Journaling
and
Log-structured file systems

Johan Montelius

KTH

2019

The file system

A file system is the user space implementation of *persistent storage*.

The file system

A file system is the user space implementation of *persistent storage*.

- a *file* is persistent i.e. it survives the termination of a process
- a *file* can be access by several processes i.e. a shared resource
- a *file* can be located given a *path* name

let's write to a file

Assume we want to write to a file `bar.txt`, that requires a new block to be allocated.

We need to:

- update the block bitmap - we have allocated one more data block

let's write to a file

Assume we want to write to a file `bar.txt`, that requires a new block to be allocated.

We need to:

- update the block bitmap - we have allocated one more data block
- update the inode of `bar.txt` - a new data block, size and access time

let's write to a file

Assume we want to write to a file `bar.txt`, that requires a new block to be allocated.

We need to:

- update the block bitmap - we have allocated one more data block
- update the inode of `bar.txt` - a new data block, size and access time
- update the block - the new data (it might contain old data).

let's write to a file

Assume we want to write to a file `bar.txt`, that requires a new block to be allocated.

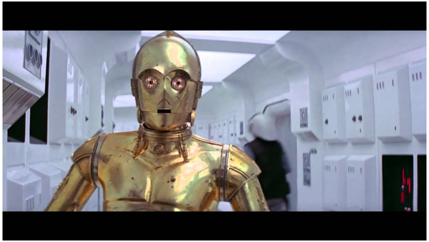
We need to:

- update the block bitmap - we have allocated one more data block
- update the inode of `bar.txt` - a new data block, size and access time
- update the block - the new data (it might contain old data).

In what order should we perform these operations?

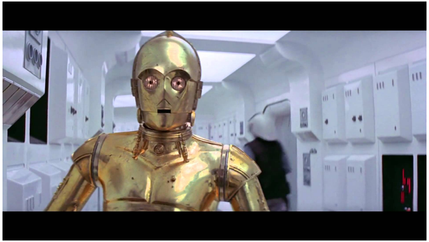
what if we crash

what if we crash



We’re doomed!

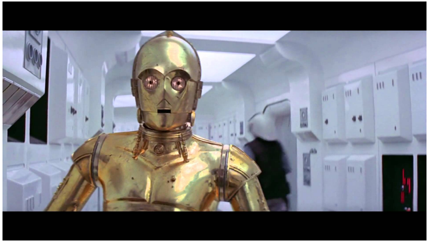
what if we crash



We're doomed!

How do we cope with crashing drives?

what if we crash



We're doomed!

How do we cope with crashing drives?

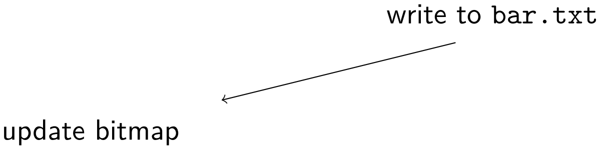
How do we cope with the operating system crashing?

one or two out of three

one or two out of three

write to `bar.txt`

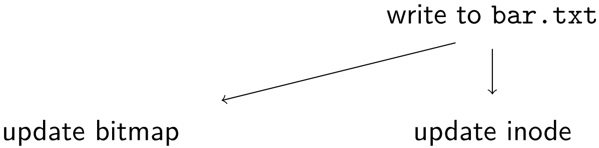
one or two out of three



write to `bar.txt`

update bitmap

one or two out of three

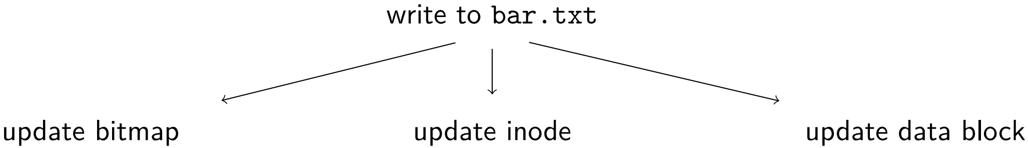


write to `bar.txt`

update bitmap

update inode

one or two out of three



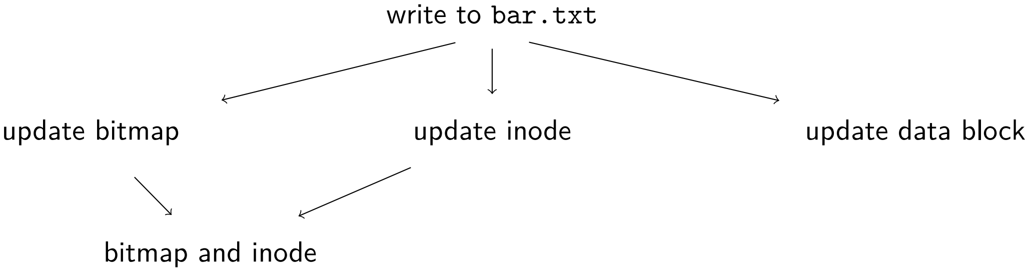
write to `bar.txt`

update bitmap

update inode

update data block

one or two out of three



write to `bar.txt`

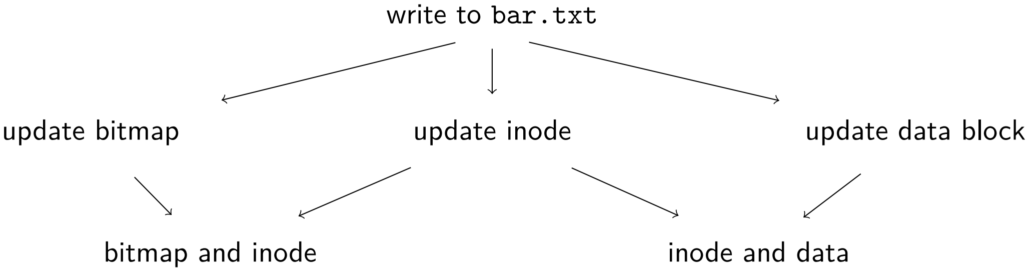
update bitmap

update inode

update data block

bitmap and inode

one or two out of three



write to `bar.txt`

update bitmap

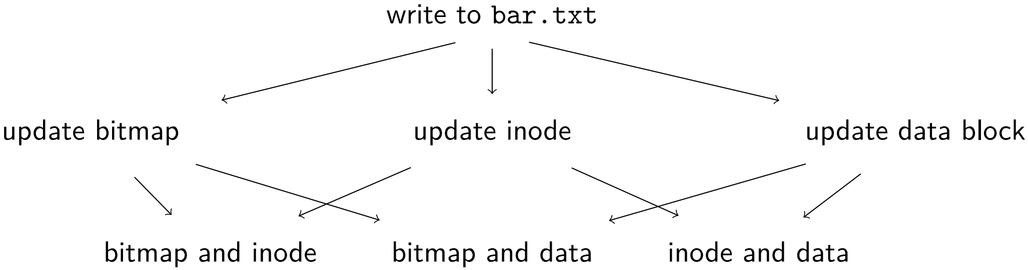
update inode

update data block

bitmap and inode

inode and data

one or two out of three



write to `bar.txt`

update bitmap

update inode

update data block

bitmap and inode

bitmap and data

inode and data

two out of three ...

two out of three ...

Two out of three is - when it comes to file systems - bad.

Survive a crash

Survive a crash

Approaches:

Survive a crash

Approaches:

- file system check - recover as much as possible

Survive a crash

Approaches:

- file system check - recover as much as possible
- journal - write down what you want to do, before you do it

Survive a crash

Approaches:

- file system check - recover as much as possible
- journal - write down what you want to do, before you do it
- log - the file system is a log of changes

Survive a crash

Approaches:

- file system check - recover as much as possible
- journal - write down what you want to do, before you do it
- log - the file system is a log of changes
- copy on write - create a perfect copy and flip a pointer

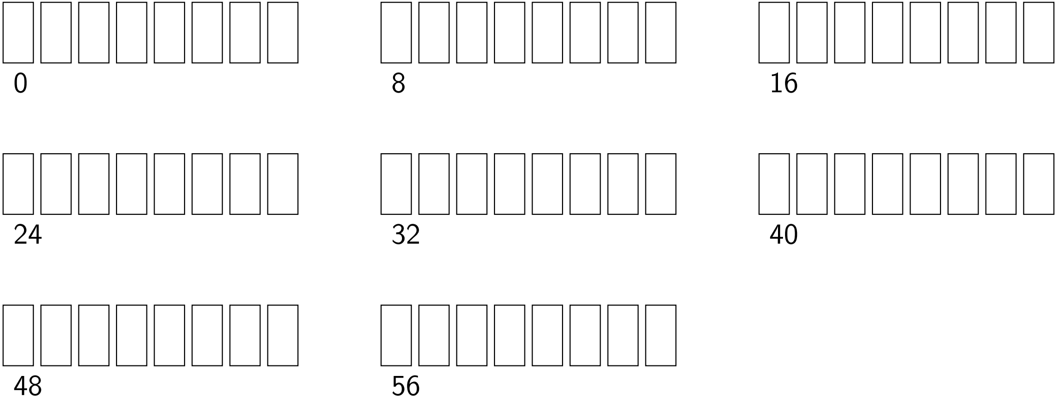
Survive a crash

Approaches:

- file system check - recover as much as possible
- journal - write down what you want to do, before you do it
- log - the file system is a log of changes
- copy on write - create a perfect copy and flip a pointer

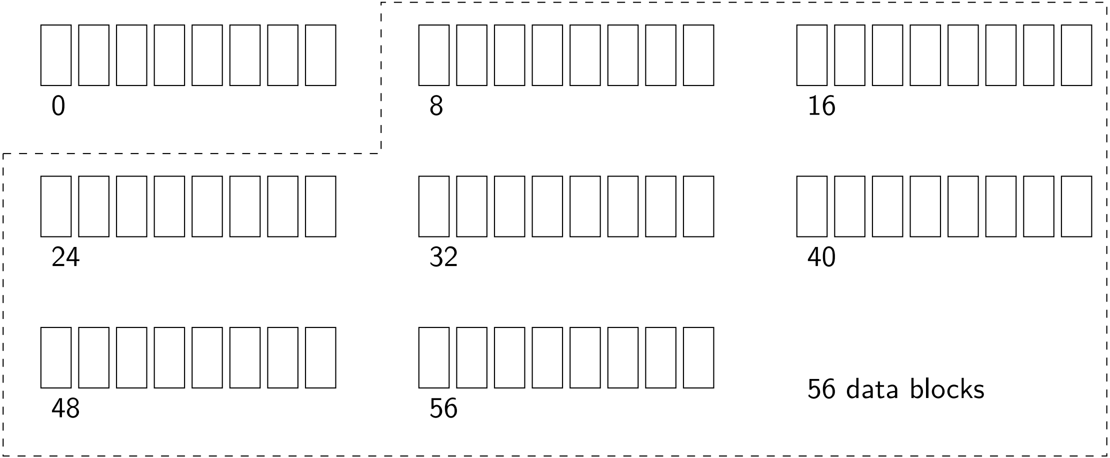
crash recovery

Remember the Very Simple File System:



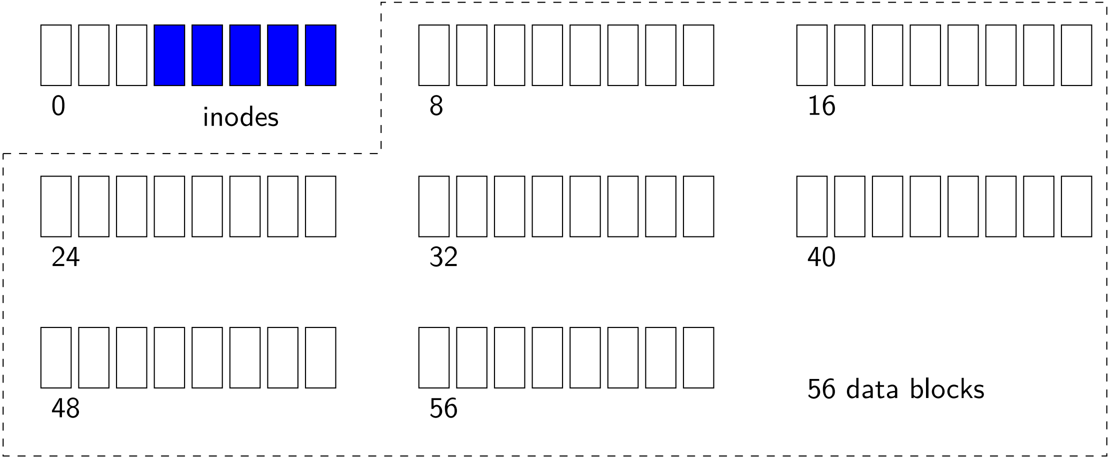
crash recovery

Remember the Very Simple File System:



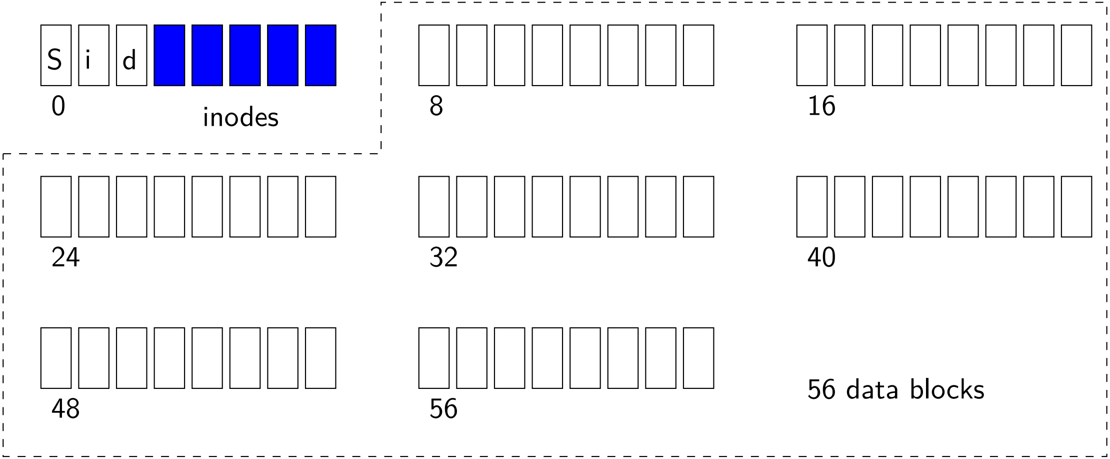
crash recovery

Remember the Very Simple File System:



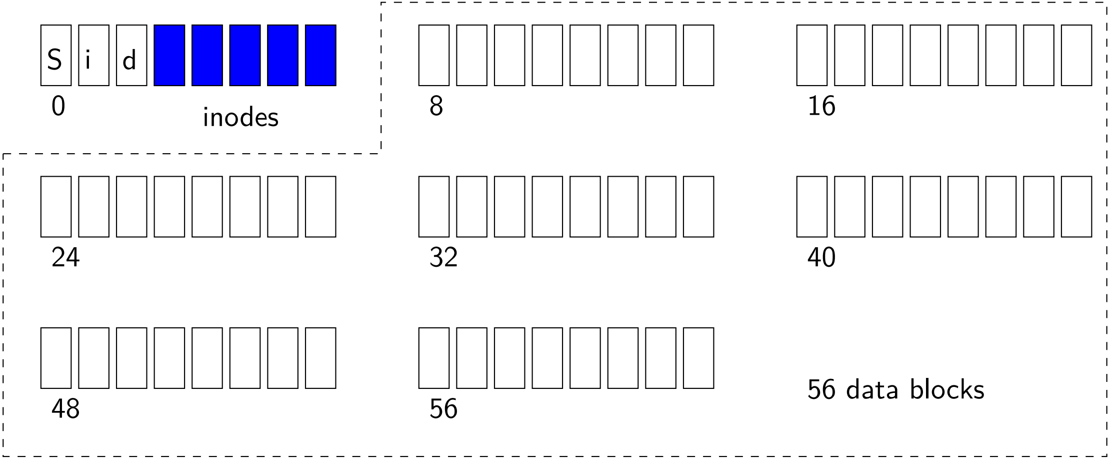
crash recovery

Remember the Very Simple File System:



crash recovery

Remember the Very Simple File System:

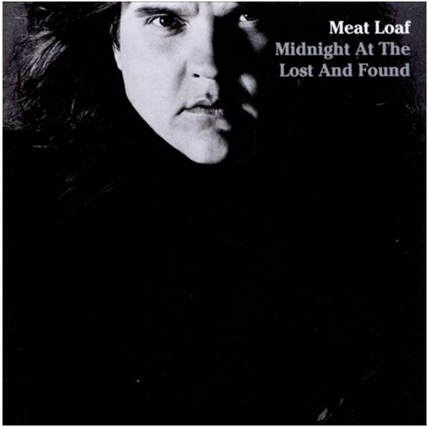


fsck /dev/sdb1

```
$ sudo fsck  -f /dev/sdb1
fsck from util-linux 2.27.1
e2fsck 1.42.13 (17-May-2015)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb1: 3339/125952 files (0.1% non-contiguous), 318256/503808 blocks
```

the lost and found

the lost and found



lost+found

```
> ls -il /
    :
    :

11010049 drwxr-xr-x   2 root root 12288 nov 28 17:49 libx32

      11 drwx------   2 root root 16384 maj  8  2016 lost+found

14155777 drwxr-xr-x   3 root root  4096 jun 29 14:13 media

  262145 drwxr-xr-x   3 root root  4096 okt 22 10:17 mnt

    :
    :
```

journal

We need to move from *a consistent state* to a *consistent state*.

journal

We need to move from *a consistent state* to a *consistent state*.

Let's keep a *journal* of things we are about to do.

journal

We need to move from *a consistent state* to a *consistent state*.

Let's keep a *journal* of things we are about to do.

Journal or Write-Ahead Logging

journal

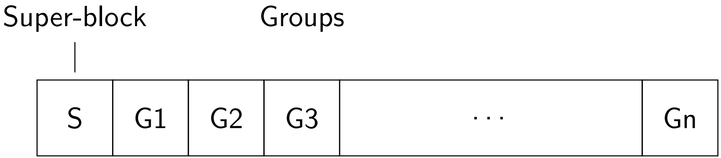
We need to move from *a consistent state* to a *consistent state*.

Let's keep a *journal* of things we are about to do.

Journal or Write-Ahead Logging

If we crash we can look at the journal to repeat the last sequence of operations.

Linux ext2 - no journaling

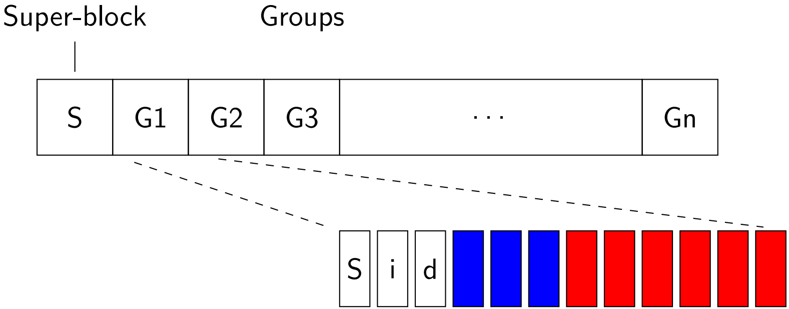


Super-block

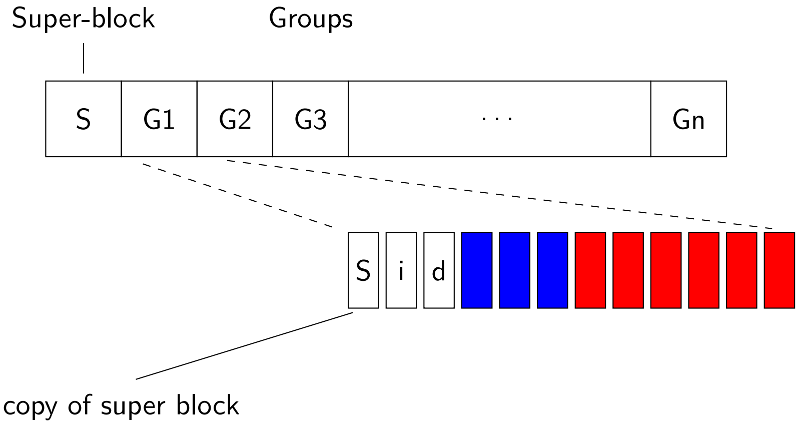
Groups

S	G1	G2	G3	· · ·	Gn

Linux ext2 - no journaling



Linux ext2 - no journaling

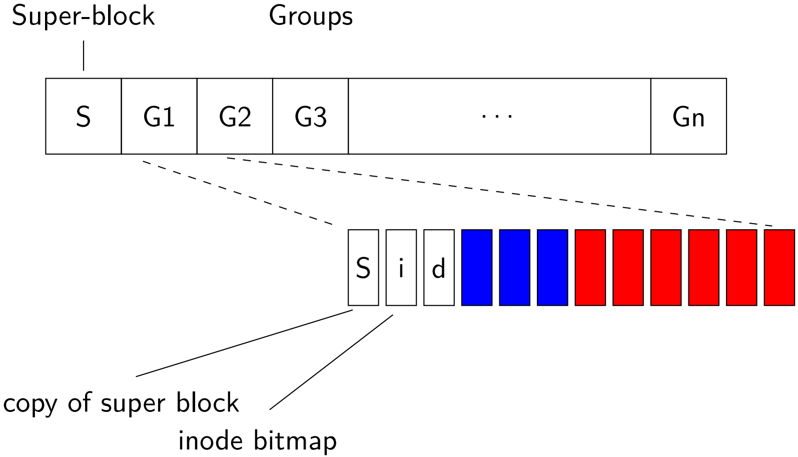


Super-block

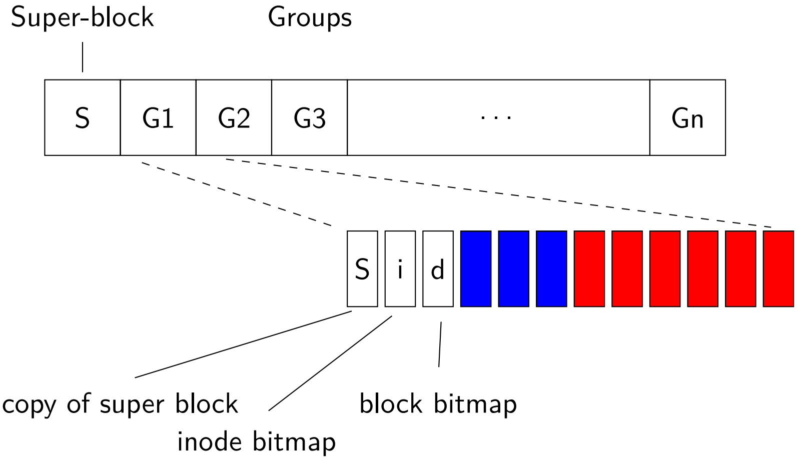
Groups

copy of super block

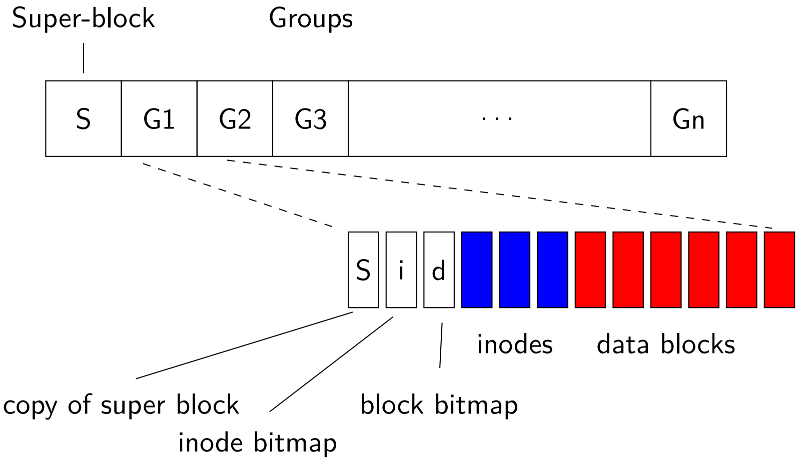
Linux ext2 - no journaling



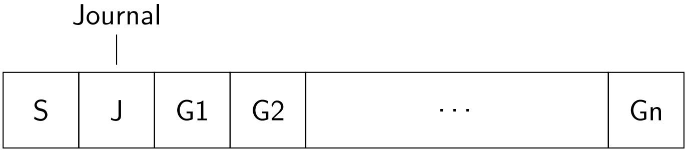
Linux ext2 - no journaling



Linux ext2 - no journaling



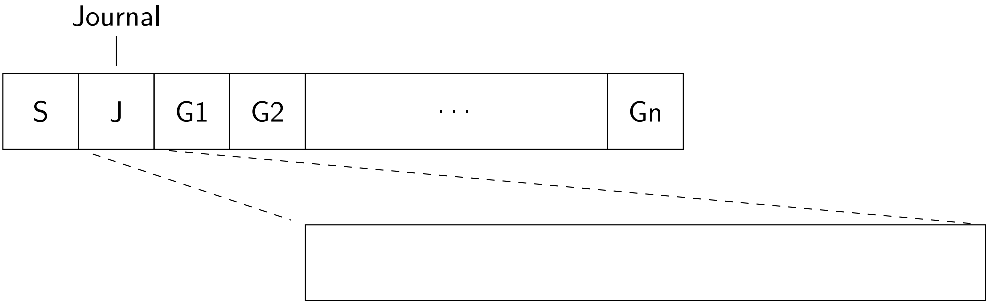
Linux ext3 - journaling



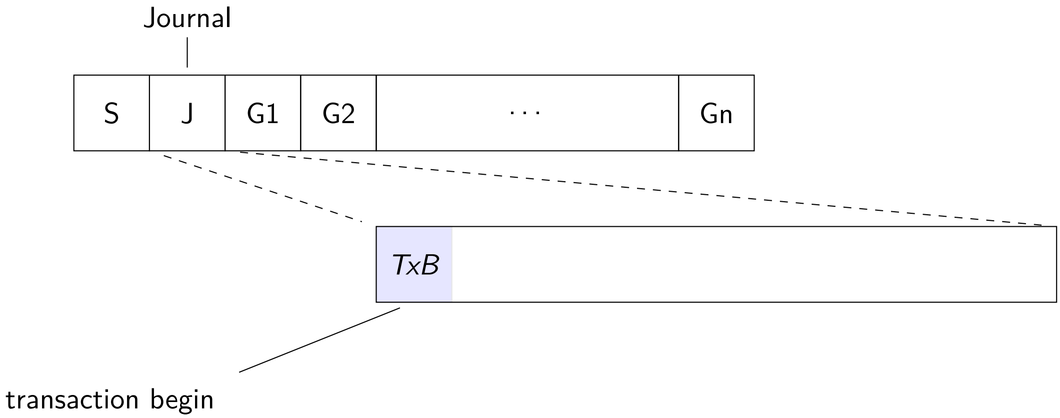
Journal

S J G1 G2 ... Gn

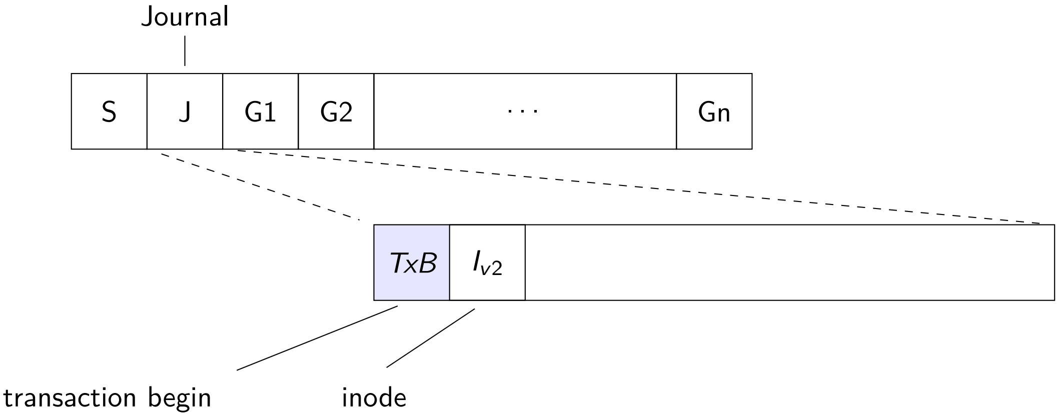
Linux ext3 - journaling



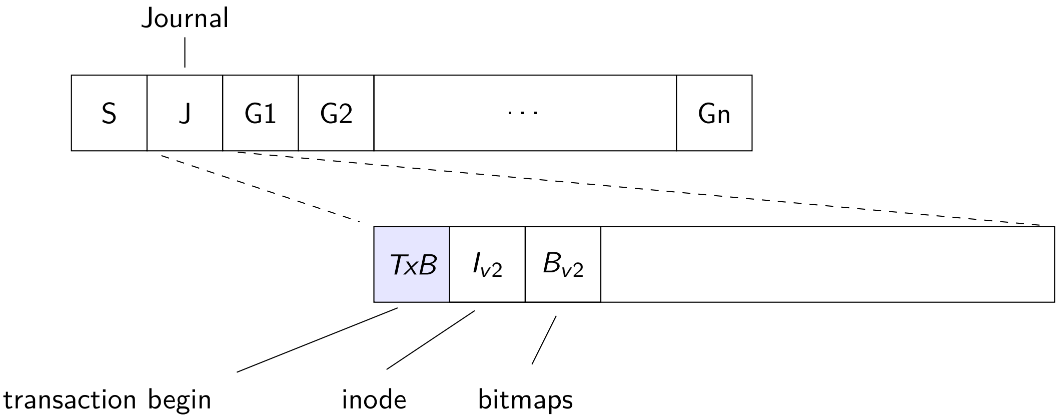
Linux ext3 - journaling



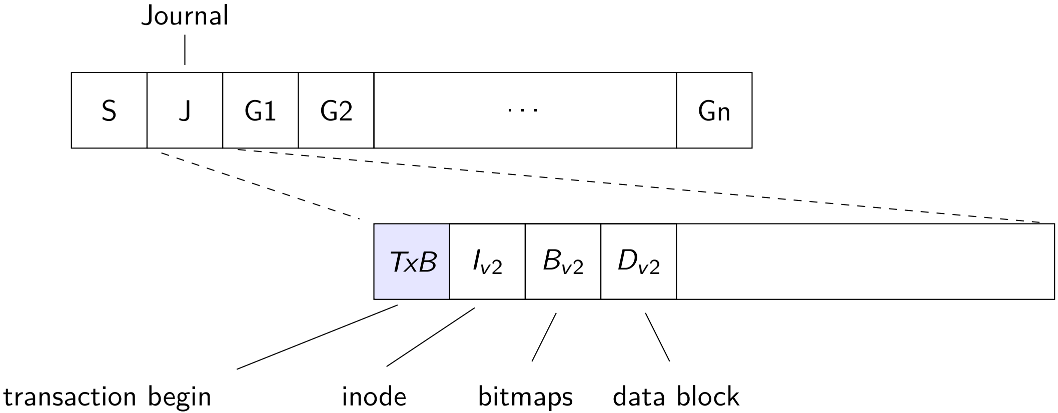
Linux ext3 - journaling



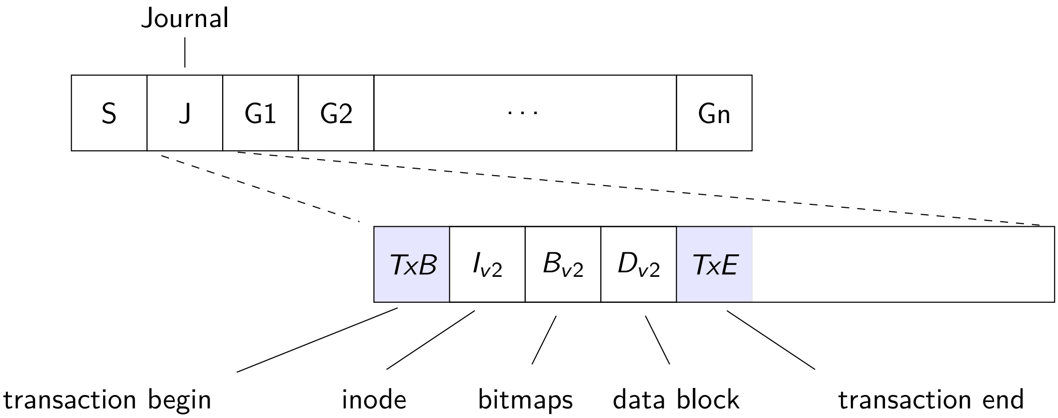
Linux ext3 - journaling



Linux ext3 - journaling



Linux ext3 - journaling



the safe way

- Commit: write the transaction

the safe way

- Commit: write the transaction
    - *TxB* : transaction id, inode id, bit map id, data block id

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id
- Checkpoint: perform the changes

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id
- Checkpoint: perform the changes
  - update the blocks: inode, bit maps and data block

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id
- Checkpoint: perform the changes
  - update the blocks: inode, bit maps and data block
  - remove transaction

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id
- Checkpoint: perform the changes
  - update the blocks: inode, bit maps and data block
  - remove transaction

the safe way

- Commit: write the transaction
  - TxB : transaction id, inode id, bit map id, data block id
  - I_{v2} : the updated inode
  - B_{v2} : the updated bitmaps
  - D_{v2} : the updated data block
  - TxE : transaction id
- Checkpoint: perform the changes
  - update the blocks: inode, bit maps and data block
  - remove transaction

disaster scenarios

disaster scenarios

We manage to write half of the transaction.

disaster scenarios

We manage to write half of the transaction.

We manage to write the whole transaction but not updating the blocks.

disaster scenarios

We manage to write half of the transaction.

We manage to write the whole transaction but not updating the blocks.

We manage to write the whole transaction, updating the blocks but not remove the transaction.

disaster scenarios

We manage to write half of the transaction.

We manage to write the whole transaction but not updating the blocks.

We manage to write the whole transaction, updating the blocks but not remove the transaction.

We manage to write TxB, I_{v2} and TxE and then crash.

disaster scenarios

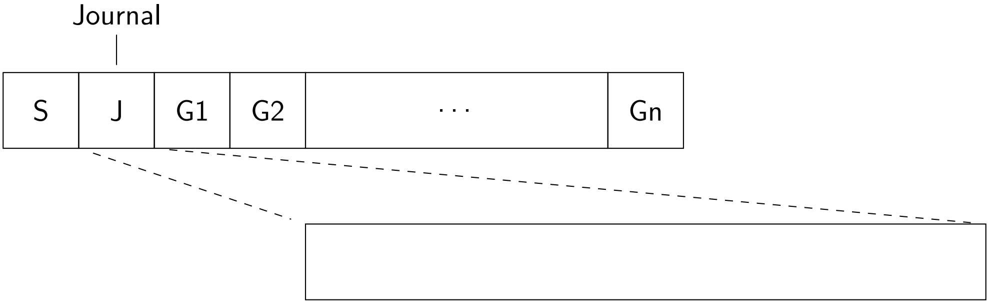
We manage to write half of the transaction.

We manage to write the whole transaction but not updating the blocks.

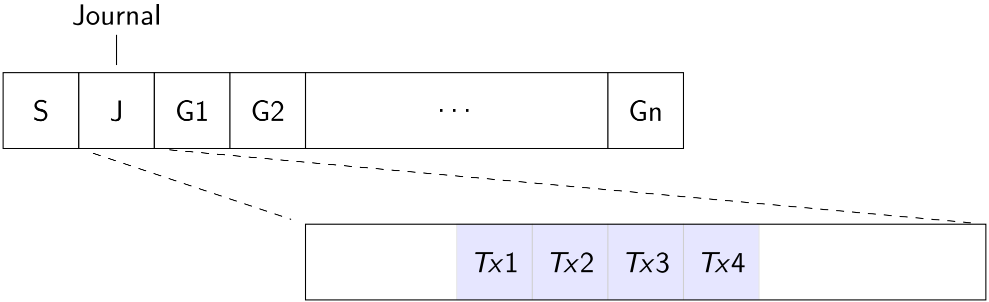
We manage to write the whole transaction, updating the blocks but not remove the transaction.

We manage to write TxB, I_{v2} and TxE and then crash.

pending transactions

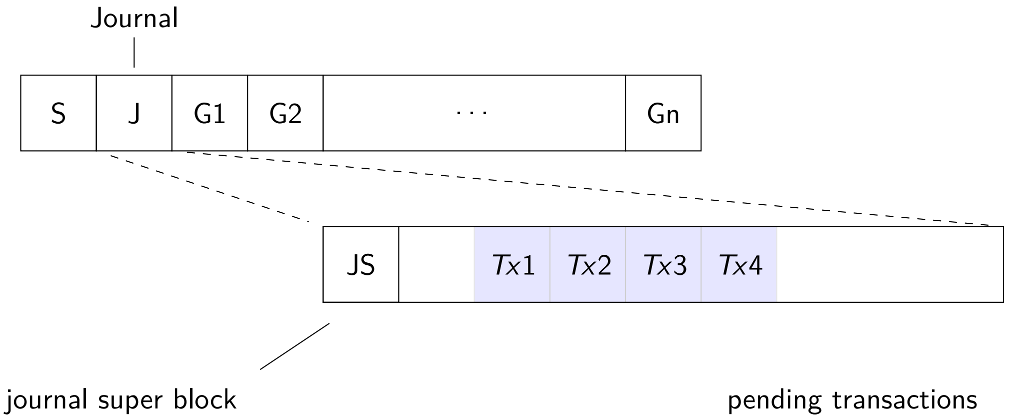


pending transactions

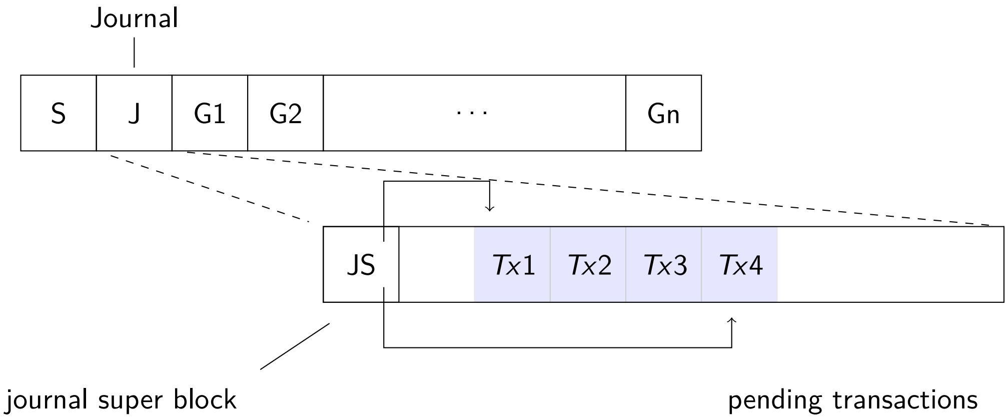


pending transactions

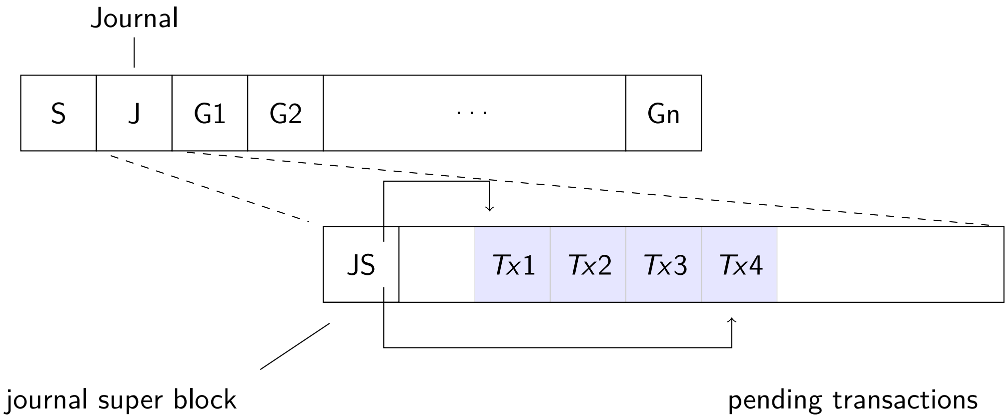
pending transactions



pending transactions

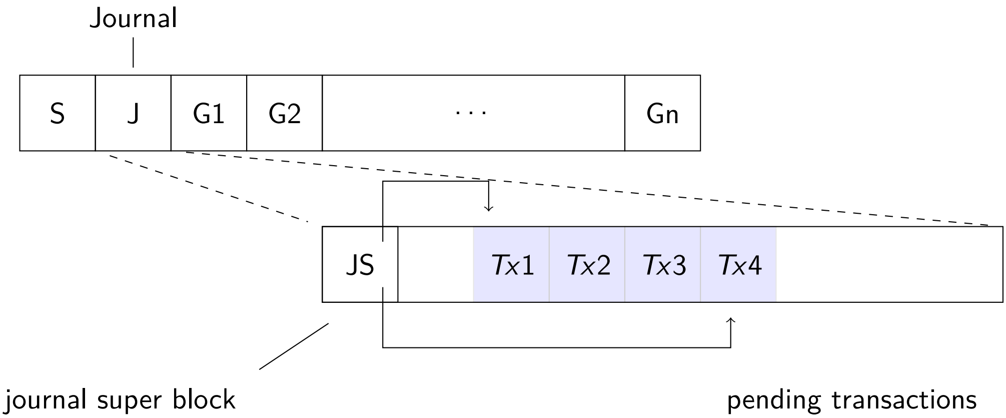


pending transactions



What is the state of the file system?

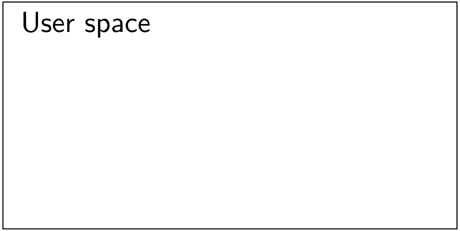
pending transactions



What is the state of the file system?

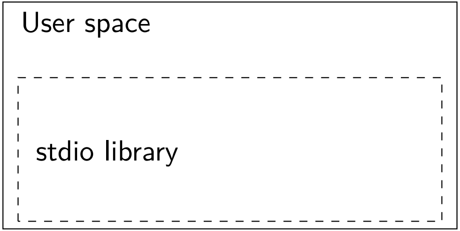
Can we read from the file system?

Layers of caches

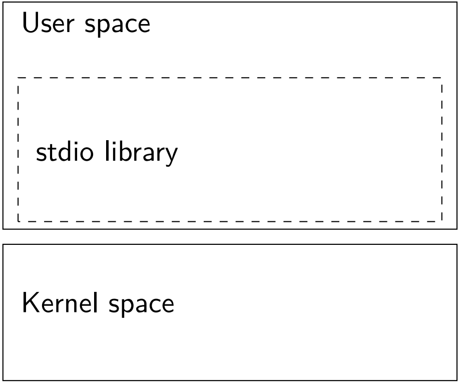


User space

Layers of caches



Layers of caches

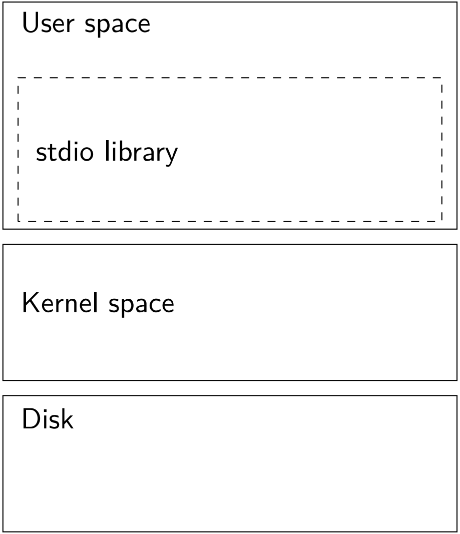


User space

stdio library

Kernel space

Layers of caches



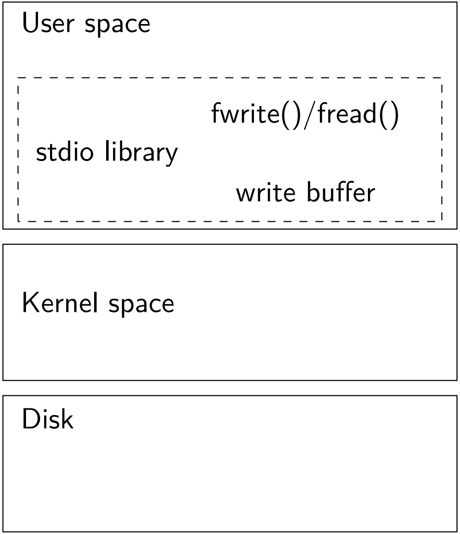
User space

stdio library

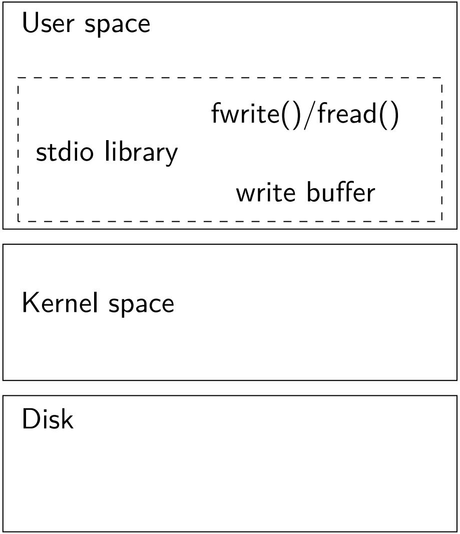
Kernel space

Disk

Layers of caches



Layers of caches



User space

fwrite()/fread()

stdio library

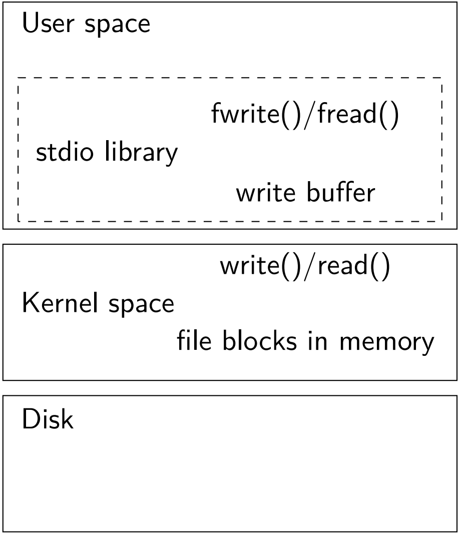
write buffer

Kernel space

Disk

- flush(): changes in buffer to kernel

Layers of caches



User space

stdio library

fwrite()/fread()

write buffer

write()/read()

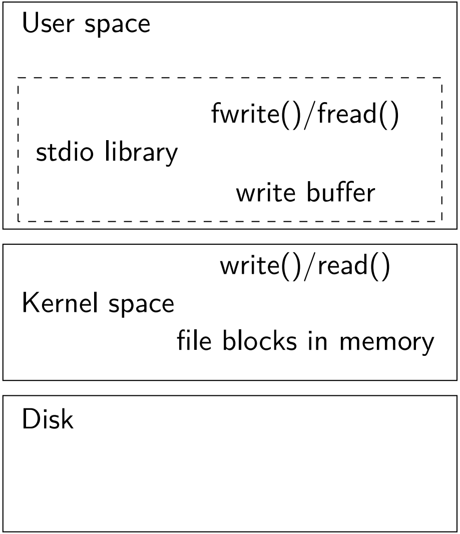
Kernel space

file blocks in memory

Disk

- flush(): changes in buffer to kernel

Layers of caches



User space

stdio library

fwrite()/fread()

write buffer

Kernel space

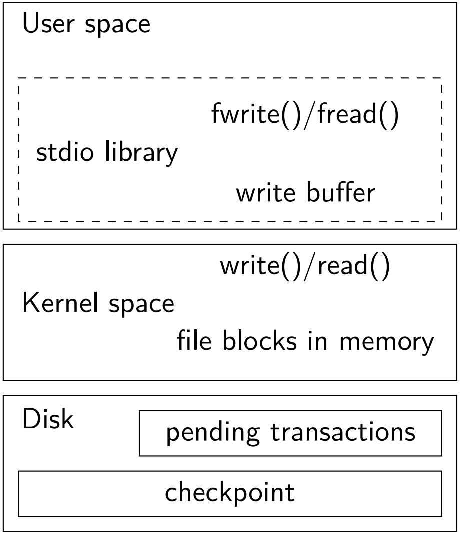
write()/read()

file blocks in memory

Disk

- flush(): changes in buffer to kernel
- sync(): changes to file system journal/checkpoint

Layers of caches



User space

stdio library

fwrite()/fread()

write buffer

write()/read()

Kernel space

file blocks in memory

Disk

pending transactions

checkpoint

- flush(): changes in buffer to kernel
- sync(): changes to file system journal/checkpoint
- checkpointing: from journal to inodes, maps and blocks

do the right thing....?

Journal is slow:

- Commit: write meta-data and data in a transaction (make sure it's a complete transaction).

do the right thing....?

Journal is slow:

- Commit: write meta-data and data in a transaction (make sure it's a complete transaction).
- Checkpointing: update the inode, bitmap and data blocks given the transaction.

do the right thing....?

Journal is slow:

- Commit: write meta-data and data in a transaction (make sure it's a complete transaction).
- Checkpointing: update the inode, bitmap and data blocks given the transaction.

Everything is written twice to disk!

do the right thing....?

Journal is slow:

- Commit: write meta-data and data in a transaction (make sure it's a complete transaction).
- Checkpointing: update the inode, bitmap and data blocks given the transaction.

Everything is written twice to disk!

Idea - do the wrong thing and pray for the best.

data only once

Faster:

- Commit data : write data directly to block.

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction.

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction.

Even faster:

- Commit data : write data directly to block... eventually, hopefully.

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction.

Even faster:

- Commit data : write data directly to block... eventually, hopefully.
- Commit meta-data: write meta-data in transaction.

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction.

Even faster:

- Commit data : write data directly to block... eventually, hopefully.
- Commit meta-data: write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction (let's hope the data is there)

data only once

Faster:

- Commit data : write data directly to block.
- Commit meta-data: when data is in block, write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction.

Even faster:

- Commit data : write data directly to block... eventually, hopefully.
- Commit meta-data: write meta-data in transaction.
- Checkpointing: update the inode and bitmap given transaction (let's hope the data is there)

ext4/jdb2

- `journal`: all data and meta-data is written through journal

ext4/jdb2

- `journal`: all data and meta-data is written through journal
- `ordered` (default): data is written immediately to block, meta-data through journal

ext4/jdb2

- `journal`: all data and meta-data is written through journal
- `ordered` (default): data is written immediately to block, meta-data through journal
- `write-back` : data is not guaranteed to be written before meta-data

inode - is everything important

```
> sudo istat /dev/sda1 2236582
inode: 2236582                                Group: 273
Generation Id: 3805640679
uid/gid: 1000/1000      mode: rrw-rw-r--   Flags: Extents,

size: 43  num of links: 1

Inode Times:
Accessed: 2016-12-06 14:51:17.003254544 (CET)
File Modified: 2016-12-06 15:46:55.667041193 (CET)
Inode Modified: 2016-12-06 15:46:55.667041193 (CET)
File Created: 2016-12-06 13:39:15.084806928 (CET)

Direct Blocks:  6946002
```

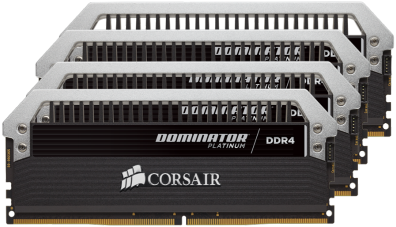
What about this?

What about this?

This album has nothing to do with the following material.

Disk vs Memory

Disk vs Memory



Log-structured file systems

Log-structured file systems

Reading is mostly done from cached copies in memory.

Log-structured file systems

Reading is mostly done from cached copies in memory.

Focus on write operations, try to avoid moving the arm.

Log-structured file systems

Reading is mostly done from cached copies in memory.

Focus on write operations, try to avoid moving the arm.

Writing is best done in large consecutive segments.

Log-structured file systems

Reading is mostly done from cached copies in memory.

Focus on write operations, try to avoid moving the arm.

Writing is best done in large consecutive segments.

The state of the file system is *a log of events*.

the log

D

the log

D	D

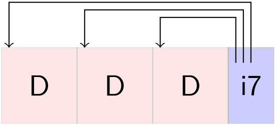
the log

D	D	D

the log

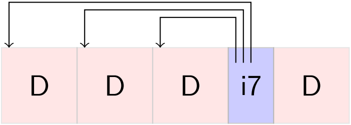
D	D	D	i7

the log

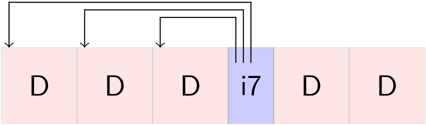


D	D	D	i7

the log

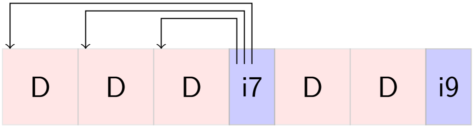


the log



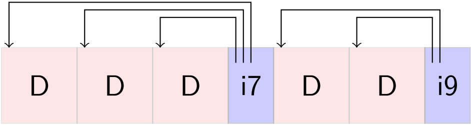
D D D i7 D D

the log

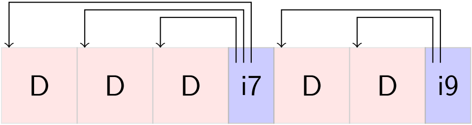


D D D i7 D D i9

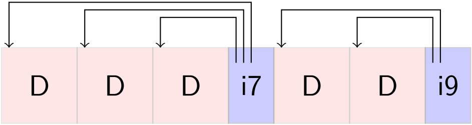
the log



the log

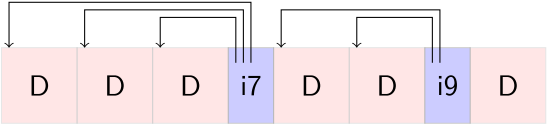


the log

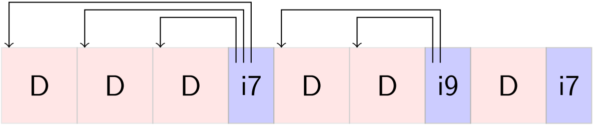


D D D i7 D D i9

the log

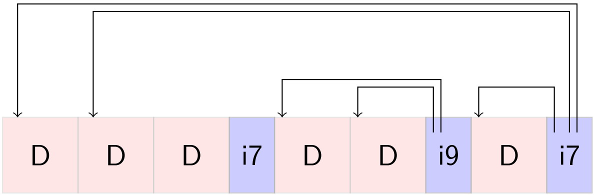


the log

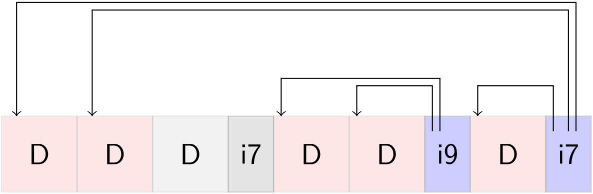


D D D i7 D D i9 D i7

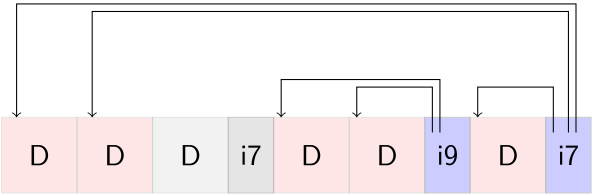
the log



the log



the log

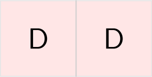


How do we find the inodes?

the inode map

D

the inode map



D D

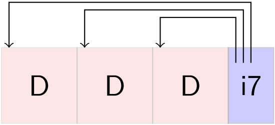
the inode map

D	D	D

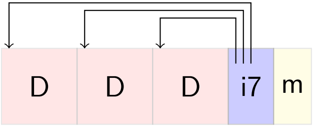
the inode map

D	D	D	i7

the inode map

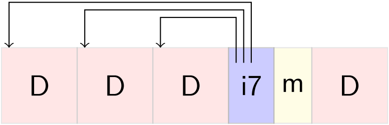


the inode map



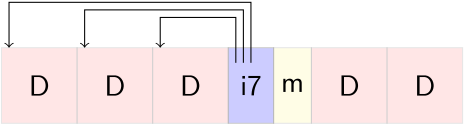
The inode map holds mapping from inode number to block addresses.

the inode map



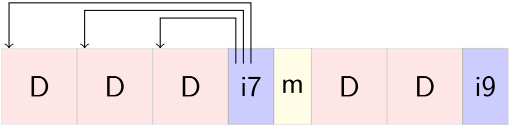
The inode map holds mapping from inode number to block addresses.

the inode map



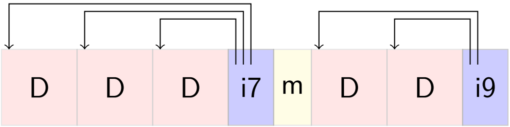
The inode map holds mapping from inode number to block addresses.

the inode map



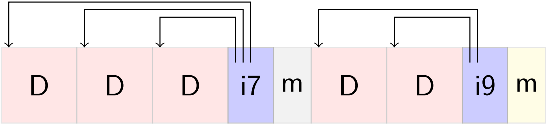
The inode map holds mapping from inode number to block addresses.

the inode map



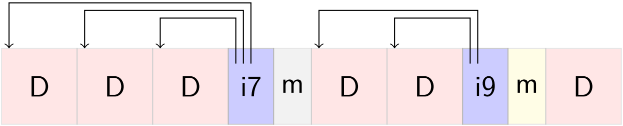
The inode map holds mapping from inode number to block addresses.

the inode map



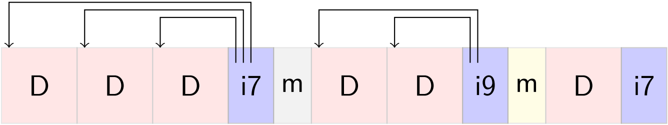
The inode map holds mapping from inode number to block addresses.

the inode map



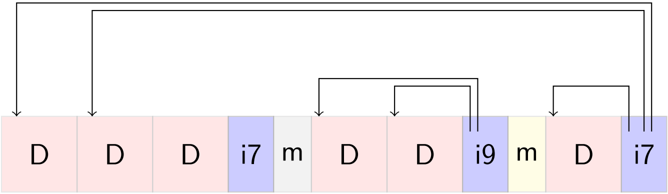
The inode map holds mapping from inode number to block addresses.

the inode map



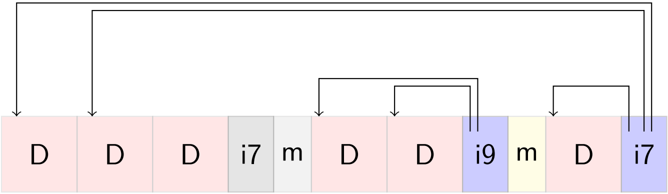
The inode map holds mapping from inode number to block addresses.

the inode map



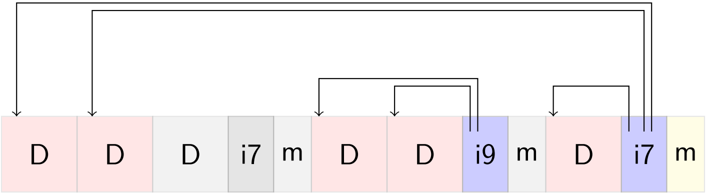
The inode map holds mapping from inode number to block addresses.

the inode map



The inode map holds mapping from inode number to block addresses.

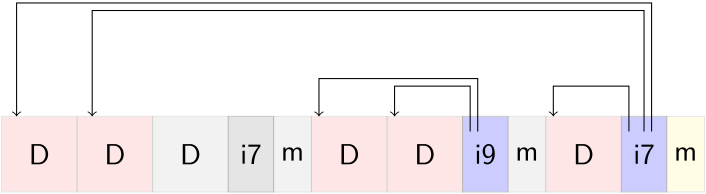
the inode map



The inode map holds mapping from inode number to block addresses.

How do we find the last inode map?

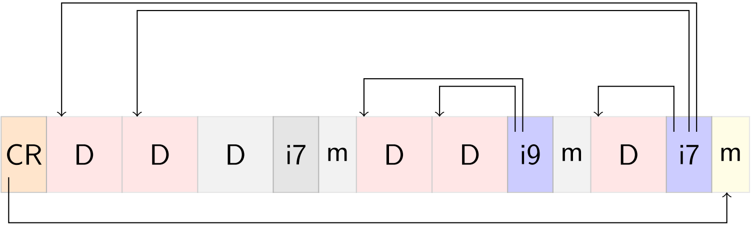
the inode map



The inode map holds mapping from inode number to block addresses.

How do we find the last inode map?

the inode map



The inode map holds mapping from inode number to block addresses.

How do we find the last inode map?

pros and cons

reading a file

- read the check region
- find the location of the inode map
- find inode
- read data block

pros and cons

reading a file

- read the check region
- find the location of the inode map
- find inode
- read data block

writing a file

- write data block
- write new copy of inode
- write new copy of inode map
- update check region

pros and cons

reading a file

- read the check region
- find the location of the inode map
- find inode
- read data block

writing a file

- write data block
- write new copy of inode
- write new copy of inode map
- update check region

How much can we cache in memory?

pros and cons

reading a file

- read the check region
- find the location of the inode map
- find inode
- read data block

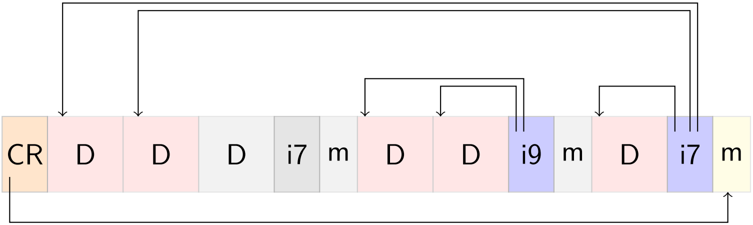
writing a file

- write data block
- write new copy of inode
- write new copy of inode map
- update check region

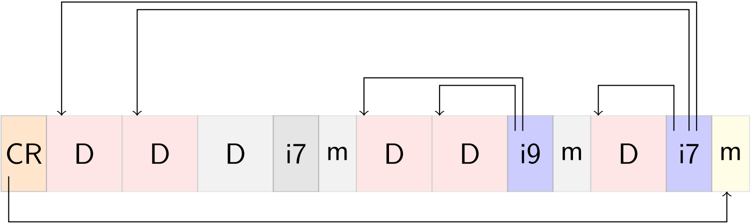
How much can we cache in memory?

Can we delay updating the check region?

reclaim sectors



reclaim sectors



Where is the bit map that keeps track of available blocks?

Hmmm.....

Do we want to know where to find blocks ..

Hmmm.....

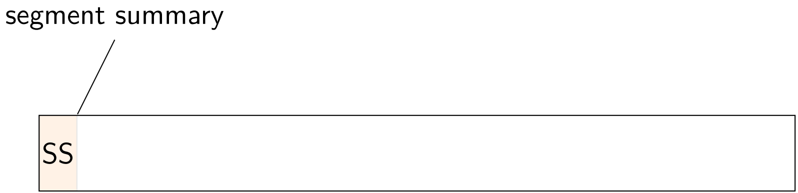
Do we want to know where to find blocks ..

if they are scattered around the disk?

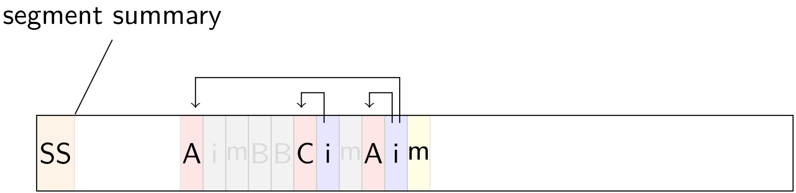
garbage collection and compaction

garbage collection and compaction

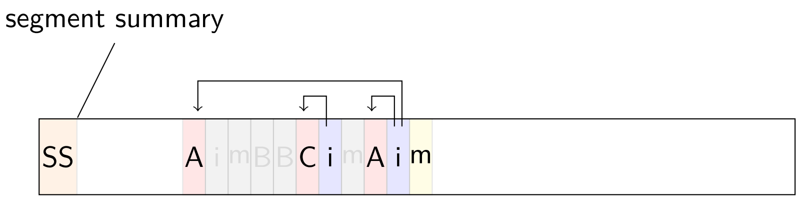
garbage collection and compaction



garbage collection and compaction

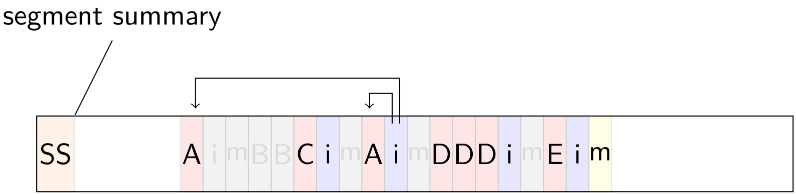


garbage collection and compaction



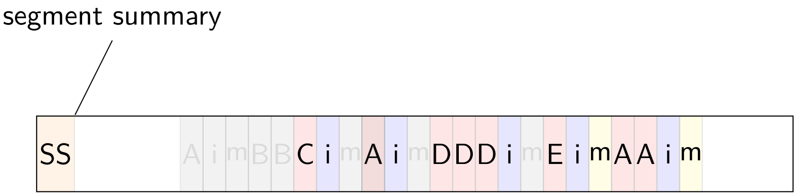
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



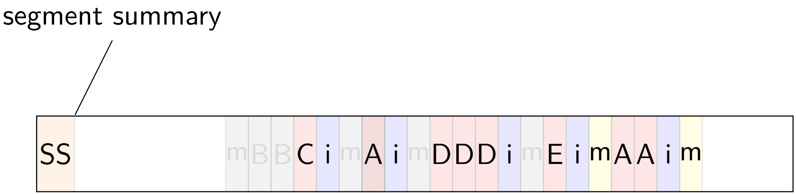
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



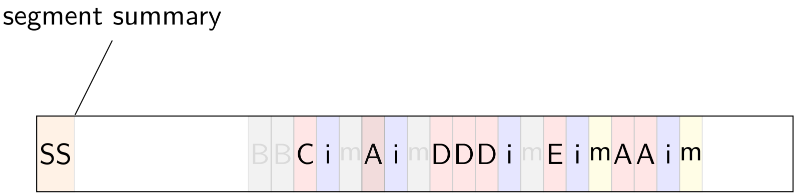
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



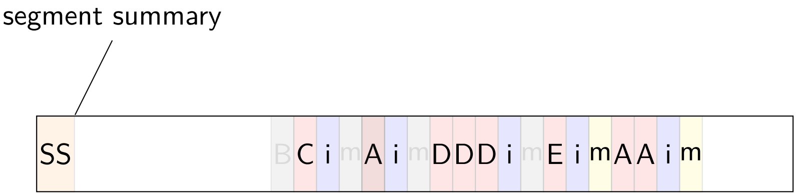
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



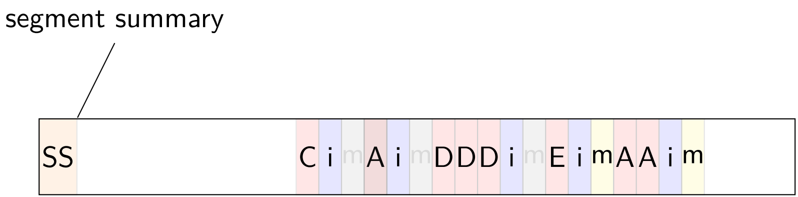
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



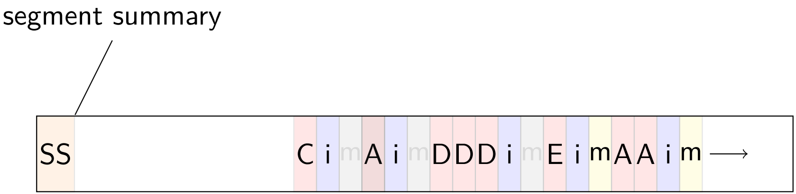
Segment summary keeps a mapping from block to inode.

garbage collection and compaction



Segment summary keeps a mapping from block to inode.

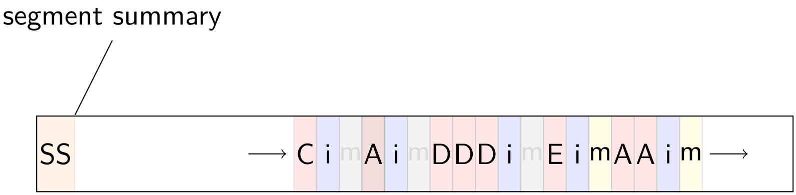
garbage collection and compaction


segment summary

SS C i m A i m D D D i m E i m A A i m →

Segment summary keeps a mapping from block to inode.

garbage collection and compaction



Segment summary keeps a mapping from block to inode.

What about SSDs?

What about SSDs?

old enough to remember this

The file system UDF used a log structure to do updates on a write-once CD/DVD

Some file systems

- `ext4` : default Linux system, journaling

Some file systems

- `ext4` : default Linux system, journaling
- `F2FS` : by Samsung, log-structured, optimised for SSD

Some file systems

- `ext4` : default Linux system, journaling
- `F2FS` : by Samsung, log-structured, optimised for SSD
- `NILFS` : Nipon Telecom, log-structured

Some file systems

- `ext4` : default Linux system, journaling
- `F2FS` : by Samsung, log-structured, optimised for SSD
- `NILFS` : Nipon Telecom, log-structured
- `btrfs` : originally by Oracle, a copy-on-write system

Some file systems

- ext4 : default Linux system, journaling
- F2FS : by Samsung, log-structured, optimised for SSD
- NILFS : Nipon Telecom, log-structured
- btrfs : originally by Oracle, a copy-on-write system
- APFS : next generation for OSX (Sierra 2017), copy-on-write

Some file systems

- ext4 : default Linux system, journaling
- F2FS : by Samsung, log-structured, optimised for SSD
- NILFS : Nipon Telecom, log-structured
- btrfs : originally by Oracle, a copy-on-write system
- APFS : next generation for OSX (Sierra 2017), copy-on-write
- ReFS : latest file system for Windows servers, copy-on-write

Some file systems

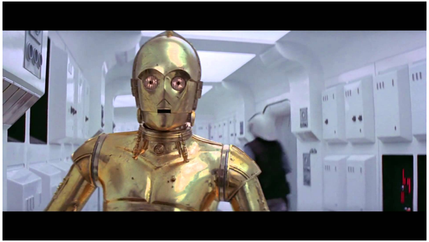
- ext4 : default Linux system, journaling
- F2FS : by Samsung, log-structured, optimised for SSD
- NILFS : Nipon Telecom, log-structured
- btrfs : originally by Oracle, a copy-on-write system
- APFS : next generation for OSX (Sierra 2017), copy-on-write
- ReFS : latest file system for Windows servers, copy-on-write
- exFAT : Microsoft system used by SD cards

Some file systems

- ext4 : default Linux system, journaling
- F2FS : by Samsung, log-structured, optimised for SSD
- NILFS : Nipon Telecom, log-structured
- btrfs : originally by Oracle, a copy-on-write system
- APFS : next generation for OSX (Sierra 2017), copy-on-write
- ReFS : latest file system for Windows servers, copy-on-write
- exFAT : Microsoft system used by SD cards

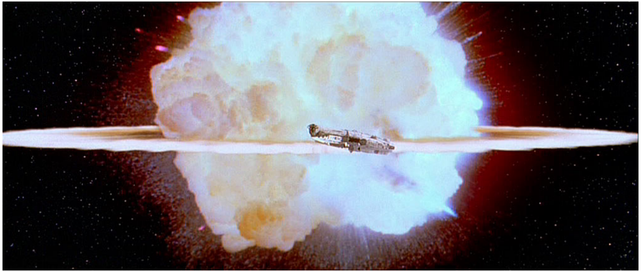
Summary

Summary



We're doomed!

Summary



Saved!