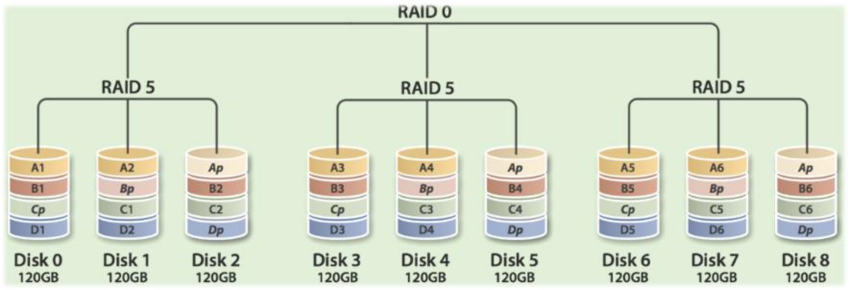
Operating Systems

Homework Assignment #2

Due 4.12.2014, 23:59

Part 1



Read about **RAID-50**, then change the code seen in class to implement a RAID-50 simulator.

- Shut down your VM, add disks as needed, and restart the VM.
  - Verify the devices were added: "`ls -l /dev/sd*`" (should be `/dev/sdb, /dev/sdc`, etc.)
- Write a program **raid50.o**. The first parameter $\boldsymbol{r_5}$ is the number of disks in each **RAID-5** array. The second parameter $\boldsymbol{r_0}$ is the number of **RAID-5** arrays in the main **RAID-0** array. The rest of the parameters are device names (/dev/sdb, /dev/sdc, etc.). You may assume that $\boldsymbol{r_0 \geq 2, r_5 > 2}$ and that there are exactly $\boldsymbol{r_0 \cdot r_5}$ device names.
  - Initialize all cells of the read/write buffer to **0**.
  - Change `do_raid0_rw()` to perform reads and writes according to the structure of RAID-50.
    - Rename it `do_raid50_rw()`
    - When operating on a failed disk, the operation should succeed if other disks are available.
    - Remember to update the relevant parity block on every **WRITE**.
  - Modify the **READ** operation
    - So that it prints the content of each sector.
    - You may assume a sector contains a single **byte** value, repeated. Thus, print a single **byte** value for each sector – the first byte read from the sector.
  - Add the following operations, along with the relevant implementations:
    - **REPAIR** - `do_raid50_repair()`
      - The first parameter ("sectors") determines the disk index to repair, the second parameter is ignored (but still provided for convenience in parsing).
      - The device should be reopened, and the entire data of the disk should be restored – as if it is a replacement disk for a previously-defected drive.
    - **SETBUF** - `do_raid50_setbuf()`
      - The first parameter ("sectors") determines the **byte** value to set each cell of the read/write buffer to. The second parameter is ignored.

Guidelines

- Use CTRL+D to send EOF to the console and exit gracefully.
- Use only system calls to access the devices. You can use standard C functions to read user input only (`stdin`).
- You can assume correctness of input – all devices exist, of same size and have enough capacity, input is valid, etc.
- If you fail writing/reading to a device (or closing it) – treat it as failed (**KILL**) and continue reading/writing. If more devices fail than can be recovered – exit with an error message.
- **Submit** a single C file: **raid50.c**