Homework
Part 3: Persistence –
11) RAID, Files and Directories

master@323240b (20230907-115823)

P. Mainini / E. Benoist / C. Fuhrer / L. Ith

BTI1341 / Fall 2023/24

1 Understanding RAID

Capacity

1. Using RAID 0 and two disks of 1 TiB each, how much data can be stored?
2. Using RAID 0 and four disks of 2 TiB each, how much data can be stored?
3. Using RAID 1 and two disks of 1 TiB each, how much data can be stored?
4. Using RAID 5 and five disks of 2 TiB each, how much data can be stored?

Parity Bits

1. Given the following bits, compute the XOR parity bit!
   a) 1, 0, 1, 1
   b) 0, 0, 1, 1
   c) 1, 0, 0, 0
2. Given the following bytes, compute the XOR parity *byte*!
   a) `0x9A`, `0xAA`, `0x12`, `0xFF`
   b) `0xFF`, `0x00`, `0xAB`, `0xC7`
3. Assuming the following bytes:

   `0x89`, `0x56`, `0x34`, `0xA5`, parity: `0x4E`

   Compute the *subtractive parity* if the value of byte `0x89` changes to `0xFF`!

2 GNU/Linux Software RAID

In GNU/Linux, software RAIDs can easily be created and managed using `mdadm`. Together with *loop devices,*[1] you can even create RAID devices which are based on files instead of HDDs!

Try to recreate the different RAID levels given in the lecture using these tools. Also try to simulate failures and spare disks etc.

For example, a simple RAID-0 device can be created as follows:

```
# Create files to be used as "HDDs" (size: 200 MiB)
$ dd if=/dev/zero of=disk-0 bs=1M count=200
$ dd if=/dev/zero of=disk-1 bs=1M count=200

# Setup loop devices
$ losetup /dev/loop0 disk-0
$ losetup /dev/loop1 disk-1

# Setup RAID-0
$ mdadm --create --verbose --level=0 /dev/md0 \
        --raid-devices=2 /dev/loop0 /dev/loop1

# Create filesystem
$ mkfs.ext4 /dev/md0

# Mount it!
$ mount /dev/md0 /mnt/
```

⚠ Attention: Verify first if there are any loop devices already in use on your computer, using "`losetup -a`". Create the new loop devices starting with the first number available (eg. `/dev/loop7` if `/dev/loop0` to `/dev/loop6` are in use).

[1] A loop device is a pseudo-device which makes the contents of a file accessible as a block device.

3 Files and Directories

Write your own implementation of the program `tail`, which supports the following options:

`-c` If invoked with "`-c NUM`", only the last `NUM` bytes must be printed. If "`-c +NUM`" has been given, the output *starts* at the given position/byte.

`-f` If this option is given, the program "follows" the file, i.e. any new data appended to the file must be printed immediately (`tail` keeps running until `Ctrl-C` has been pressed).

`-n` Invocation with "`-n NUM`" or "`-n +NUM`" works similar to "`-c`", but prints the last (or first) `NUM` lines (instead of bytes). If not specified, the *default is 10*!

Searching through the file should be implemented *efficiently*, making use of the syscalls introduced in the lecture. You do not need to support invocation on `stdin`.

See also "`man tail`" for more details.