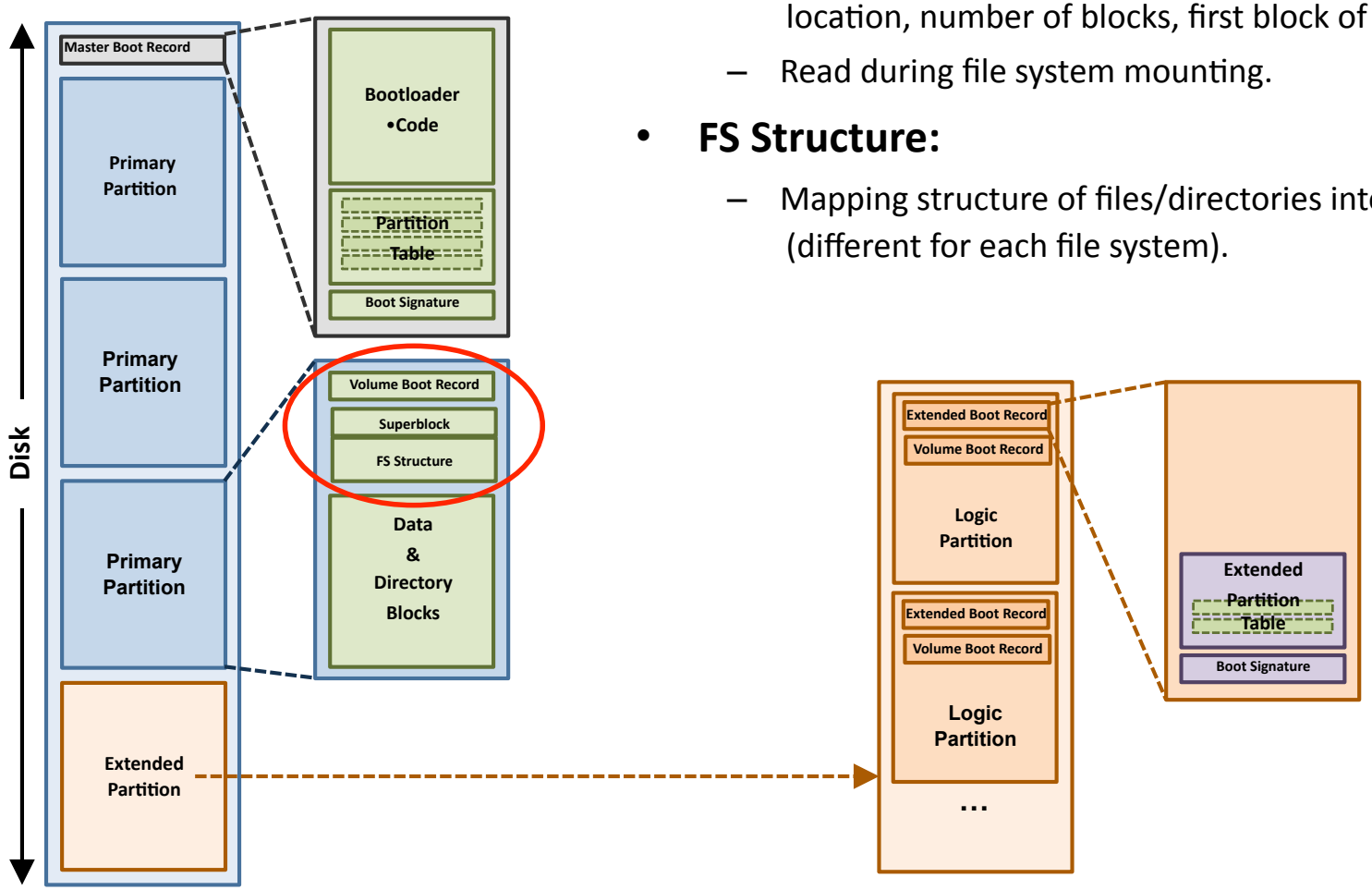
- File storage:
  - A file has two main components:
    - Data: one or more disk blocks with binary information.
    - Metadata: name, size, permission, directory, **block mapping**…
  - Any file is stored in at least 1 disk block:
    - How can I map files in multiple blocks?



  - FAT, EXT1 EXT2 and EXT3 employ list of blocks, EXT4 start-length pair.

Introduction

- **Where is FS info stored?:**



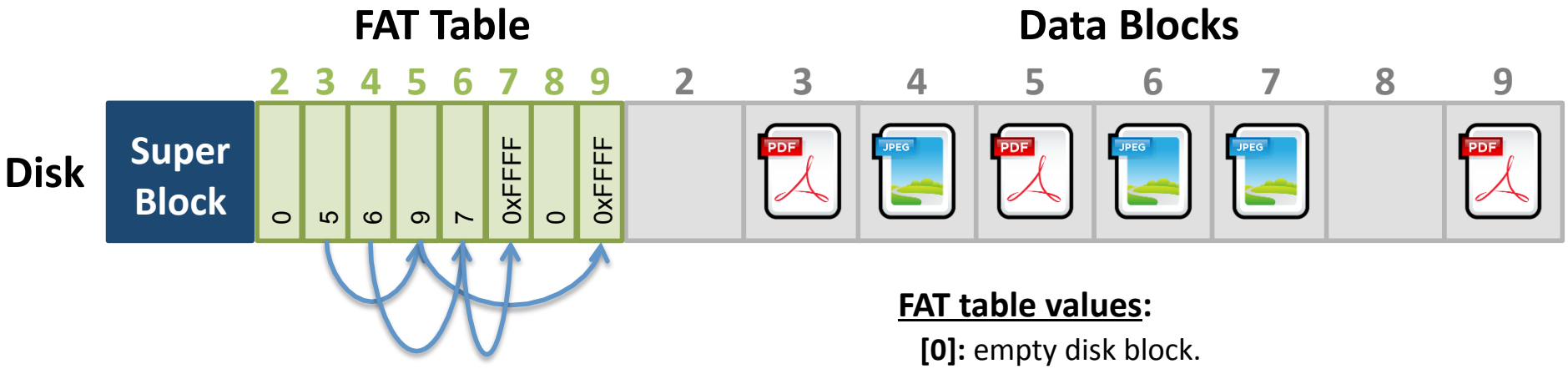
- **Superblock:**
  - Keeps information about file system (version, boot file location, number of blocks, first block of / directory…).
  - Read during file system mounting.
- **FS Structure:**
  - Mapping structure of files/directories into blocks (different for each file system).

Index (Getting started)

- **Introduction:**
  - Devices.
  - Basic aspects about File Systems.
  - Partitions, Mount/Umount.
- **FAT File System.**
- **EXT File System:**
  - I-nodes and blocks.
  - Block groups (ext2).
  - Journaling (ext3).
  - Extents and B-Trees (ext4).
- **Virtual File System.**
- **Administration.**

FAT File System

- FAT: File Allocation Tables:
  - File system created in 1977 and popularized thanks to MS-DOS.
  - Still popular today (FAT32): USB, mem cards, EFI boot partition.
  - **File Allocation Table**: Linked list structure that holds information about the blocks occupied by each file.
  - It also determines whether a block is in use or not.



FAT table values:

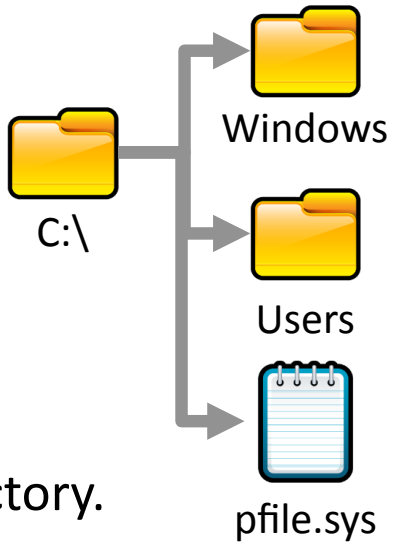
[0]: empty disk block.

[1]: reserved by the O.S.

[1<N<0xFFFF]: next block in the list.

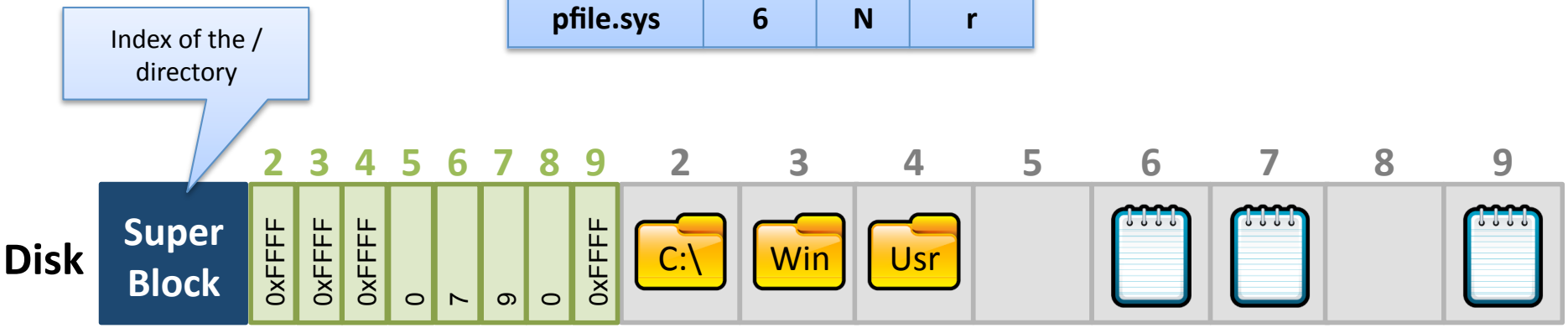
[0xFFFF]: end of the list.

FAT File System



- Directories:
  - Treated as special files.
  - It is a file containing a list with the elements in the directory.

Name	Index	Dir?	Perm
.	2	Y	rwx
Windows	3	Y	rwx
Users	4	Y	rwx
pfile.sys	6	N	r



FAT File System

- Problems/Limitations:
  - Upper limit, FAT32 supports a maximum disk size of 2TB.
  - Locating free blocks requires scanning the whole FAT table.
  - Prone to file fragmentation (poor locality in blocks from the same file). Metadata fragmentation -> very expensive searches.
  - Linked lists are not efficient in the presence of small files (a 4-block file requires 4 readings of the FAT).
- What is the common case, small or big files?:
  - Seems to be small ones: 2KB is the most common size, 200KB the average size.
- Make use of more efficient structures: i-nodes (index node):
  - Employed in the Linux file system.

Index (Getting started)

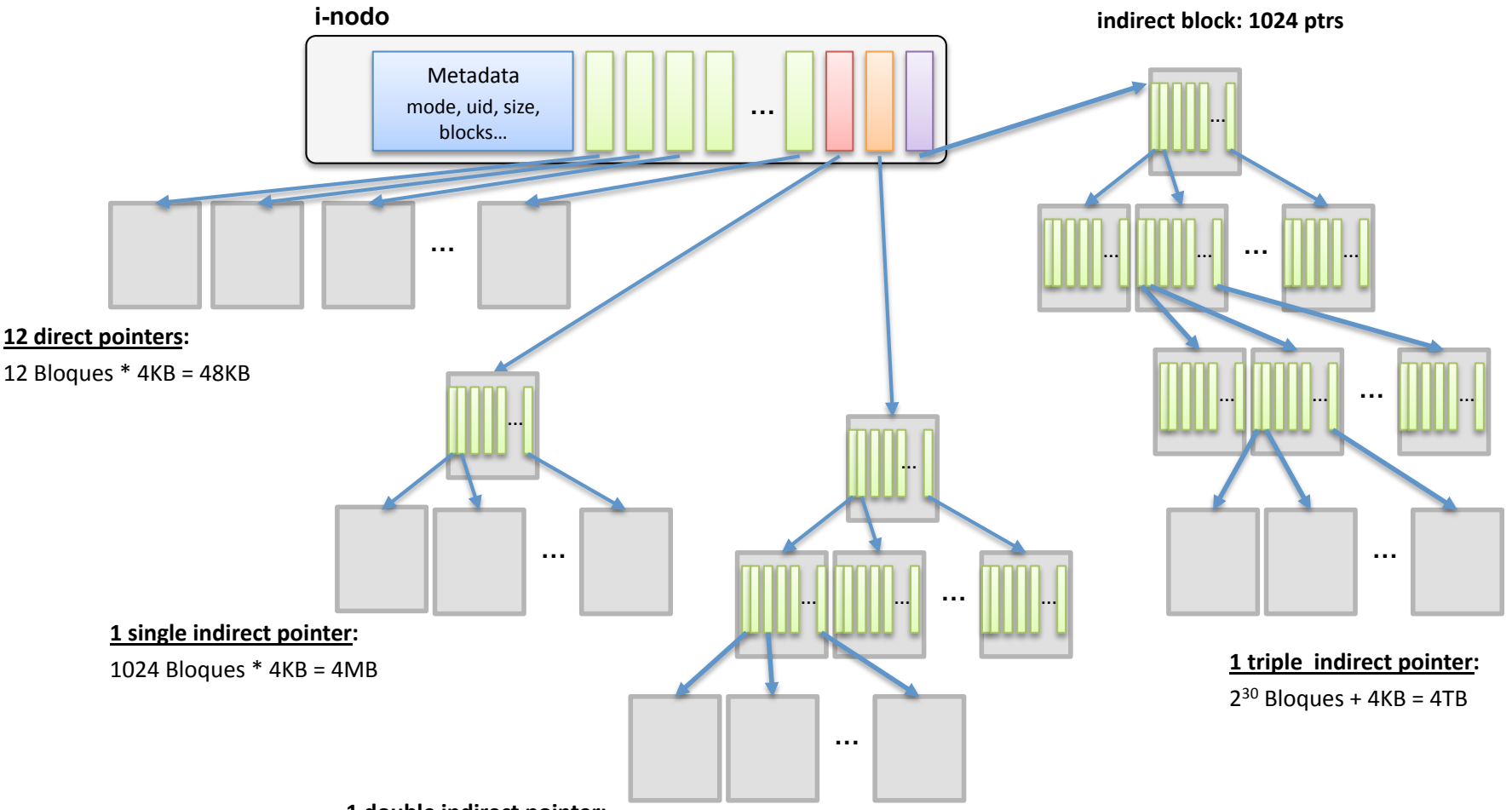
- **Introduction:**
  - Devices.
  - Basic aspects about File Systems.
  - Partitions, Mount/Umount.
- **FAT File System.**
- **EXT File System:**
  - I-nodes and blocks.
  - Block groups (ext2).
  - Journaling (ext3).
  - Extents and B-Trees (ext4).
- **Virtual File System.**
- **Administration.**

EXT File System

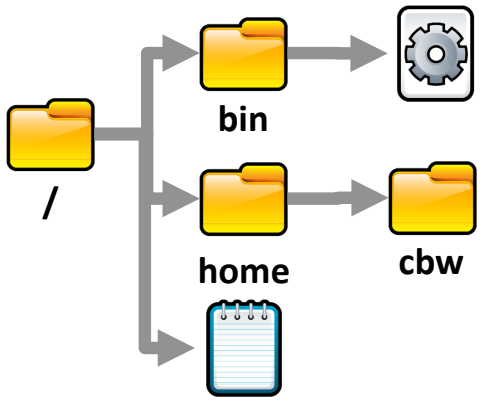
- The i-nodes:
  - Basic building element of the file system.
  - Each file (or directory) has at least one i-node associated.
  - By default, they consume 10% of disk storage (can be configured at FS creation time).

EXT File System

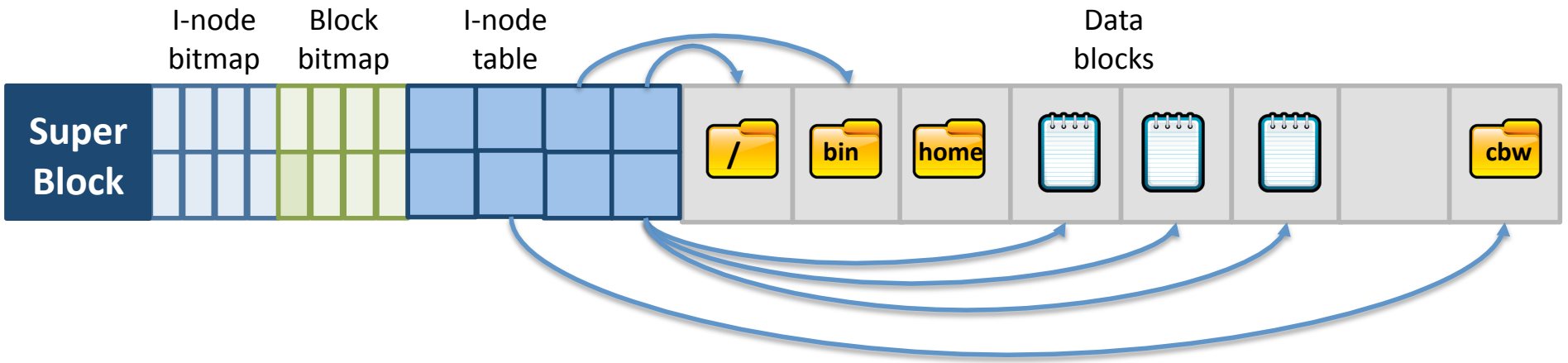
- The i-nodes:



EXT File System

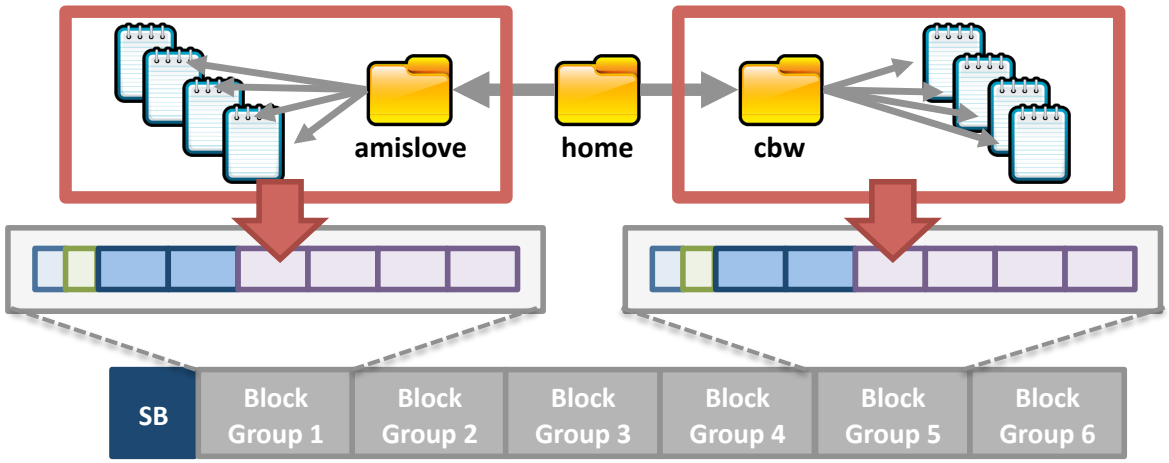


- EXT File System structure:
  - I-node bitmap: bit map of occupied/free i-nodes.
  - Block bitmap: bit map of occupied/free blocks.
  - I-node table: each input is a single i-node.



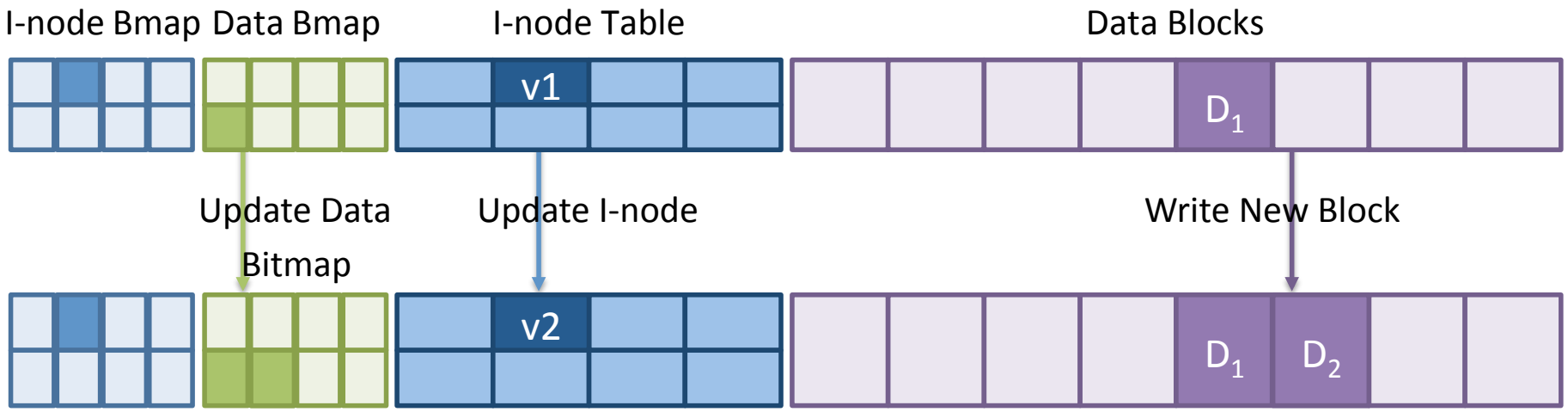
EXT File System (ext2)

- Problems/Limitations of EXT:
  - Less fragmentation of metadata, but data fragmentation still present.
  - I-nodes and their associated data can be far away in the disk.
- ext2 improves data-metadata locality:
  - Disk is divided into block groups (group size usually depends on disk physical properties: cylinder size).
  - Each group replicates FS structures: inode/data bitmap, inode table.



EXT File System (ext3)

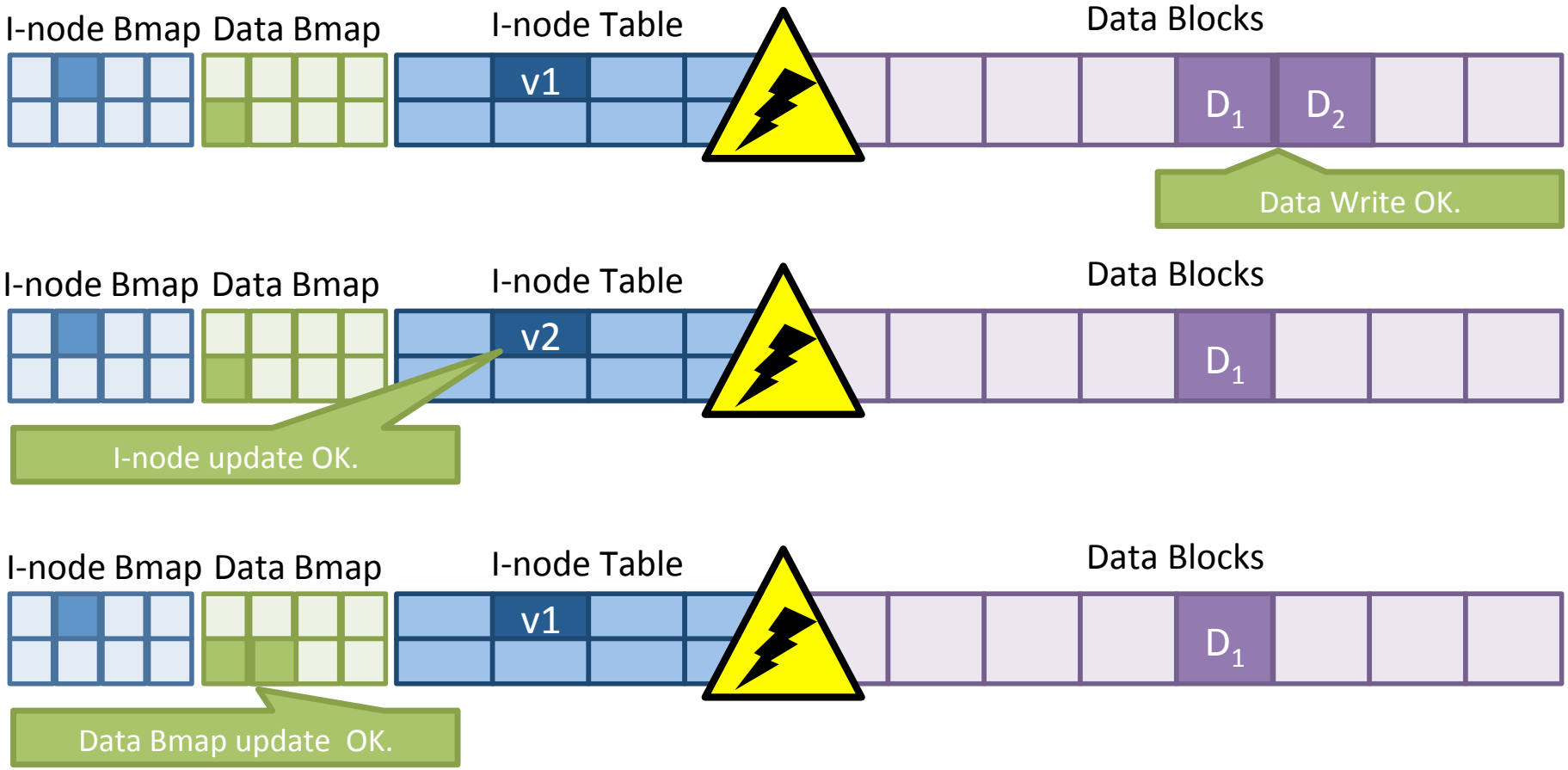
- Consistency of the file system:
  - Some operations require multiple and independent write operations in the file system.
  - Example: add a block to an existing file (size increase).



- Operations performed in random order:
  - What happens if the process is interrupted at an intermediate point?

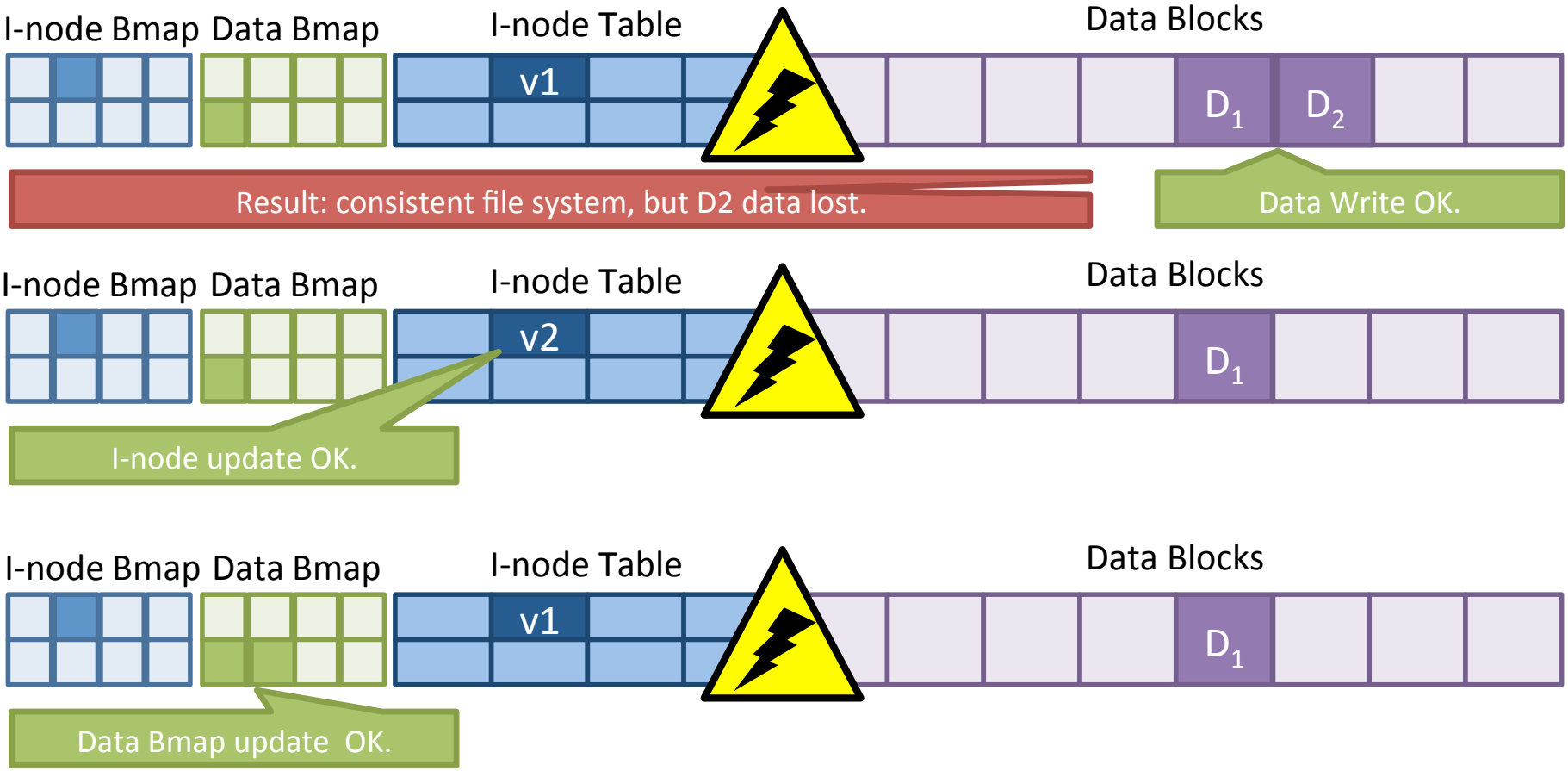
EXT File System (ext3)

- Consistency of the file system:



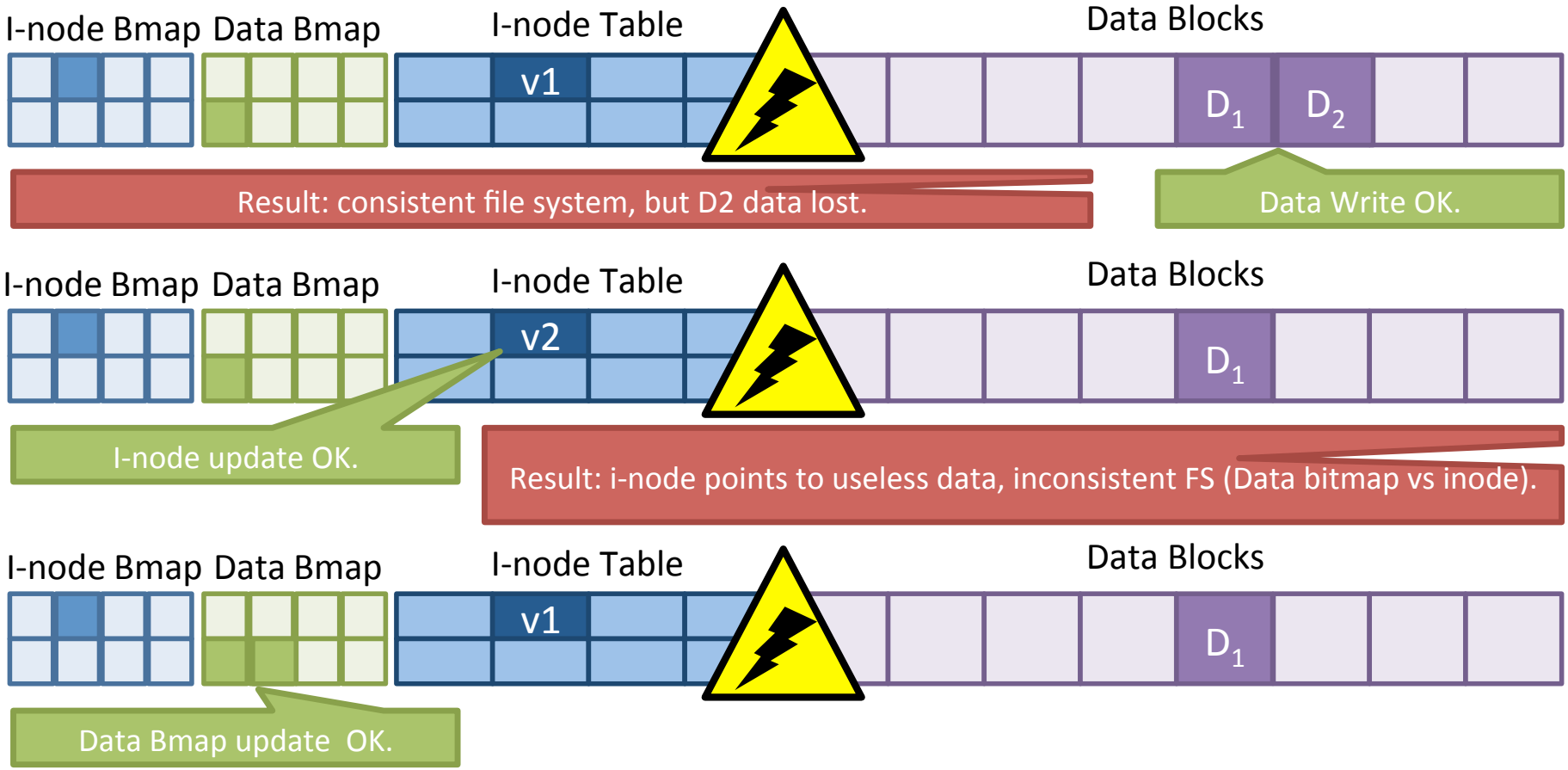
EXT File System (ext3)

- Consistency of the file system:



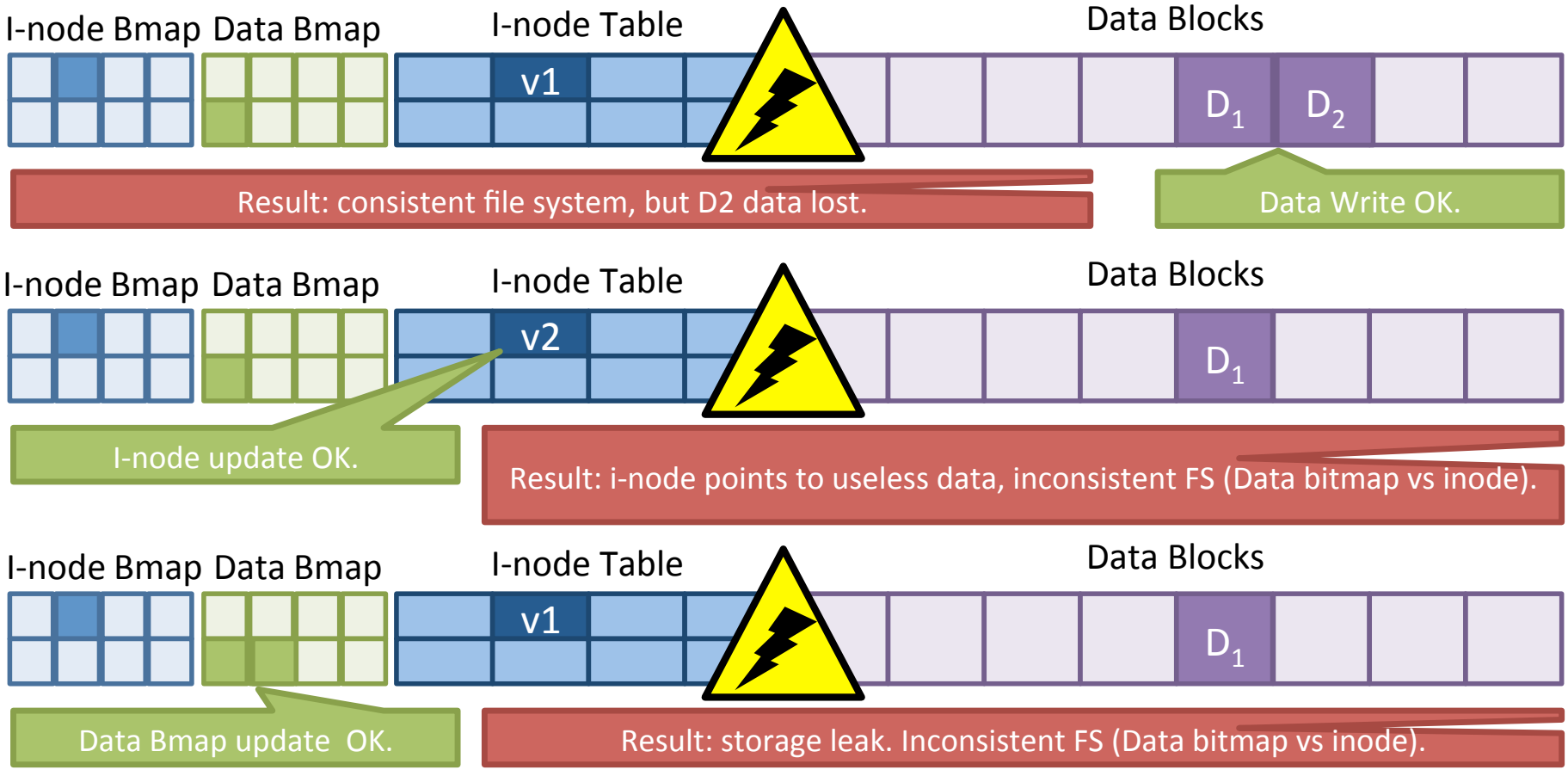
EXT File System (ext3)

- Consistency of the file system:



EXT File System (ext3)

- Consistency of the file system:



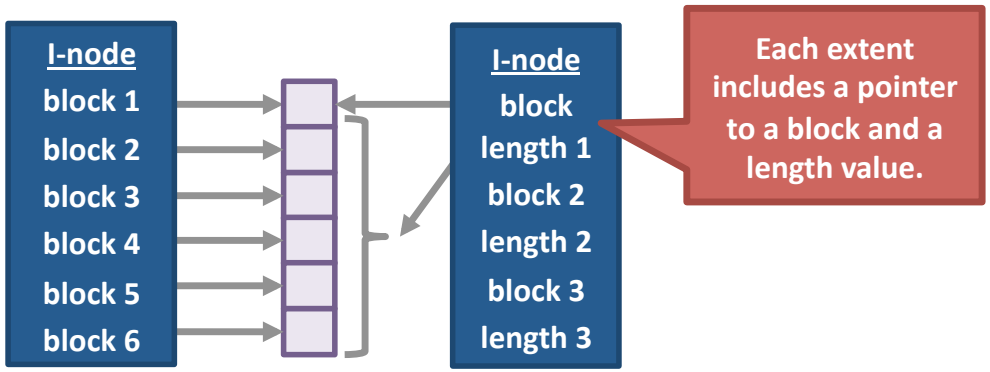
EXT File System (ext3)

- Journaling:
  - Atomic pre-writing (at the same time) disk data.
  - Disk writes are pre-annotated in a log. Each input: journal.

  - What happens if log write is interrupted?:
    - Transaction is not completed (data lost) but the FS remains consistent.
  - What happens if journal is written correctly, but disk not?:
    - Temporarily, file system misses consistency.
    - The log has the information to restore it (during boot, unfinished journals are completed).
  - How do we improve performance?:
    - Buffering sequential writes in memory, grouping them as a single log.
    - Performing journaling only to Metadata (Data Bitmap + I-node).

EXT File System (ext4)

- Pointers vs Extents:
  - Inode pointers are not efficient for big files:
    - Example: a 100MB file requires 25600 pointers.
    - Cannot be avoided if no contiguous blocks, but what happens in the presence of locality?
  - Current file systems try to minimize data fragmentation:
    - Less searches, better performance.
    - Extents behave better in the case of files with adjacent blocks.



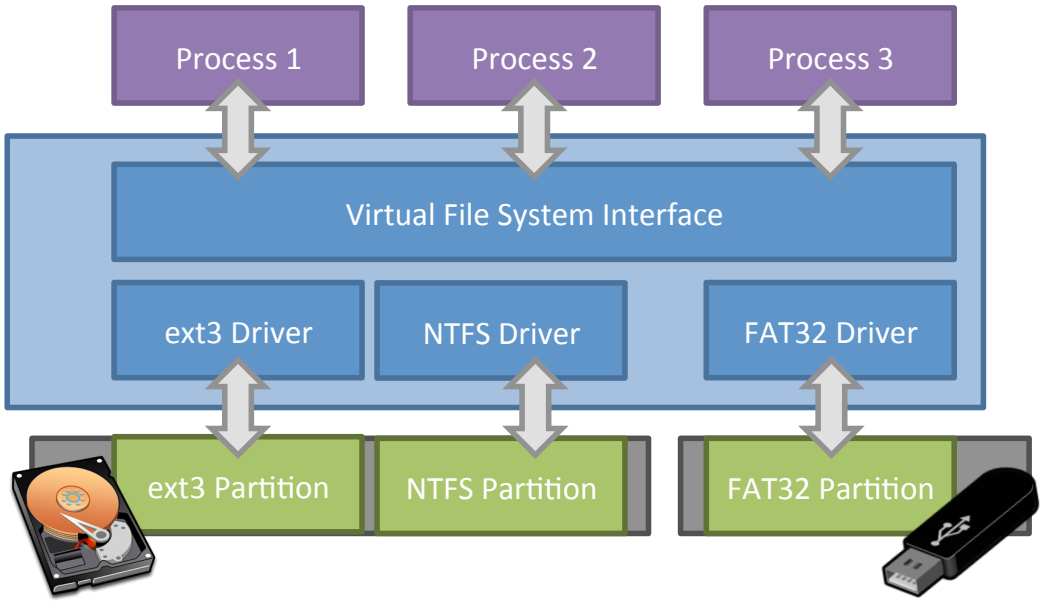
- Btrees:
  - Improved directory encoding to speed up file search.

Index (Getting started)

- **Introduction:**
  - Devices.
  - Basic aspects about File Systems.
  - Partitions, Mount/Umount.
- **FAT File System.**
- **EXT File System:**
  - I-nodes and blocks.
  - Block groups (ext2).
  - Journaling (ext3).
  - Extents and B-Trees (ext4).
- **Virtual File System.**
- **Administration.**

Virtual File System

- Problem:
  - The OS can mount multiple partitions with different file systems.
  - Does a process need to use different APIs for each FS?
- Linux makes use of an interface known as Virtual File System (VFS):
  - Exposes a POSIX API to the processes.
  - Re-sends requests to the specific driver of the underlying file system.



Index (Getting started)

- **Introduction:**
  - Devices.
  - Basic aspects about File Systems.
  - Partitions, Mount/Umount.
- **FAT File System.**
- **EXT File System:**
  - I-nodes and blocks.
  - Block groups (ext2).
  - Journaling (ext3).
  - Extents and B-Trees (ext4).
- **Virtual File System.**
- **Administration.**

Administration

- **Tasks** of system administrator on the file system:
  - Guarantee user access to local and remote File Systems.
  - Supervision and management of storage capacity.
  - Protect information against corruption, hw failures and user errors through periodic backups.
  - Guarantee data confidentiality.
  - Check File systems and repair possible corruptions.
  - Connect and configure new disks.

Administration

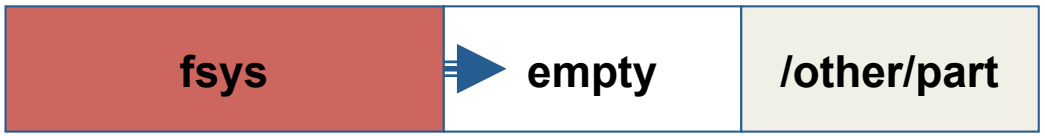
- **Adding** a new disk:
  - Command **fdisk**: manipulation of the partition table:
    - Syntax: fdisk /dev/sda (Includes a descriptive menu of the available operations [m]).
    - Think carefully about what you are doing ([q] exit without saving changes).
    - [v]: look at the content of a unpartitioned disk.
    - [n]: new partition.
    - [w]: write the new partition table (Prior revision with [p]).
  - BIOS limitations for a PC: only 4 primary partitions (the rest extended).
- **Formatting** the new disk:
  - File systems supported by the kernel: /proc/filesystems:
    - Most recommended in linux is: ext3/ext4.
  - Command **mkfs**: builds a file system in a partition:
    - Syntax: mkfs [-V –t fs-tipo] /dev/sda3.

Administration

- **Checking** the file system:
  - Command **fsck:** detection and correction (some cases) of corruption problems in the FS:
    - Compares the list of free blocks with the directions stored in the i-nodes.
    - It also verifies the list of free i-nodes in contrast to the i-nodes in directory inputs.
    - Important limitations against file corruption.
    - Should be performed without mounting the file system.
    - Periodically it is performed during the boot process.
  - Command **badblocks:** detect and exclude broken disk sectors:
    - Physical error, replace the disk immediately.
  - S.M.A.R.T.:
    - Utilities to access reliability/usage information about the disk (requires firmware support).
    - Smartmontools.

Administration

- **Resizing** the file system:
  - Command **resize2fs:**
    - Supports ext4 and requires kernel >= 2.6.
    - Adjacent partitions must allow it.



    - First make room with fdisk, then resize (increase) with resize2fs.
    - It is also useful to reduce the file system size:
      - Combined with fdisk we can do anything: break, increase, etc.
      - Before working with partition table, make a backup dd if=/dev/sda of=part.bkp count=1 bs=1.
    - Dangerous.
  - Command **parted**: manipulation of partition table and FS:
    - Syntax: parted /dev/sdX.
    - Can copy, move, change file systems, very powerful.
    - Dangerous if commands are not executed correctly!!

Administration

- Modify file system parameters:
  - Command **tune2fs:** adjust configurable parameters of the FS:
    - [-e] policy in the presence of error.
    - [-j] add journaling.

- Other tools: **dd:**
  - Images of the file system:
    - dd if=/dev/sda1 | gzip > imagen_disco.gz.
    - gzip –dc imagen_disco.gz | dd of=/dev/sda2.
  - Copy of the file system:
    - dd if=/dev/sda1 of=/dev/sda2.
  - Backup of the partition table:
    - dd if=/dev/sda1 of=backup_part count=512 bs=1.