# Practice M6: FHS, Disks, Filesystems, and Archives (Additional)

\* NOTE: Most of the exercises included in this practice guide **are not suitable** for execution in **WSL** or **Docker** environments. It is recommended that a virtual infrastructure to be used instead.

For the purpose of the current practice, we will need one virtual machine powered on, but for different parts, we will use different machine based on different distribution.

All commands that we will use until the end of this practice will be accompanied by the appropriate prompt. This way it will be easier for us to know which user, in which folder, and on which machine is executing the command.

Every part will state clearly which distribution is used.

## Btrfs and openSUSE

Let's check where we start from:

lsauser@opensuse:~> **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 8M 0 part

└─sda2 8:2 0 8G 0 part /

sr0 11:0 1 1024M 0 rom

lsauser@opensuse:~>

Now, let's turn off the **VM** and add four additional hard disks:

* First with size of **1GB**
* Second with size of **2GB**
* Third with size of **500MB**
* Fourth with size of **500MB**

Then power on the **VM** again and check the situation:

lsauser@opensuse:~> **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 8M 0 part

└─sda2 8:2 0 8G 0 part /

sdb 8:16 0 1G 0 disk

sdc 8:32 0 2G 0 disk

sdd 8:48 0 512M 0 disk

sde 8:64 0 512M 0 disk

sr0 11:0 1 1024M 0 rom

lsauser@opensuse:~>

Our new disks are listed and reachable via **sdb**, **sdc**, **sdd**, and **sde** identifiers.

Now, we can create a filesystem that in a similar way to **LVM** spans over multiple drives with different sizes. Please note that if one of the drives fails, you will lose all information.

Let's use the **sdb** and **sdc** drives:

lsauser@opensuse:~> **sudo mkfs.btrfs -d single -f -L VOLUME-BTRFS /dev/sd[bc]**

btrfs-progs v4.15

See http://btrfs.wiki.kernel.org for more information.

Label: VOLUME-BTRFS

UUID: fd667810-f59d-4d0f-bf2d-b8228efb2315

Node size: 16384

Sector size: 4096

Filesystem size: 3.00GiB

Block group profiles:

Data: single 8.00MiB

Metadata: RAID1 153.56MiB

System: RAID1 8.00MiB

SSD detected: no

Incompat features: extref, skinny-metadata

Number of devices: 2

Devices:

ID SIZE PATH

1 1.00GiB /dev/sdb

2 2.00GiB /dev/sdc

lsauser@opensuse:~>

As we can see, metadata and system data are stored in **RAID1** mode. This means that those will be mirrored across all drives. We can force them (system + metadata) to have only one copy and follow the data distribution (mode) by adding a **-M** switch.

Similarly, we can create any of the supported **RAID** levels. For example, we can create a striped **RAID** or **RAID0** with:

lsauser@opensuse:~> **sudo mkfs.btrfs -d raid0 -f -L RAID0-BTRFS /dev/sd[de]**

btrfs-progs v4.15

See http://btrfs.wiki.kernel.org for more information.

Label: RAID0-BTRFS

UUID: 808842cd-afc2-42fb-9184-c5754ba7d3ea

Node size: 16384

Sector size: 4096

Filesystem size: 1.00GiB

Block group profiles:

Data: RAID0 128.00MiB

Metadata: RAID1 51.19MiB

System: RAID1 8.00MiB

SSD detected: no

Incompat features: extref, skinny-metadata

Number of devices: 2

Devices:

ID SIZE PATH

1 512.00MiB /dev/sdd

2 512.00MiB /dev/sde

lsauser@opensuse:~>

Should we want to change the mode data is stored, we must change the value after the **-d** switch to **raid1**, **raid5** or something else. In the same way, we can force the system and metadata to follow the data mode.

Now the mounting is the same as with other filesystems:

lsauser@opensuse:~> **sudo mkdir -p /disks/btrfs-{volume,raid}**

lsauser@opensuse:~> **sudo mount /dev/sdb /disks/btrfs-volume**

lsauser@opensuse:~> **sudo mount /dev/sdd /disks/btrfs-raid**

We can use any drive member of the appropriate group for mounting.

If we execute the **lsblk** command, we will get:

lsauser@opensuse:~> **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 8M 0 part

└─sda2 8:2 0 8G 0 part /

sdb 8:16 0 1G 0 disk /disks/btrfs-volume

sdc 8:32 0 2G 0 disk

sdd 8:48 0 512M 0 disk /disks/btrfs-raid

sde 8:64 0 512M 0 disk

sr0 11:0 1 1024M 0 rom

lsauser@opensuse:~>

One way to ask for free space is to use the well-known command **df**:

lsauser@opensuse:~> **df -h /disks/btrfs-\***

Файлова с-ма Размер Заети Свобод Изп% Монтирана на

/dev/sdd 1,0G 17M 904M 2% /disks/btrfs-raid

/dev/sdb 3,0G 17M 2,7G 1% /disks/btrfs-volume

lsauser@opensuse:~>

An alternative way is to use the **btrfs** command. For example:

lsauser@opensuse:~> **sudo btrfs filesystem show**

Label: 'VOLUME-BTRFS' uuid: fd667810-f59d-4d0f-bf2d-b8228efb2315

Total devices 2 FS bytes used 192.00KiB

devid 1 size 1.00GiB used 169.56MiB path /dev/sdb

devid 2 size 2.00GiB used 161.56MiB path /dev/sdc

Label: 'RAID0-BTRFS' uuid: 808842cd-afc2-42fb-9184-c5754ba7d3ea

Total devices 2 FS bytes used 128.00KiB

devid 1 size 512.00MiB used 123.19MiB path /dev/sdd

devid 2 size 512.00MiB used 123.19MiB path /dev/sde

lsauser@opensuse:~>

If we want to automount both filesystems by adding them to the **/etc/fstab**, we can use either the **UUID** value or any of the member devices, as we did earlier.

We can ask for detailed information for one of the filesystems with:

lsauser@opensuse:~> **sudo btrfs filesystem usage /disks/btrfs-raid**

Overall:

Device size: 1.00GiB

Device allocated: 246.38MiB

Device unallocated: 777.62MiB

Device missing: 0.00B

Used: 256.00KiB

Free (estimated): 905.62MiB (min: 516.81MiB)

Data ratio: 1.00

Metadata ratio: 2.00

Global reserve: 16.00MiB (used: 0.00B)

Data,RAID0: Size:128.00MiB, Used:0.00B

/dev/sdd 64.00MiB

/dev/sde 64.00MiB

Metadata,RAID1: Size:51.19MiB, Used:112.00KiB

/dev/sdd 51.19MiB

/dev/sde 51.19MiB

System,RAID1: Size:8.00MiB, Used:16.00KiB

/dev/sdd 8.00MiB

/dev/sde 8.00MiB

Unallocated:

/dev/sdd 388.81MiB

/dev/sde 388.81MiB

lsauser@opensuse:~>

Or use a sub-command similar to **df**:

lsauser@opensuse:~> **sudo btrfs filesystem df /disks/btrfs-raid**

Data, RAID0: total=128.00MiB, used=0.00B

System, RAID1: total=8.00MiB, used=16.00KiB

Metadata, RAID1: total=51.19MiB, used=112.00KiB

GlobalReserve, single: total=16.00MiB, used=0.00B

lsauser@opensuse:~>

Okay, but the filesystem is empty. We can create a file full of zeroes with:

lsauser@opensuse:~> **sudo dd if=/dev/zero of=/disks/btrfs-raid/file.dat bs=1M count=200**

200+0 прочетени блока

200+0 записани блока

209715200 bytes (210 MB, 200 MiB) copied, 0,303935 s, 690 MB/s

lsauser@opensuse:~>

Now, we can execute first:

lsauser@opensuse:~> **ls -alh /disks/btrfs-raid/**

общо 201M

drwxr-xr-x 1 root root 16 13 мар 16,23 .

drwxr-xr-x 4 root root 4,0K 13 мар 16,13 ..

-rw-r--r-- 1 root root 200M 13 мар 16,23 file.dat

lsauser@opensuse:~>

Then we can try with the regular **df** command:

lsauser@opensuse:~> **df -h /disks/btrfs-raid/**

Файлова с-ма Размер Заети Свобод Изп% Монтирана на

/dev/sdd 1,0G 217M 704M 24% /disks/btrfs-raid

lsauser@opensuse:~>

It can take some time to reflect the changes. We can force it with **sync**. More or less similar out we can get from the:

lsauser@opensuse:~> **sudo btrfs filesystem df /disks/btrfs-raid**

Data, RAID0: total=256.00MiB, used=200.12MiB

System, RAID1: total=8.00MiB, used=16.00KiB

Metadata, RAID1: total=51.19MiB, used=320.00KiB

GlobalReserve, single: total=16.00MiB, used=0.00B

lsauser@opensuse:~>

And if we want really detailed information, we can use:

lsauser@opensuse:~> **sudo btrfs filesystem usage /disks/btrfs-raid**

Overall:

Device size: 1.00GiB

Device allocated: 374.38MiB

Device unallocated: 649.62MiB

Device missing: 0.00B

Used: 200.78MiB

Free (estimated): 705.50MiB (min: 380.69MiB)

Data ratio: 1.00

Metadata ratio: 2.00

Global reserve: 16.00MiB (used: 0.00B)

Data,RAID0: Size:256.00MiB, Used:200.12MiB

/dev/sdd 128.00MiB

/dev/sde 128.00MiB

Metadata,RAID1: Size:51.19MiB, Used:320.00KiB

/dev/sdd 51.19MiB

/dev/sde 51.19MiB

System,RAID1: Size:8.00MiB, Used:16.00KiB

/dev/sdd 8.00MiB

/dev/sde 8.00MiB

Unallocated:

/dev/sdd 324.81MiB

/dev/sde 324.81MiB

lsauser@opensuse:~>

## RAID with MDRaid

For this part we will use an **AlmaLinux** **VM**. Let's check where we start from:

[lsauser@almalinux ~]$ **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 7G 0 part

├─almalinux-root 253:0 0 6,2G 0 lvm /

└─almalinux-swap 253:1 0 820M 0 lvm [SWAP]

sr0 11:0 1 1024M 0 rom

[lsauser@almalinux ~]$

Now, let's turn off the **VM** and add four additional hard disks each with **1GB** of size.

Then power on the **VM** again and check the situation:

[lsauser@almalinux ~]$ **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 7G 0 part

├─almalinux-root 253:0 0 6,2G 0 lvm /

└─almalinux-swap 253:1 0 820M 0 lvm [SWAP]

sdb 8:16 0 1G 0 disk

sdc 8:32 0 1G 0 disk

sdd 8:48 0 1G 0 disk

sde 8:64 0 1G 0 disk

sr0 11:0 1 1024M 0 rom

[lsauser@almalinux ~]$

We should check if **mdadm** is installed, and if not – install it with:

[lsauser@almalinux ~]$ **sudo dnf install -y mdadm**

...

*Use the package manager for your distribution.*

Before we can continue with **RAID** creation, we must prepare the disks. For this purpose, we will create one primary partition on each disk with type **fd** (Linux raid autodetect). The procedure for the first spare drive (**/dev/sdb**) is:

[lsauser@almalinux ~]$ **sudo fdisk /dev/sdb**

...

Command (m for help): **n**

Partition type:

p primary (0 primary, 0 extended, 4 free)

e extended

Select (default p): **p**

Partition number (1-4, default 1): **1**

First sector (2048-2097151, default 2048):

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151):

Using default value 2097151

Partition 1 of type Linux and of size 1023 MiB is set

Command (m for help): **t**

Selected partition 1

Hex code (type L to list all codes): **fd**

Changed type of partition 'Linux' to 'Linux raid autodetect'

Command (m for help): **p**

Disk /dev/sdb: 1073 MB, 1073741824 bytes, 2097152 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk label type: dos

Disk identifier: 0x08e5f00b

Device Boot Start End Blocks Id System

/dev/sdb1 2048 2097151 1047552 fd Linux raid autodetect

Command (m for help): **w**

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

[lsauser@almalinux ~]$

We must repeat this for the other three spare drives (**/dev/sdc**, **/dev/sdd** and **/dev/sde**).

An alternative approach is to follow the scripted approach and use the **parted** utility. In fact, we will execute four similar commands, which will contain all the arguments needed to fulfill the task:

[lsauser@almalinux ~]$ **sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 1m -0m set 1 raid on**

[lsauser@almalinux ~]$ **sudo parted -s /dev/sdc -- mklabel msdos mkpart primary 1m -0m set 1 raid on**

[lsauser@almalinux ~]$ **sudo parted -s /dev/sdd -- mklabel msdos mkpart primary 1m -0m set 1 raid on**

[lsauser@almalinux ~]$ **sudo parted -s /dev/sde -- mklabel msdos mkpart primary 1m -0m set 1 raid on**

Either way – interactive approach with **fdisk**, or scripted approach with **parted**, at the end we should have four spare drives each with single partition. The **lsblk** output should now be similar to this:

[lsauser@almalinux ~]$ **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 7G 0 part

├─almalinux-root 253:0 0 6,2G 0 lvm /

└─almalinux-swap 253:1 0 820M 0 lvm [SWAP]

sdb 8:16 0 1G 0 disk

└─sdb1 8:17 0 1023M 0 part

sdc 8:32 0 1G 0 disk

└─sdc1 8:33 0 1023M 0 part

sdd 8:48 0 1G 0 disk

└─sdd1 8:49 0 1023M 0 part

sde 8:64 0 1G 0 disk

└─sde1 8:65 0 1023M 0 part

sr0 11:0 1 1024M 0 rom

[lsauser@almalinux ~]$

We are ready to proceed with **RAID** creation. First, we will create one **RAID0** array:

[lsauser@almalinux ~]$ **sudo mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1**

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md0 started.

We can check **RAID** status either with this:

[lsauser@almalinux ~]$ **cat /proc/mdstat**

Personalities : [raid0]

md0 : active raid0 sdc1[1] sdb1[0]

2091008 blocks super 1.2 512k chunks

unused devices: <none>

Or we can ask for more detailed information with this:

[lsauser@almalinux ~]$ **sudo mdadm -D /dev/md0**

/dev/md0:

Version : 1.2

Creation Time : Thu Mar 14 12:16:55 2019

Raid Level : raid0

Array Size : 2091008 (2042.00 MiB 2141.19 MB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu Mar 14 12:16:55 2019

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Chunk Size : 512K

Consistency Policy : none

Name : almalinux:0 (local to host almalinux)

UUID : c2321fb6:799c84a3:1c3839dc:4b5ba067

Events : 0

Number Major Minor RaidDevice State

0 8 17 0 active sync /dev/sdb1

1 8 33 1 active sync /dev/sdc1

[lsauser@almalinux ~]$

In a similar fashion we can create **RAID1** array out of the other two spare disks:

[lsauser@almalinux ~]$ **sudo mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdd1 /dev/sde1**

mdadm: Note: this array has metadata at the start and

may not be suitable as a boot device. If you plan to

store '/boot' on this device please ensure that

your boot-loader understands md/v1.x metadata, or use

--metadata=0.90

Continue creating array? **yes**

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md1 started.

[lsauser@almalinux ~]$ **cat /proc/mdstat**

Personalities : [raid0] [raid1]

md1 : active raid1 sde1[1] sdd1[0]

1046528 blocks super 1.2 [2/2] [UU]

md0 : active raid0 sdc1[1] sdb1[0]

2091008 blocks super 1.2 512k chunks

unused devices: <none>

[lsauser@almalinux ~]$ **sudo mdadm -D /dev/md1**

/dev/md1:

Version : 1.2

Creation Time : Thu Mar 14 12:24:13 2019

Raid Level : raid1

Array Size : 1046528 (1022.00 MiB 1071.64 MB)

Used Dev Size : 1046528 (1022.00 MiB 1071.64 MB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu Mar 14 12:24:16 2019

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Consistency Policy : resync

Name : almalinux:1 (local to host almalinux)

UUID : c0c75f3a:75857e0c:e79bd352:2e25e062

Events : 17

Number Major Minor RaidDevice State

0 8 49 0 active sync /dev/sdd1

1 8 65 1 active sync /dev/sde1

[lsauser@almalinux ~]$

Now, we must take one more step to ensure that both RAID arrays will be reassembled on boot (the file for **Debian**/**Ubuntu** is **/etc/mdadm/mdadm.conf**):

[lsauser@almalinux ~]$ **sudo mdadm --detail --scan | sudo tee -a /etc/mdadm.conf**

ARRAY /dev/md/0 metadata=1.2 name=almalinux:0 UUID=c2321fb6:799c84a3:1c3839dc:4b5ba067

ARRAY /dev/md/1 metadata=1.2 name=almalinux:1 UUID=c0c75f3a:75857e0c:e79bd352:2e25e062

[lsauser@almalinux ~]$

It is time to create filesystems for each **RAID** array. Here we will use ext4:

[lsauser@almalinux ~]$ **sudo mkfs.ext4 /dev/md0**

...

[lsauser@almalinux ~]$ **sudo mkfs.ext4 /dev/md1**

...

[lsauser@almalinux ~]$

Then, we must prepare the mount points:

[lsauser@almalinux ~]$ **sudo mkdir -p /disks/raid{0,1}**

The final step is to add records in the **/etc/fstab** file. This can be done either by opening the file in text editor, or by appending information in it with:

[lsauser@almalinux ~]$ **echo '/dev/md0 /disks/raid0 ext4 defaults 0 0' | sudo tee -a /etc/fstab**

/dev/md0 /disks/raid0 ext4 defaults 0 0

[lsauser@almalinux ~]$ **echo '/dev/md1 /disks/raid1 ext4 defaults 0 0' | sudo tee -a /etc/fstab**

/dev/md1 /disks/raid1 ext4 defaults 0 0

[lsauser@almalinux ~]$

Now, we can reboot, and once the system is up and running again, we can check how both arrays are seen:

[lsauser@almalinux ~]$ **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 8G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 7G 0 part

├─almalinux-root 253:0 0 6,2G 0 lvm /

└─almalinux-swap 253:1 0 820M 0 lvm [SWAP]

sdb 8:16 0 1G 0 disk

└─sdb1 8:17 0 1023M 0 part

└─md0 9:0 0 2G 0 raid0 /disks/raid0

sdc 8:32 0 1G 0 disk

└─sdc1 8:33 0 1023M 0 part

└─md0 9:0 0 2G 0 raid0 /disks/raid0

sdd 8:48 0 1G 0 disk

└─sdd1 8:49 0 1023M 0 part

└─md1 9:1 0 1022M 0 raid1 /disks/raid1

sde 8:64 0 1G 0 disk

└─sde1 8:65 0 1023M 0 part

└─md1 9:1 0 1022M 0 raid1 /disks/raid1

sr0 11:0 1 1024M 0 rom

[lsauser@almalinux ~]$ **df -h**

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/almalinux-root 6,2G 1,2G 5,1G 19% /

devtmpfs 484M 0 484M 0% /dev

tmpfs 496M 0 496M 0% /dev/shm

tmpfs 496M 6,8M 489M 2% /run

tmpfs 496M 0 496M 0% /sys/fs/cgroup

/dev/sda1 1014M 132M 883M 13% /boot

/dev/md0 2,0G 6,0M 1,9G 1% /disks/raid0

/dev/md1 990M 2,6M 921M 1% /disks/raid1

tmpfs 100M 0 100M 0% /run/user/1000

[lsauser@almalinux ~]$

## Using Physical USB Device

In this exercise we will take all necessary steps to use a physical **USB** flash drive formatted with **FAT32** in a **VM** running in **VirtualBox**. The steps are more or less the same in other virtualization solutions.

We can continue with the machine from the previous exercise or create/import a new one. It could be running any distribution, but next steps are executed on **AlmaLinux**.

Our **VM** must be in a running state. Then we must execute the following preparation steps:

* Plug in the **USB** stick
* Choose the **USB** drive from the **Devices > USB > Kingston USB** menu. Your **USB** drive most probably will have different name. This way we are attaching the device directly to the **VM**

Now, in a terminal session, execute:

[lsauser@almalinux ~]$ **lsblk**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

...

**sdf 8:80 1 3,8G 0 disk**

**└─sdf1 8:81 1 3,8G 0 part**

...

[lsauser@almalinux ~]$

The name and size of the device will be different in your case.

We can create a dedicated folder to mount the **USB** drive's filesystem or use existing one. Let's create one:

[lsauser@almalinux ~]$ **sudo mkdir -p /media/usb**

Now that we know the path to the device, we can mount it:

[lsauser@almalinux ~]$ **sudo mount /dev/sdf1 /media/usb**

If we check the available block devices:

[lsauser@almalinux ~]$ **lsblk -f**

NAME FSTYPE LABEL UUID MOUNTPOINT

...

**sdf**

**└─sdf1 vfat DT 4GB FA4D-608E /media/usb**

...

[lsauser@almalinux ~]$

Let's create a small text file on the mounted flash drive:

[lsauser@almalinux ~]$ **echo 'Hello from Linux!' | sudo tee /media/usb/hello.txt**

Hello from Linux!

[lsauser@almalinux ~]$ **ls -al /media/usb/**

общо 460

drwxr-xr-x. 3 root root 4096 14 мар 17,33 .

drwxr-xr-x. 3 root root 17 14 мар 17,28 ..

-rwxr-xr-x. 1 root root 18 14 мар 17,33 hello.txt

-rwxr-xr-x. 1 root root 457281 13 мар 16,37 M6-Practice-FHS-Disks-Filesystems-and-Archives.docx

drwxr-xr-x. 2 root root 4096 9 дек 15,14 System Volume Information

[lsauser@almalinux ~]$

Now, let's unmount it:

[lsauser@almalinux ~]$ **sudo umount /media/usb**

[lsauser@almalinux ~]$ **ls -al /media/usb/**

общо 0

drwxr-xr-x. 2 root root 6 14 мар 17,28 .

drwxr-xr-x. 3 root root 17 14 мар 17,28 ..

[lsauser@almalinux ~]$

This gives us a valid option for data exchange between any **VM** and the host.