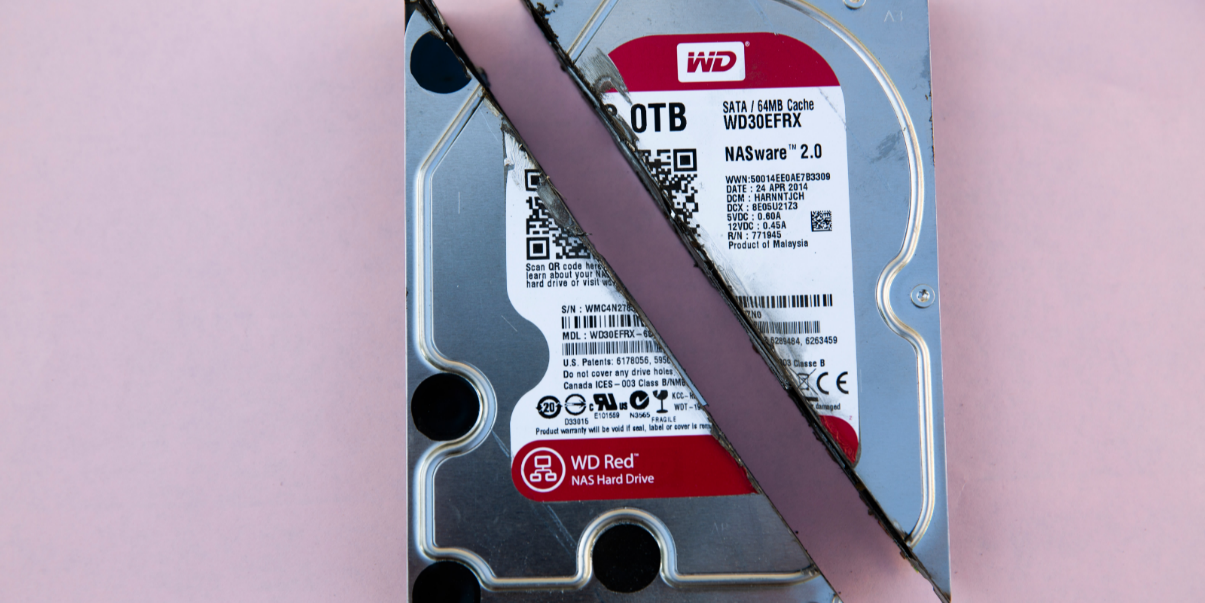
# Linux Disk Management

* [](https://devopsroot.com/author/alexandru/)

[**Alexandru Calcatinge**](https://devopsroot.com/author/alexandru/)

Dec 1, 2018 • 27 min read



In this laboratory, you will exercise using disk devices, partitions, RAID, LVM, swap and cryptsetup.

## Contents:

1. Lab 1 Partitioning and formatting disks

* Lab objective 1: Using a file as a disk partition image
* Lab objective 2: Partitioning a disk image file
* Lab objective 3: Using losetup and parted
* Lab objective 4: Partitioning and formatting a real hard drive/stick
* Lab objective 5: Working with swap

1. Lab 2 Encryption and filesystem features

* Lab objective 6: Disk encryption
* Lab objective 7: Encrypted swap

1. Lab 3 Using LVM

* Lab objective 8: Managing logical volumes
* Lab objective 9: Creating a RAID device

# Part I – Partitioning and formatting disks

## Laboratory objective 1: Using a file as a disk partition image

Commands used: dd, mkfs, ls, mount, mkdir

Solution:

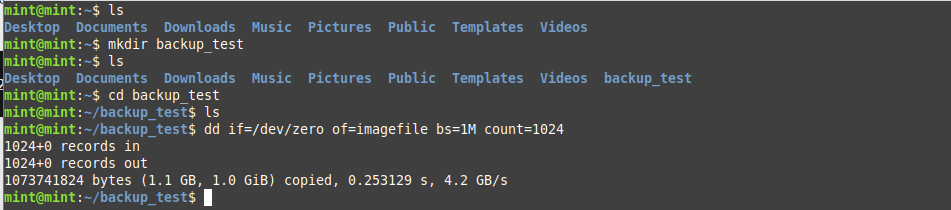
For the purpose of this exercise, you will need unpartitioned disk space available on your disk. If you don't, and you are using your own system, you will need to shrink a partition and the filesystem on it first, and then make it available to the system. If you don't use partitions on your system, only one large partition for the entire operating system, the you can use the loop device mechanism with or without the parted program. In the first exercises we will use the loop mechanism.

We are going to create a file that will be used as a container for a full hard disk partition image, and thus it can be used as a real partitioned hard disk.

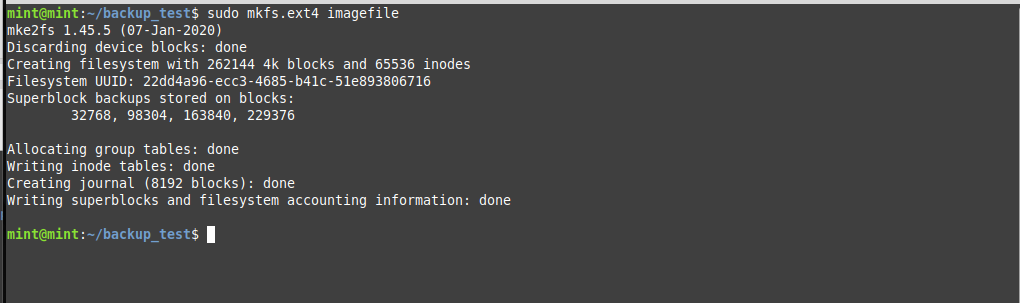
1. Create a file full of zeros that will be 1GB in length

dd if=/dev/zero of=imagefile bs=1M count=1024

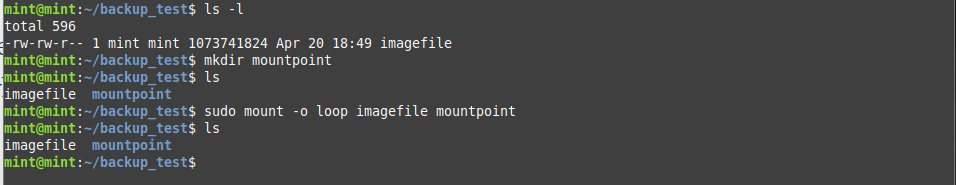
For this, I created a test directory in my home directory, and the output is:



1. Put a file system on the file you just created. For convenience, I created a ext4 file system, but you can user another one, such as ext3, vfat, xfs or btrfs.



1. Next, mount the file you created. For this, I created a new directory called mount point inside my backup\_test directory, where the image file was created



## Laboratory objective 2: Partitioning a disk image file

Commands used: fdisk, umount

Solution:

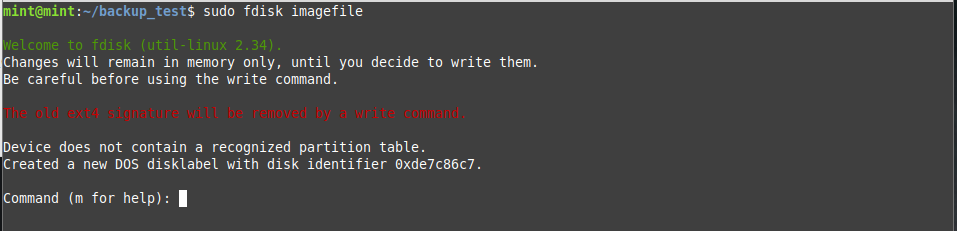
In the previous exercise, you created an image file that was formatted as a disk drive. Now, the next level of complication is to divide the container file into multiple partitions, in order to hold a file system or a swap area. We will reuse the imagefile that we created earlier.

1. run fdisk on the imagefile

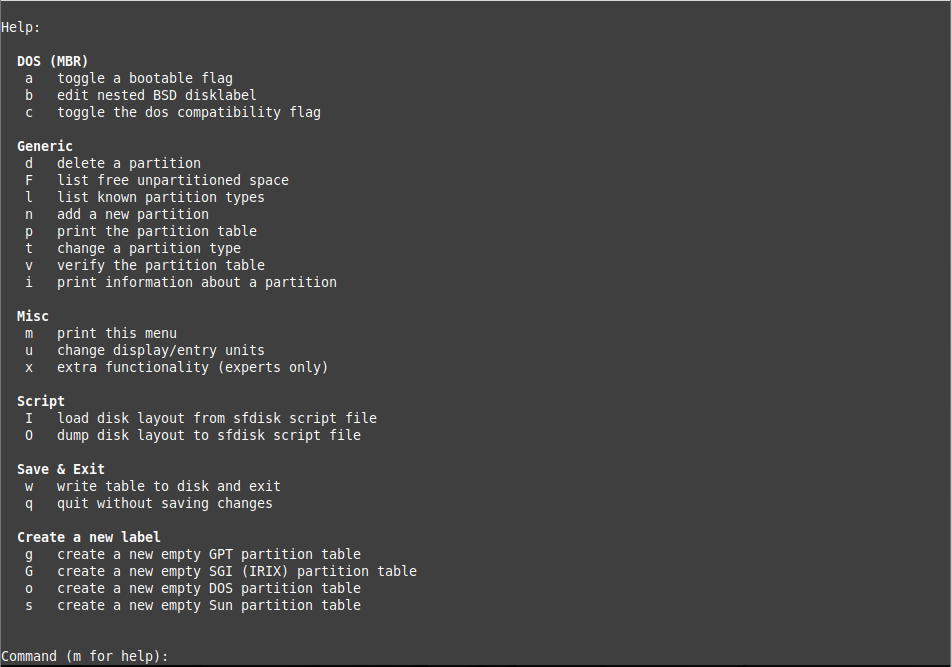
When finished with the above Lab, please make sure to unmount the imagefile using the unmount command. Note: In Linux Mint, the unmount command did not work. I unmounted the drive using the GUI.

Next, enter the following command:

sudo fdisk imagefile

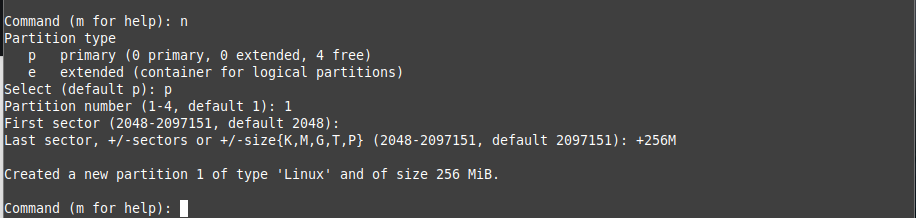


1. type m to get a list of commands



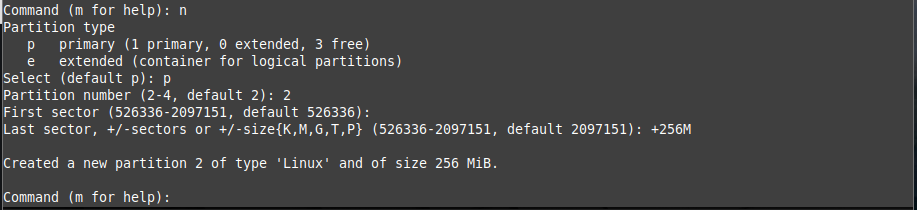
1. Create a new primary partition and make it 256MB

Command (m for help): n

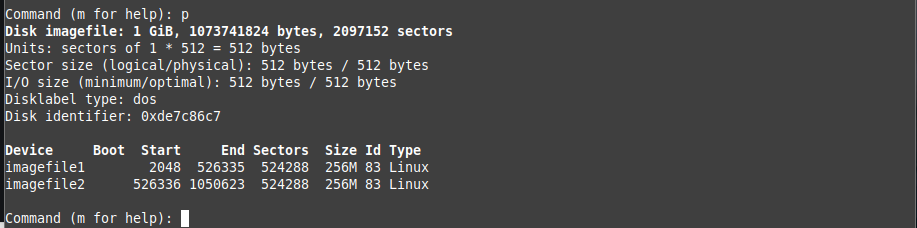


1. Add a secondary primary partition also of 256MB in size

Command (m for help): n



Now print the partition table by hitting the p key



1. write the partition table to disk and exit

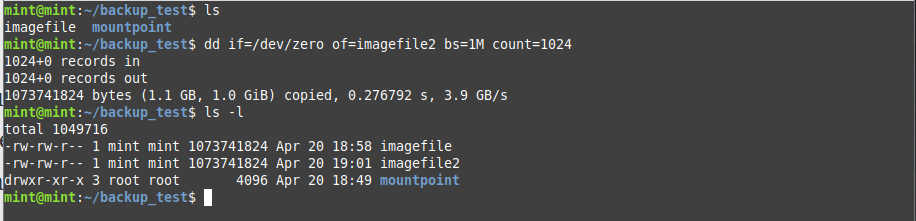
### 

## Laboratory objective 3: Using losetup and parted

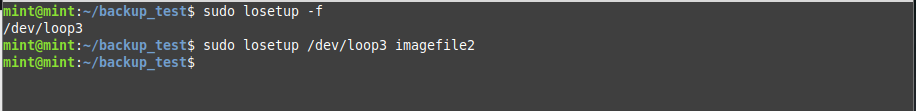
Commands used: losetup, parted, fdisk, ls, mkfs, mount, mkdir, df, umount, rmdir

Solution:

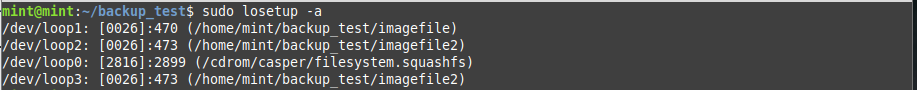
In this exercise, you are going to experiment with loop devices, losetup and parted to partition at the command line. You can use the previous file, or you can create a new one. For the purpose of this exercise, better create a new one.



1. associate the image file with a loop device



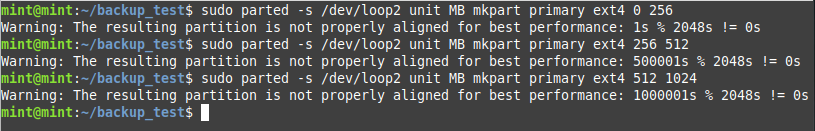
With the first command, we find the first loop device that is free, just to make sure in case you already use others on your system. With the option –a you can find all the currently used loop devices.



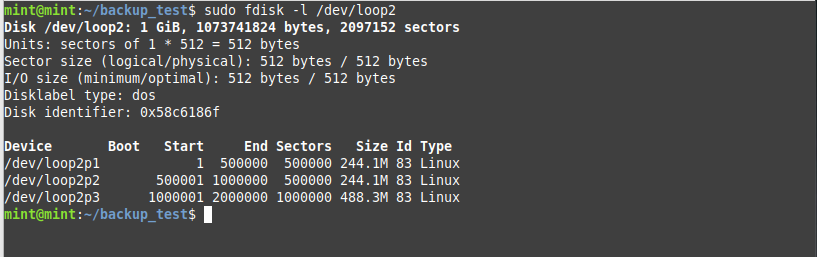
1. Create a disk partition label on the loop device image file



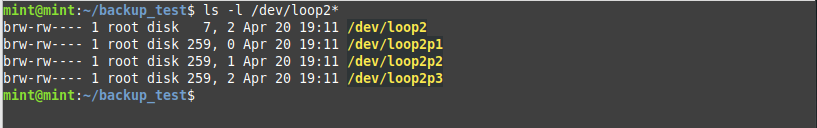
1. Create three primary partitions on the loop device



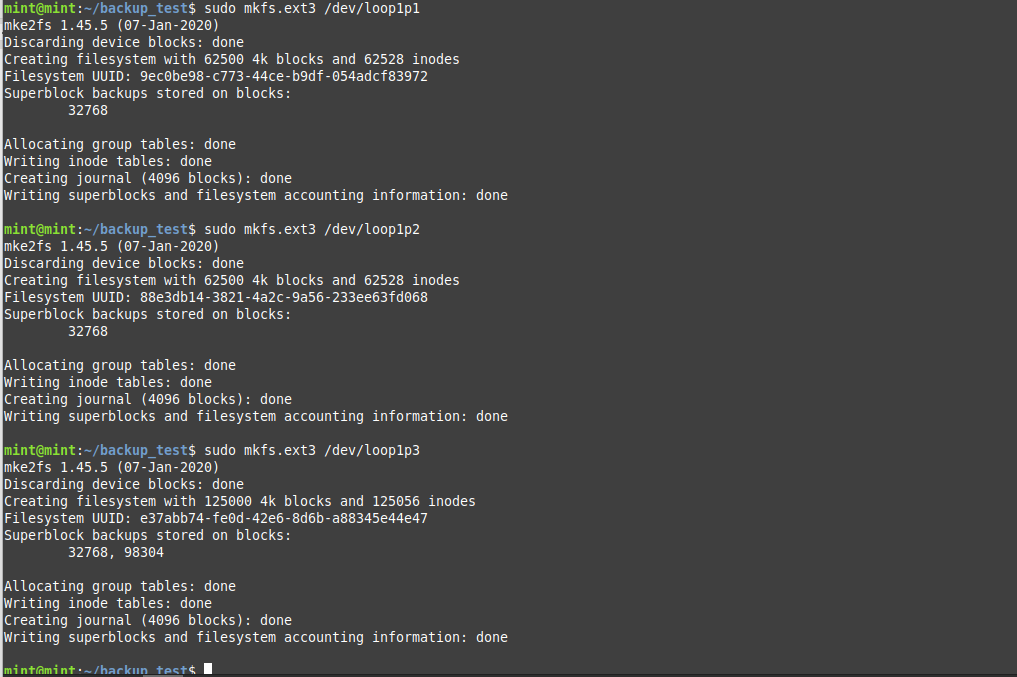
1. check the partition table



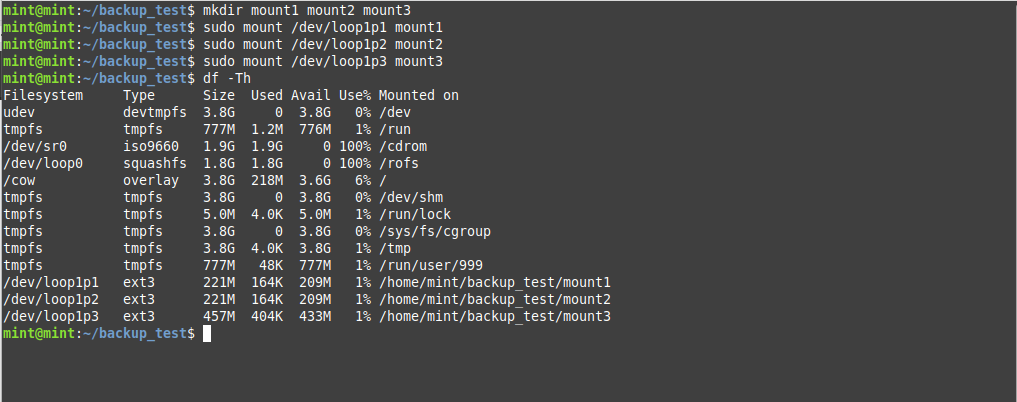
1. check if the device nodes have been created. This depends on your distribution. In open SUSE 42.x, RHEL 7 and Ubuntu (starting 14.04), the following command will work. On older distributions, it might not work.



1. put filesystems on the partitions

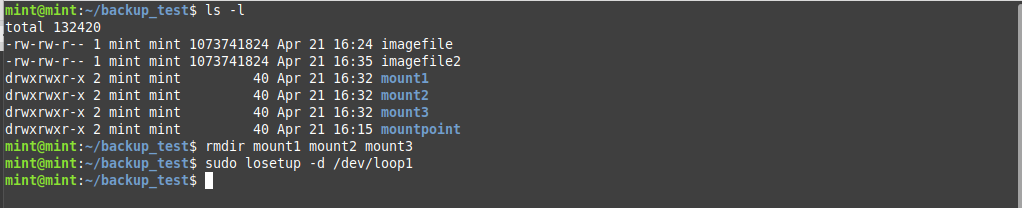


1. mount all three filesystems and show that they are available



1. after using the filesystems to your needs, you can unmount it all

\*I unmounted the drives using the GUI…



## Laboratory objective 4: Partitioning and formatting a hard drive/stick

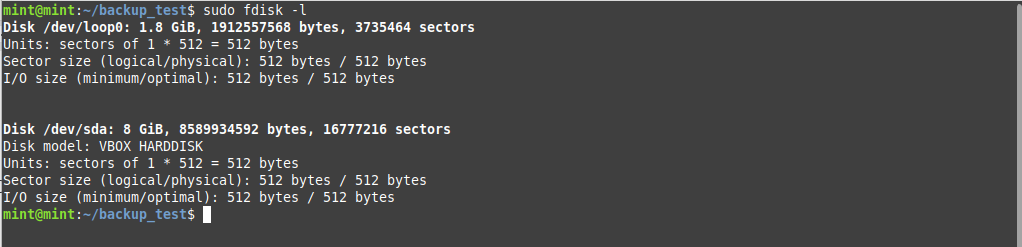
Commands used: fdisk, mount, umount, mkdir, mkfs, ls

Solution:

Now we will exercise partitioning and formatting with a real memory stick or external hard drive. Depending on what you have available, you can use either of those.

I have a 32GB memory stick that I will use for this exercise.

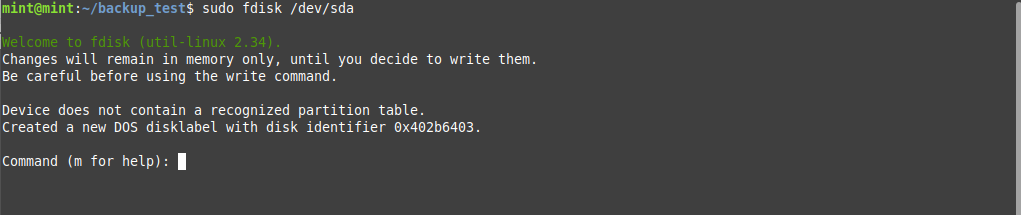
1. Insert your memory stick into your computer's USB port and use fdisk to see details about it.



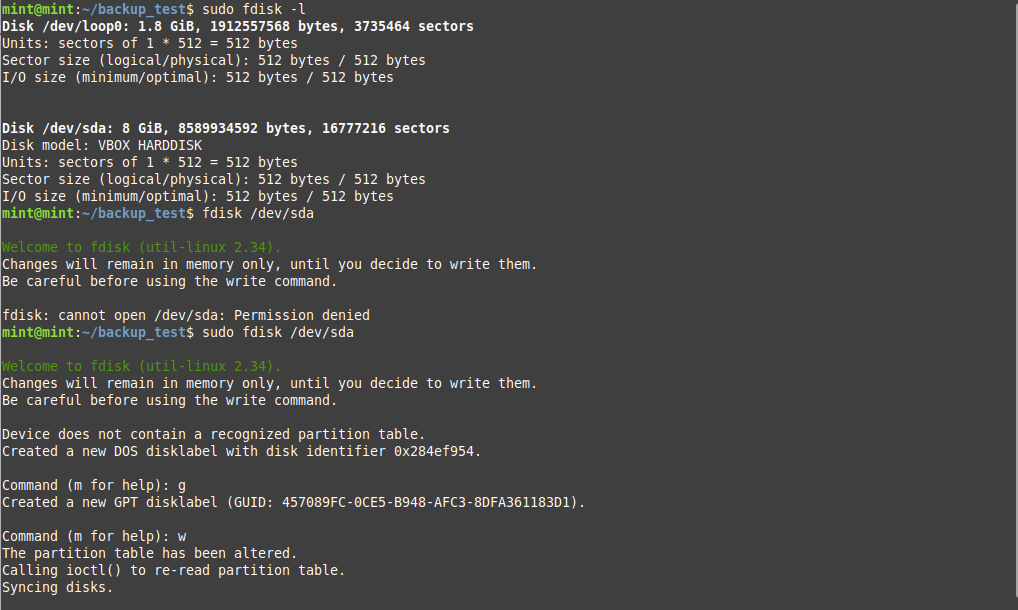
You will see that your memory stick is under /dev/sdb

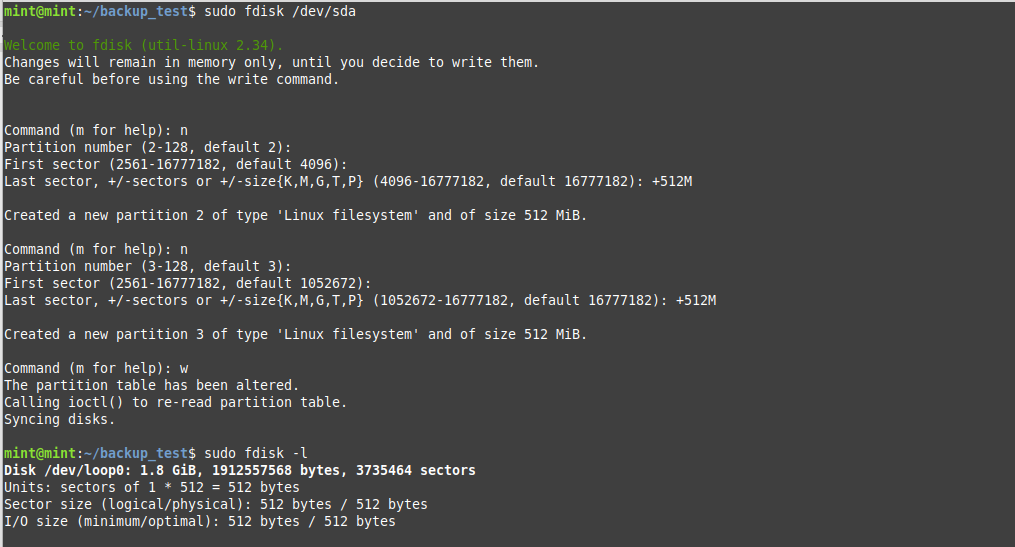
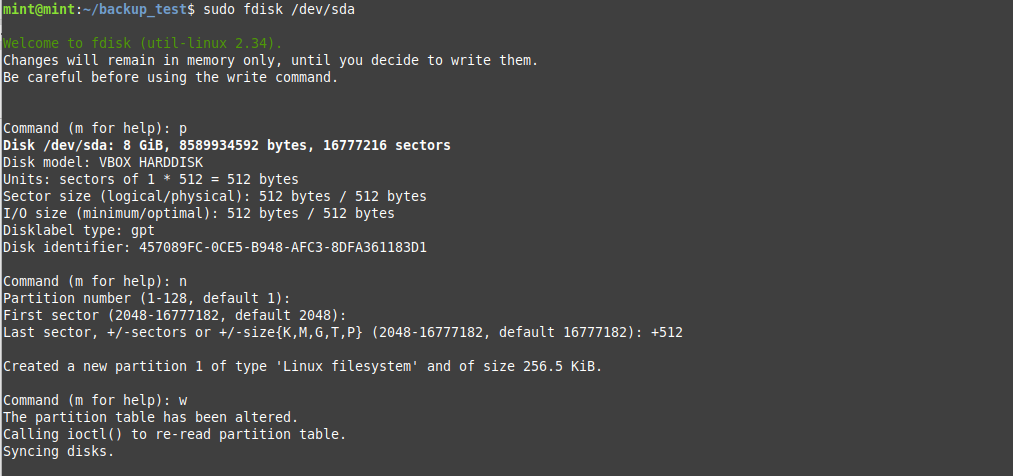
In the example above, the memory stick is not recognized by the VM.

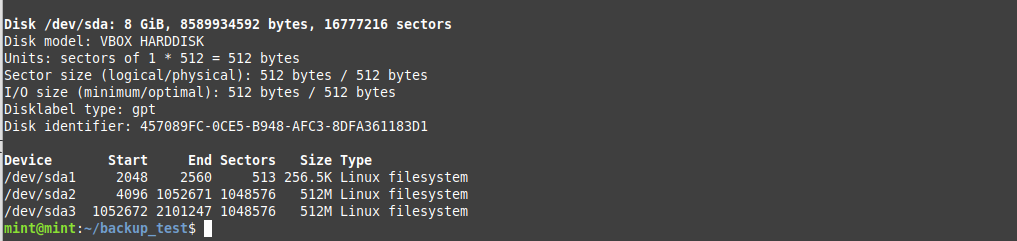
1. Now use fdisk for the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



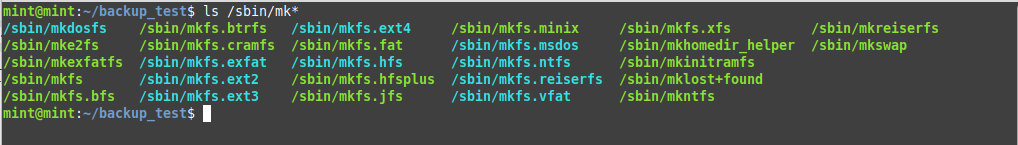
1. Now we will create three different partitions, as follows:



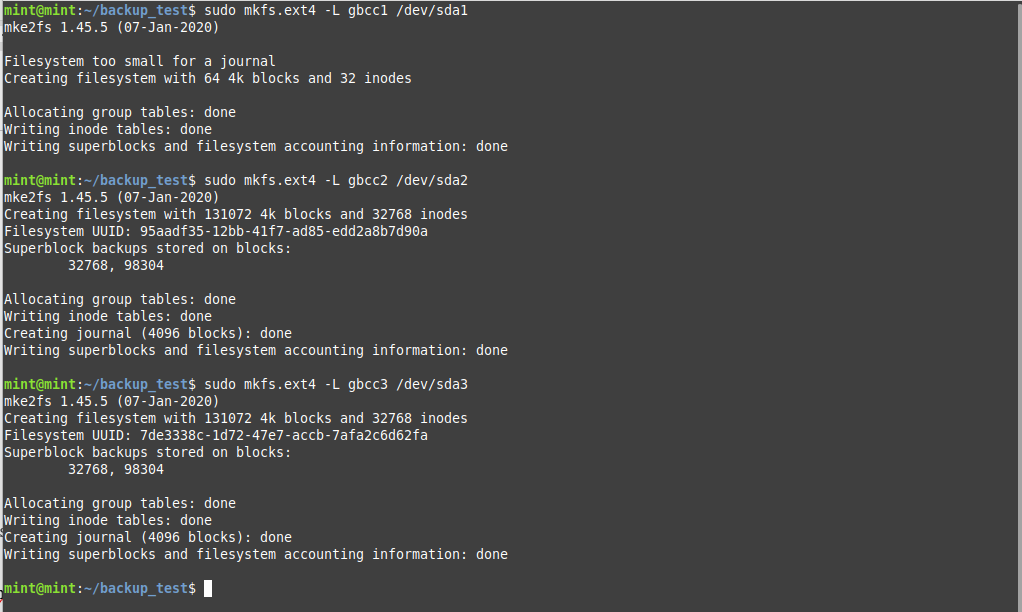


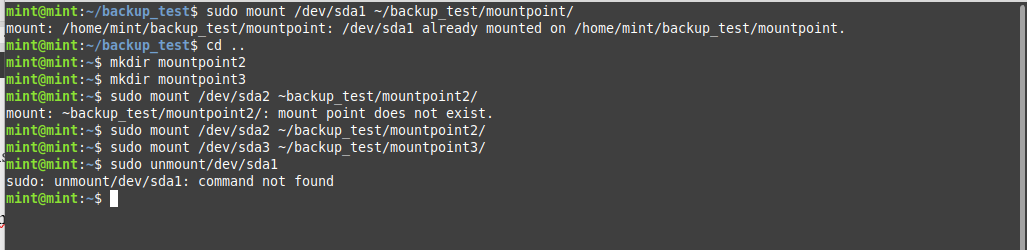


Now we have three different partitions on the external memory stick, as shown in the upper lines. The partitions are not formatted. To view the supported filesystems on our distribution, we can run the command:



You should choose your filesystem based on your requirements and needs. Most used filesystem type is ext4, supported by most distributions, but there also are XFS or btrfs that are also used by enterprise grade Linux such as SUSE. For convenience I will use ext4, even though I use XFS and btrfs on my SUSE systems. I will label the three partitions: "gbcc1", "gbcc2", "gbcc3", thus I will use the -L option for mkfs command:



1. Now lets mount the device partitions. I will use the same directory as in the previous exercise, but I will create two other directories, for each partition.
2. Now unmount the partitions:

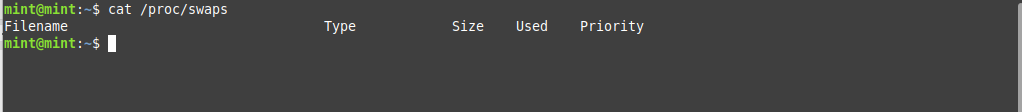
### I unmounted the drives using the GUI

## Laboratory objective 5: Working with swap

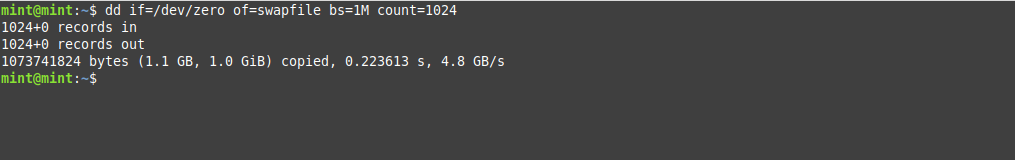
Commands used: cat, dd, mkswap, cp, swapon, chown, chmod, swapoff, rm

Solution:

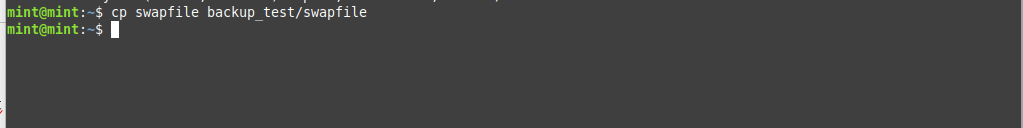
1. Examine your current swap space with the command:



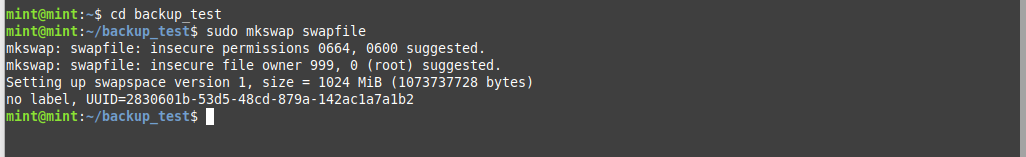
1. We will now add more swap space to the system by adding a new swap file. You can add more space with a new partition, if you have one unused.



1. Now format the file as a swap partition. First I will move the swapfile into the /backup\_test directory for more convenience.



Now I will format the swap partition:



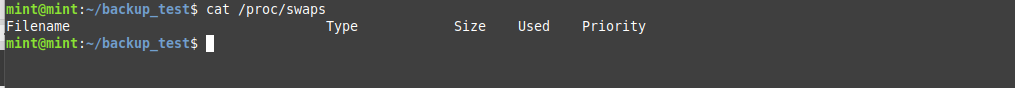
1. Activate the new swap space:



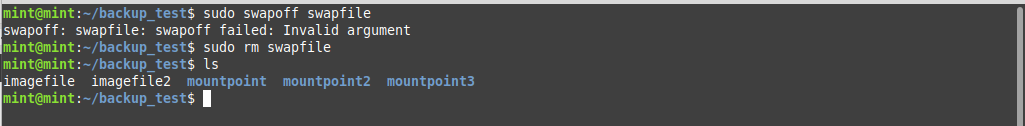
You notice that the distribution is warning us that the new swap is insecure, so we will need to take extra measures and fix this:



1. Now make sure that it is being used:



1. If you are not using the file, you can then remove it from use and delete it from your system with the commands:



We did an ls to make sure that the file was removed.

# Part II: Encryption and Filesystem Features

## Laboratory objective 6: Disk encryption

Commands used: partprobe, umount, cryptsetup, mount, fdisk, mount

Solution:

1. Create a new partition for the encrypted block device with fdisk, make sure that the kernel is aware of the new partition table, without rebooting.

You can create a new partition, but we will use one of the partitions from the memory stick we used in the previous exercise. You can do the same. To make the kernel aware of a new partition table, use the command:

sudo partprobe -s

alexandru@linux-vje9:~/backup\_test> sudo partprobe -s

[sudo] password for root:

/dev/sda: gpt partitions 1 2 3

/dev/sdc: gpt partitions 1 2 3

1. Format one of the partitions on the memory stick with cryptsetup using LUKS for the crypto layer. If you don't have cryptsetup installed on your system, install it using your software management tools. We will encrypt the second partition /dev/sdc2

alexandru@linux-vje9:~> sudo umount /dev/sdc2

alexandru@linux-vje9:~> sudo cryptsetup luksFormat /dev/sdc2

WARNING!

This will overwrite data on /dev/sdc2 irrevocably.

Are you sure? (Type uppercase yes): YES

Enter passphrase:

Verify passphrase:

Notice that first, you have to unmount the partition that you are planning on encrypting.

1. create an un-encrypted pass through device by opening the encrypted block device

alexandru@linux-vje9:~> sudo cryptsetup luksOpen /dev/sdc2 secret-disk

Enter passphrase for /dev/sdc2:

1. Add an entry to /etc/crypttab so that the system prompts for the passphrase on reboot:

sudo vim /etc/crypttab

secret-disk /dev/sdc2

1. Format the filesystem as an ext4 filesystem

alexandru@linux-vje9:~> sudo mkfs.ext4 /dev/mapper/secret-disk

mke2fs 1.42.11 (09-Jul-2014)

Creating filesystem with 2620928 4k blocks and 655360 inodes

Filesystem UUID: dd1635b9-8a09-42b6-949c-8b72f0db5e35

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done

Writing inode tables: done

Creating journal (32768 blocks): done

Writing superblocks and filesystem accounting information: done

1. Create a mount point for the new filesystem (for example /secret)

alexandru@linux-vje9:~> cd backup\_test/

alexandru@linux-vje9:~/backup\_test> ls

imagefile imagefile2 imagefilex mnt mountpoint mountpoint2 mountpoint3

alexandru@linux-vje9:~/backup\_test> sudo mkdir -p secret

alexandru@linux-vje9:~/backup\_test> ls

imagefile imagefilex mountpoint mountpoint3

imagefile2 mnt mountpoint2 secret

alexandru@linux-vje9:~/backup\_test>

1. Add an entry to /etc/fstab so that the filesystem is mounted on boot

/dev/mapper/secret-disk /home/alexandru/backup\_test/secret ext4 defaults 1 2

1. try and mount the encrypted filesystem and verify by running fdisk -l:

alexandru@linux-vje9:~/backup\_test> sudo mount -a

alexandru@linux-vje9:~/backup\_test> sudo fdisk -l

Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 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

Disklabel type: gpt

Disk identifier: E6E55437-CAAE-42EC-A5C0-233DC0CF4CE0

Device Start End Sectors Size Type

/dev/sda1 2048 8386559 8384512 4G Microsoft basic data

/dev/sda2 8386560 16771071 8384512 4G EFI System

/dev/sda3 16771072 234440703 217669632 103.8G Microsoft basic data

Disk /dev/sdc: 29.8 GiB, 32015679488 bytes, 62530624 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

Disklabel type: gpt

Disk identifier: CE694922-069C-43F2-8BBA-D8FC3580EA6B

Device Start End Sectors Size Type

/dev/sdc1 2048 20973567 20971520 10G Linux filesystem

/dev/sdc2 20973568 41945087 20971520 10G Linux filesystem

/dev/sdc3 41945088 62530590 20585503 9.8G Linux filesystem

Disk /dev/mapper/secret-disk: 10 GiB, 10735321088 bytes, 20967424 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

You will have to unmount all the partitions, including the encrypted one, using the commands:

alexandru@linux-vje9:~/backup\_test> sudo umount /dev/sdc1

[sudo] password for root:

alexandru@linux-vje9:~/backup\_test> sudo umount /dev/sdc2

umount: /dev/sdc2: not mounted

alexandru@linux-vje9:~/backup\_test> sudo umount /dev/sdc3

alexandru@linux-vje9:~/backup\_test> sudo umount /dev/mapper/secret-disk

## Laboratory objective 7: Encrypted swap

Commands used: swapoff, swapon, cryptsetup, mkswap, cat

Solution:

In this exercise we will work with the active swap partition on your hard drive, so you need to be extra careful on what you do!!!!

1. Find out what partition you are currently using for swap and then deactivate it:

alexandru@linux-vje9:~/backup\_test> cat /proc/swaps

Filename Type Size Used Priority

/dev/sda1 partition 4192252 1916 -1

alexandru@linux-vje9:~/backup\_test> sudo swapoff /dev/sda1

1. do the same steps as in the previous exercise to set up encryption:

alexandru@linux-vje9:~/backup\_test> sudo cryptsetup luksFormat /dev/sda1

WARNING!

This will overwrite data on /dev/sda1 irrevocably.

Are you sure? (Type uppercase yes): YES

Enter passphrase:

Verify passphrase:

alexandru@linux-vje9:~/backup\_test> sudo cryptsetup luksOpen /dev/sda1 swapcrypt

Enter passphrase for /dev/sda1:

alexandru@linux-vje9:~/backup\_test>

1. Format the encrypted device to use with swap:

alexandru@linux-vje9:~/backup\_test> sudo mkswap /dev/mapper/swapcrypt

Setting up swapspace version 1, size = 4 GiB (4290768896 bytes)

no label, UUID=c4c1ed09-f29d-427a-b3c3-85de230ac2b7

1. Now test to see if it actually works by activating it:

alexandru@linux-vje9:~/backup\_test> sudo swapon /dev/mapper/swapcrypt

alexandru@linux-vje9:~/backup\_test> cat /proc/swaps

Filename Type Size Used Priority

/dev/dm-1 partition 4190204 0 -1

1. To make sure that the encrypted swap partition can be activated at boot, you need to do two things:  
   1. Add a line to /etc/crypttab so that the system prompts for the passphrase on reboot:

swapcrypt /dev/sda1 /dev/urandom swap,cipher=aes-cbc-essiv:sha256,size=256

* 1. Add an entry to the /etc/fstab so that the swap device is activated on boot

/dev/mapper/swapcrypt none swap defaults 0 0

1. You can reboot and see if everything works.
2. To restore your original unencrypted swap partition, do the following:

sudo swapoff /dev/mapper/swapcrypt

sudo cryptsetup luksClose swapcrypt

sudo mkswap /dev/sda1

sudo swapon -a

## Part III: Using LVM

## Laboratory objective 8: Managing logical volumes

Commands used: fdisk, partprobe, pvcreate, pvdisplay, vgcreate, vgdisplay, lvcreate, lvdisplay, mkfs, mkdir, mount, df, lvextend, resize2fs

Solution:

We are going to create a logical volume using two 250 MB partitions. We will assume that you have real partitionable disk space available. I will use an external 32 GB memory stick for this exercise. Note that this exercise was done using Debian 9.1 GNU/Linux.

1. Create two 250 MB partitions of type (in my case is /dev/sdb)

sudo fdisk /dev/sdb

type n and follow the on screen messages, type Y to remove the ext4 signature, type t to change the partition type and select 31 (Linux LVM), type 2 to write changes to disk. Do this twice, for the second partition too. Then execute:

sudo partprobe -s

1. Convert the partitions to physical volumes. The new partitions are: /dev/sdb1 and /dev/sdb2, thus execute:

alexandru@debian:~$ sudo pvcreate /dev/sdb1

Physical volume "/dev/sdb1" successfully created.

alexandru@debian:~$ sudo pvcreate /dev/sdb2

Physical volume "/dev/sdb2" successfully created.

alexandru@debian:~$ sudo pvdisplay

--- Physical volume ---

PV Name /dev/mapper/sda5\_crypt

VG Name debian-vg

PV Size 111.55 GiB / not usable 2.00 MiB

Allocatable yes (but full)

PE Size 4.00 MiB

Total PE 28556

Free PE 0

Allocated PE 28556

PV UUID WD9D5n-PX1K-TWAf-gHHu-OtHY-XzdM-W2NCZJ

"/dev/sdb2" is a new physical volume of "250.00 MiB"

--- NEW Physical volume ---

PV Name /dev/sdb2

VG Name

PV Size 250.00 MiB

Allocatable NO

PE Size 0

Total PE 0

Free PE 0

Allocated PE 0

PV UUID pXv062-nAbB-suQV-2oq9-55nc-lqNH-XgO153

"/dev/sdb1" is a new physical volume of "250.00 MiB"

--- NEW Physical volume ---

PV Name /dev/sdb1

VG Name

PV Size 250.00 MiB

Allocatable NO

PE Size 0

Total PE 0

Free PE 0

Allocated PE 0

PV UUID UNJXcQ-QnGj-4FZq-oORr-PzDu-LiHu-f1WgI0

1. Create a volume group named myvg and add the two physical volumes to it. Use the default extent size.

alexandru@debian:~$ sudo vgcreate myvg /dev/sdb1 /dev/sdb2

Volume group "myvg" successfully created

alexandru@debian:~$ sudo vgdisplay

--- Volume group ---

VG Name debian-vg

System ID

Format lvm2

Metadata Areas 1

Metadata Sequence No 3

VG Access read/write

VG Status resizable

MAX LV 0

Cur LV 2

Open LV 2

Max PV 0

Cur PV 1

Act PV 1

VG Size 111.55 GiB

PE Size 4.00 MiB

Total PE 28556

Alloc PE / Size 28556 / 111.55 GiB

Free PE / Size 0 / 0

VG UUID INFr5q-OXoM-L1yI-btT4-YtHP-U99T-flIO87

--- Volume group ---

VG Name myvg

System ID

Format lvm2

Metadata Areas 2

Metadata Sequence No 1

VG Access read/write

VG Status resizable

MAX LV 0

Cur LV 0

Open LV 0

Max PV 0

Cur PV 2

Act PV 2

VG Size 496.00 MiB

PE Size 4.00 MiB

Total PE 124

Alloc PE / Size 0 / 0

Free PE / Size 124 / 496.00 MiB

VG UUID HTPpQr-C6zG-N8GK-V5kj-J8U4-KGdY-1uGF01

1. Allocate a 300 MB logical volume named mylvm from volume group myvg.

alexandru@debian:~$ sudo lvcreate -L 300M -n mylvm myvg

Logical volume "mylvm" created.

alexandru@debian:~$ sudo lvdisplay

--- Logical volume ---

LV Path /dev/debian-vg/root

LV Name root

VG Name debian-vg

LV UUID vsoJtP-fUXW-dyCU-Ofc6-LUMY-82he-UfYeZ5

LV Write Access read/write

LV Creation host, time debian, 2017-09-08 20:44:50 +0300

LV Status available

# open 1

LV Size 107.66 GiB

Current LE 27561

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:1

--- Logical volume ---

LV Path /dev/debian-vg/swap\_1

LV Name swap\_1

VG Name debian-vg

LV UUID Yi1QYt-coOu-bnqC-mXgA-WUPk-TMJc-JOzKrW

LV Write Access read/write

LV Creation host, time debian, 2017-09-08 20:44:50 +0300

LV Status available

# open 2

LV Size 3.89 GiB

Current LE 995

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:2

--- Logical volume ---

LV Path /dev/myvg/mylvm

LV Name mylvm

VG Name myvg

LV UUID zlEpKI-YJDV-LMtD-r1gQ-ZY54-KffL-A0Sl3W

LV Write Access read/write

LV Creation host, time debian, 2017-09-11 18:12:52 +0300

LV Status available

# open 0

LV Size 300.00 MiB

Current LE 75

Segments 2

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:3

1. Format and mount the logical volume mylvm at /mylvm

alexandru@debian:~$ sudo mkfs.ext4 /dev/myvg/mylvm

mke2fs 1.43.4 (31-Jan-2017)

Creating filesystem with 307200 1k blocks and 76912 inodes

Filesystem UUID: 5c4bccbb-2f6f-4ffa-8b43-f42f4f123295

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729, 204801, 221185

Allocating group tables: done

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information: done

alexandru@debian:~$ mkdir /mylvm

mkdir: cannot create directory ‘/mylvm’: Permission denied

alexandru@debian:~$ sudo mkdir /mylvm

alexandru@debian:~$ sudo mount /dev/myvg/mylvm /mylvm

alexandru@debian:~$

1. Use lvdisplay to view information about the logical volume

alexandru@debian:~$ sudo lvdisplay

--- Logical volume ---

LV Path /dev/debian-vg/root

LV Name root

VG Name debian-vg

LV UUID vsoJtP-fUXW-dyCU-Ofc6-LUMY-82he-UfYeZ5

LV Write Access read/write

LV Creation host, time debian, 2017-09-08 20:44:50 +0300

LV Status available

# open 1

LV Size 107.66 GiB

Current LE 27561

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:1

--- Logical volume ---

LV Path /dev/debian-vg/swap\_1

LV Name swap\_1

VG Name debian-vg

LV UUID Yi1QYt-coOu-bnqC-mXgA-WUPk-TMJc-JOzKrW

LV Write Access read/write

LV Creation host, time debian, 2017-09-08 20:44:50 +0300

LV Status available

# open 2

LV Size 3.89 GiB

Current LE 995

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:2

--- Logical volume ---

LV Path /dev/myvg/mylvm

LV Name mylvm

VG Name myvg

LV UUID zlEpKI-YJDV-LMtD-r1gQ-ZY54-KffL-A0Sl3W

LV Write Access read/write

LV Creation host, time debian, 2017-09-11 18:12:52 +0300

LV Status available

# open 1

LV Size 300.00 MiB

Current LE 75

Segments 2

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 254:3

1. Grow the logical volume and corresponding filesystem to 350MB

alexandru@debian:~$ df -h

Filesystem Size Used Avail Use% Mounted on

udev 1.9G 0 1.9G 0% /dev

tmpfs 385M 6.6M 378M 2% /run

/dev/mapper/debian--vg-root 106G 24G 77G 24% /

tmpfs 1.9G 0 1.9G 0% /dev/shm

tmpfs 5.0M 4.0K 5.0M 1% /run/lock

tmpfs 1.9G 0 1.9G 0% /sys/fs/cgroup

/dev/sda1 236M 38M 186M 17% /boot

tmpfs 385M 16K 384M 1% /run/user/117

tmpfs 385M 44K 384M 1% /run/user/1000

/dev/mapper/myvg-mylvm 283M 2.1M 262M 1% /mylvm

alexandru@debian:~$ sudo lvextend -L 350M /dev/myvg/mylvm

Rounding size to boundary between physical extents: 352.00 MiB.

Size of logical volume myvg/mylvm changed from 300.00 MiB (75 extents) to 352.00 MiB (88 extents).

Logical volume myvg/mylvm successfully resized.

alexandru@debian:~$ sudo resize2fs /dev/myvg/mylvm

resize2fs 1.43.4 (31-Jan-2017)

Filesystem at /dev/myvg/mylvm is mounted on /mylvm; on-line resizing required

old\_desc\_blocks = 3, new\_desc\_blocks = 3

The filesystem on /dev/myvg/mylvm is now 360448 (1k) blocks long.

alexandru@debian:~$ df -h

Filesystem Size Used Avail Use% Mounted on

udev 1.9G 0 1.9G 0% /dev

tmpfs 385M 6.6M 378M 2% /run

/dev/mapper/debian--vg-root 106G 24G 77G 24% /

tmpfs 1.9G 0 1.9G 0% /dev/shm

tmpfs 5.0M 4.0K 5.0M 1% /run/lock

tmpfs 1.9G 0 1.9G 0% /sys/fs/cgroup

/dev/sda1 236M 38M 186M 17% /boot

tmpfs 385M 16K 384M 1% /run/user/117

tmpfs 385M 44K 384M 1% /run/user/1000

/dev/mapper/myvg-mylvm 334M 2.1M 311M 1% /mylvm

## Laboratory objective 9: Creating a RAID device

Commands used: fdisk, mdadm, mkfs, mkdir, mount, cat

Solution:

We are using Debian 9.1 for this exercise. Normally, when creating a RAID device we would use partitions from separate disks, but for this exercise we will use two partitions on the same disk. The process is actually the same whether the partitions are on one drive or on several drives. In real life operations, there is no reason to create a RAID device with two partitions on the same disk.

1. Create two 1GB partitions of type "raid" either on your hard disk/memory stick using fdisk, or using LVM.

alexandru@debian:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.29.2).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Command (m for help): n

Partition type

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

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1):

First sector (2048-62530623, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-62530623, default 62530623): +1G

Created a new partition 1 of type 'Linux' and of size 1 GiB.

Partition #1 contains a ext4 signature.

Do you want to remove the signature? [Y]es/[N]o: Y

The signature will be removed by a write command.

Command (m for help): n

Partition type

p primary (1 primary, 0 extended, 3 free)

e extended (container for logical partitions)

Select (default p):

Using default response p.

Partition number (2-4, default 2):

First sector (2099200-62530623, default 2099200):

Last sector, +sectors or +size{K,M,G,T,P} (2099200-62530623, default 62530623): +1G

Created a new partition 2 of type 'Linux' and of size 1 GiB.

Command (m for help): t

Partition number (1,2, default 2): 1

Partition type (type L to list all types): L

0 Empty 24 NEC DOS 81 Minix / old Lin bf Solaris

1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 DRDOS/sec (FAT-

2 XENIX root 39 Plan 9 83 Linux c4 DRDOS/sec (FAT-

3 XENIX usr 3c PartitionMagic 84 OS/2 hidden or c6 DRDOS/sec (FAT-

4 FAT16 <32M 40 Venix 80286 85 Linux extended c7 Syrinx

5 Extended 41 PPC PReP Boot 86 NTFS volume set da Non-FS data

6 FAT16 42 SFS 87 NTFS volume set db CP/M / CTOS / .

7 HPFS/NTFS/exFAT 4d QNX4.x 88 Linux plaintext de Dell Utility

8 AIX 4e QNX4.x 2nd part 8e Linux LVM df BootIt

9 AIX bootable 4f QNX4.x 3rd part 93 Amoeba e1 DOS access

a OS/2 Boot Manag 50 OnTrack DM 94 Amoeba BBT e3 DOS R/O

b W95 FAT32 51 OnTrack DM6 Aux 9f BSD/OS e4 SpeedStor

c W95 FAT32 (LBA) 52 CP/M a0 IBM Thinkpad hi ea Rufus alignment

e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a5 FreeBSD eb BeOS fs

f W95 Ext'd (LBA) 54 OnTrackDM6 a6 OpenBSD ee GPT

10 OPUS 55 EZ-Drive a7 NeXTSTEP ef EFI (FAT-12/16/

11 Hidden FAT12 56 Golden Bow a8 Darwin UFS f0 Linux/PA-RISC b

12 Compaq diagnost 5c Priam Edisk a9 NetBSD f1 SpeedStor

14 Hidden FAT16 <3 61 SpeedStor ab Darwin boot f4 SpeedStor

16 Hidden FAT16 63 GNU HURD or Sys af HFS / HFS+ f2 DOS secondary

17 Hidden HPFS/NTF 64 Novell Netware b7 BSDI fs fb VMware VMFS

18 AST SmartSleep 65 Novell Netware b8 BSDI swap fc VMware VMKCORE

1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fd Linux raid auto

1c Hidden W95 FAT3 75 PC/IX bc Acronis FAT32 L fe LANstep

1e Hidden W95 FAT1 80 Old Minix be Solaris boot ff BBT

Partition type (type L to list all types): fd

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

Command (m for help): t

Partition number (1,2, default 2): 2

Partition type (type L to list all types): fd

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

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

alexandru@debian:~$ sudo partprobe

1. Create RAID 1 device named /dev/md0 using the two partitions.

first you should run the command:

sudo apt install -y mdadm

to install the applications needed to create RAID. Then do:

alexandru@debian:~$ sudo mdadm -C /dev/md0 --level=1 --raid-disks=2 /dev/sdb1 /dev/sdb2

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?

Continue creating array? (y/n) y

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md0 started.

1. Format the RAID device as an ext4 filesystem. Then mount it at /myraid and make the mount persistent.

alexandru@debian:~$ sudo mkfs.ext4 /dev/md0

mke2fs 1.43.4 (31-Jan-2017)

Creating filesystem with 261888 4k blocks and 65536 inodes

Filesystem UUID: 3c4bfdae-9580-4876-8b3a-c628d35a29c7

Superblock backups stored on blocks:

32768, 98304, 163840, 229376

Allocating group tables: done

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

alexandru@debian:~$ sudo mkdir /myraid

alexandru@debian:~$ sudo mount /dev/md0 /myraid

Add the following line in /etc/fstab

/dev/md0 /myraid ext4 defaults 0 0

1. Place the information about /dev/md0 in /etc/mdadm.conf file using mdadm.

alexandru@debian:~$ sudo mdadm --detail --scan >> /etc/mdadm.conf

bash: /etc/mdadm.conf: Permission denied

alexandru@debian:~$ sudo su

root@debian:/home/alexandru# mdadm --detail --scan >> /etc/mdadm.conf

1. Examine /proc/mdstat to see the status of your RAID device.

root@debian:/home/alexandru# cat /proc/mdstat

Personalities : [raid1]

md0 : active raid1 sdb2[1] sdb1[0]

1047552 blocks super 1.2 [2/2] [UU]

unused devices: <none>

You can also verify that everything that is working great by doing a reboot. To clean everything up, you can erase that line from /etc/fstab file, and then remove the partitions.

alexandru@debian:~$ sudo umount /dev/md0

alexandru@debian:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.29.2).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Command (m for help): d

Partition number (1,2, default 2): 1

Partition 1 has been deleted.

Command (m for help): d

Selected partition 2

Partition 2 has been deleted.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8).

alexandru@debian:~$ sudo partprobe

Error: Partition(s) 1, 2 on /dev/sdb have been written, but we have been unable to inform the kernel of the change, probably because it/they are in use. As a result, the old partition(s) will remain in use. You should reboot now before making further changes.

alexandru@debian:~$