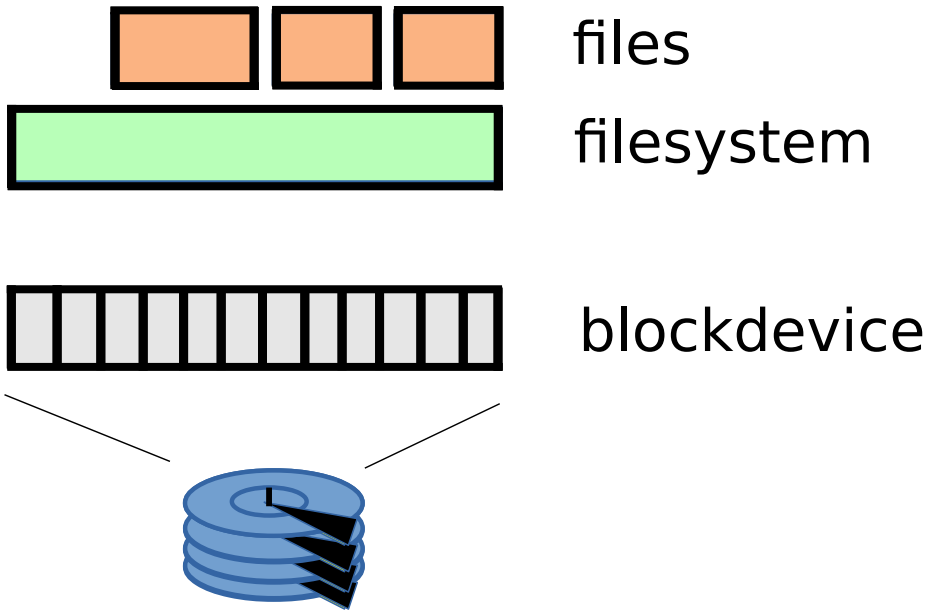
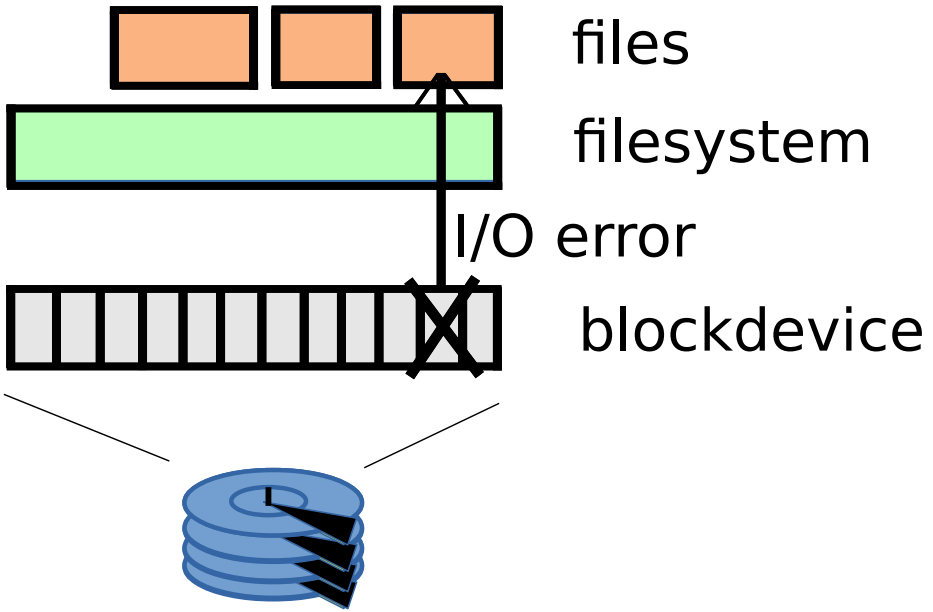
Bit rot:
Would you notice if your harddisk was lying to you?

Christian Horn
(plus lvm-list@ and many specialists from Red Hat)
chorn@fluxcoil.net / mastodon: globalc@chaos.social / https://fluxcoil.net

The most basic setup:

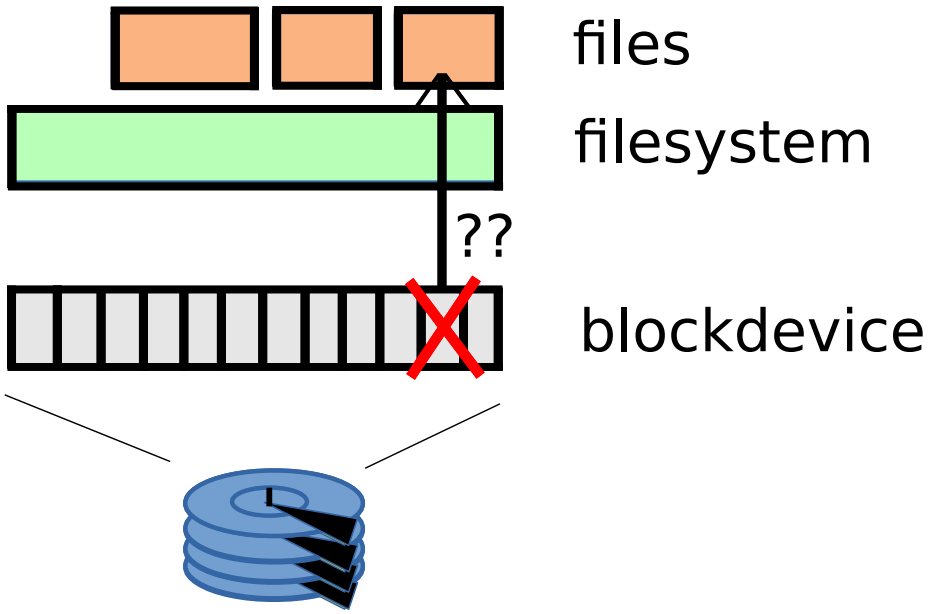


What happens with read errors?



=> We are safe: application notices.

What happens if data silently changes?



Let’s find out!

```
# dd if=/dev/zero of=rawfile bs=1M count=128
# MYLOOP=$(losetup -f --show rawfile)
# echo $MYLOOP
/dev/loop27
# mkfs.xfs $MYLOOP
# mount $MYLOOP /mnt
```

- Generating a 128MB file with zeros
- Using losetup to make the file available as blockdevice
- Creating an XFS file system
- Mounting the file system

```
# yes >/mnt/infile
yes: standard output: No space left on device
# md5sum /mnt/infile
66e48b263b313703fce56a8a5a848eef  /mnt/infile
# umount /mnt
# losetup -d $MYLOOP
```

- Store a single file on the file system, filled with the ASCII character ‘y’ and newline characters
- Then unmounting the file system
- Looking at the content at 50MB offset

```
# dd if=rawfile bs=1 count=10 skip=$((50*1024*1024)) 2>/dev/null|
hexdump -vC
00000000 79 0a 79 0a 79 0a 79 0a  79 0a                    |
y.y.y.y.y.|
0000000a
```

We flip one bit:

At offset 50MB, we have the character code for ‘y’, written in hexadecimal 0x79, or in binary 01111001.
We flip the last bit into a 0, which makes this into hex 0x78, the ASCII code for ‘x’.

```
# echo 'x' | dd of=rawfile bs=1 count=1 seek=$((50*1024*1024)) conv=notrunc
# dd if=rawfile bs=1 count=10 skip=$((50*1024*1024)) 2>/dev/null|hexdump -vC
00000000 78 0a 79 0a 79 0a 79 0a  79 0a                    |x.y.y.y.y.|
0000000a

Now we can mount the file again as filesystem, and check if the changed data gets
noticed.
# MYLOOP=$(losetup -f --show rawfile)
# mount $MYLOOP /mnt/

# md5sum /mnt/tmp/infile
adf92be755d095e898281b1146be72ce  /mnt/infile
```

=> The checksum changed, our disk is lying to us without any indication something is wrong!

So how can we detect bit rot?

With dm-integrity. Setup:

```
# dd if=/dev/zero of=rawfile bs=1M
count=128
# MYLOOP=$(losetup -f --show rawfile)
# integritysetup format $MYLOOP
# integritysetup open $MYLOOP mydata
# mkfs.xfs /dev/mapper/mydata
# mount /dev/mapper/mydata /mnt
# yes >/mnt/infile
# md5sum /mnt/infile
13e14c50aaf2054d987663ed31b5f786
/mnt/infile
```

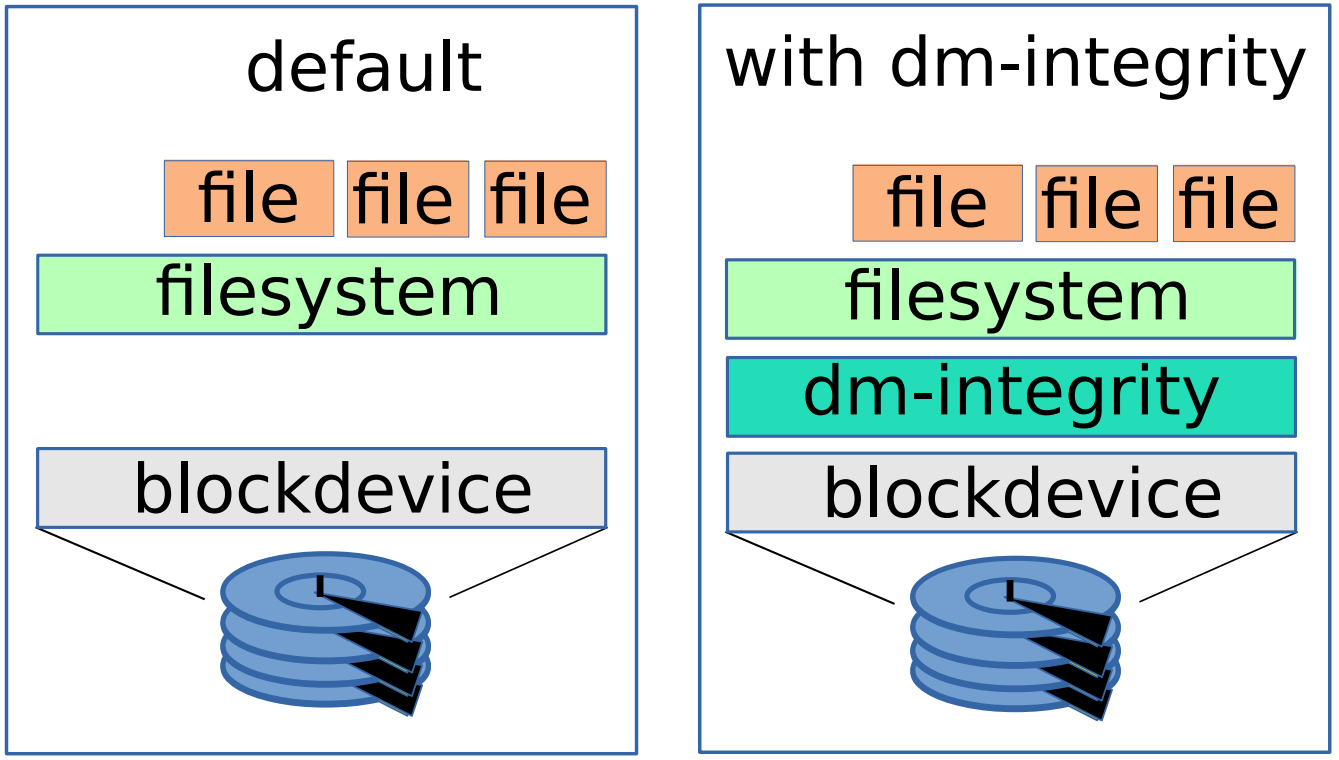
..and test.

```
# umount /mnt/
# losetup -d $MYLOOP
# integritysetup close mydata
# dd if=rawfile bs=1 count=10 skip=$((50*1024*1024)) 2>/dev/null|hexdump
00000000  79 0a 79 0a 79 0a 79 0a  79 0a                    |y.y.y.y.y.|
0000000a
# echo 'x' | dd of=rawfile bs=1 count=1 seek=$((50*1024*1024)) conv=notru
# dd if=rawfile bs=1 count=10 skip=$((50*1024*1024)) 2>/dev/null|hexdump
00000000  78 0a 79 0a 79 0a 79 0a  79 0a                    |x.y.y.y.y.|
0000000a
```

Now we detect the issue:

```
# MYLOOP=$(losetup -f --show rawfile)
# integritysetup open $MYLOOP mydata
# mount /dev/mapper/mydata /mnt
# md5sum /mnt/infile
md5sum: /mnt/infile: Input/output error
```

Minimal layers (no partitions, LVM etc.)



Details

TAM blog article, public and sharable with customers:
What is bit rot, and how can I detect it on RHEL?

chorn@fluxcoil.net / mastodon: globalc@chaos.social / https://fluxcoil.net