Lab 2:

Partitioning

Forensic Analysis 2023-2024

© Daan Pareit

## Introduction

# Lab Concept

During this lab, we will be using some tools to create & manipulate partitions. We will work on a VM which you can consider as your own machine on which you want to change partitions and recover from partition errors. This is done so you gain more understanding about partitioning and to change/repair partitioning in your professional life.

This time, we will thus not consider the VM as a suspect’s computer here, as in the next labs. In this lab you first need to get acquainted with partitioning logic. The tools used in this lab can of course also be used to analyze an acquired disk image, rather than on a disk in use.

# Learning goals

Knowledge

* Content of the MBR and EBR
* Primary versus extended and logical partitions

Skills

* Change partitioning with fdisk: creation and deletion of primary/extended/logical partitions
* Use other partitioning tools: parted, gparted and mmls (from the sleuthkit suite)
* Restore damaged partition tables

Insight

* Analyse the drive structure

# Practicalities

### Get the images and software

We will work on a Debian Virtual Machine (VM) which is preconfigured. This is a zip file which you can unzip and import in VMWare Workstation/Fusion, typically by (double) clicking the vmx file. You’ll also need following ISOs, which we will use as virtual CDs to boot from at some point in this lab:

* Debian install CD
* Live GParted CD

You can find the files at:

* the samba share: \\nas.ti.howest.be\TI-StudentShare\TI-S4-Forensics ,
* or via HTTP at <https://nas.ti.howest.be> in the TI-S4-Forensics folder.

### Running the VM

You can login at the VM as ‘user’ with password ‘user’. Furthermore, ‘root’ has password ‘toor’. However -as always- avoid logging in as root, use the regular user and sudo instead.

Some hints for easy usage:

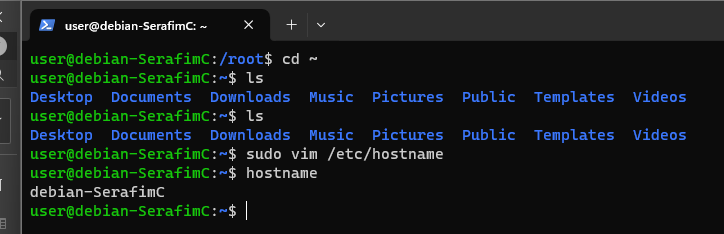
* Hint 1: There is an SSH server running, thus you can also login with SSH from your laptop into the VM, which is often handier (e.g. for copy/pasting or 4K laptop screens).
* Hint 2: changing font size on a uxterm in the xfce desktop is done via Ctrl + holding right mouse button and selecting the desired font size

## Lab assignments

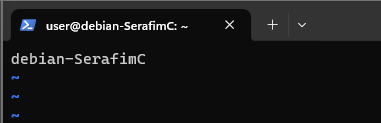
# Personalize the VM

To verify you performed the lab yourself, you need to change the hostname in the /etc/hostname file. Change the name to include your firstname and the first letter of your lastname. E.g. this would be hostname debian-DaanP if your name is Daan Pareit. This is done by performing all following steps

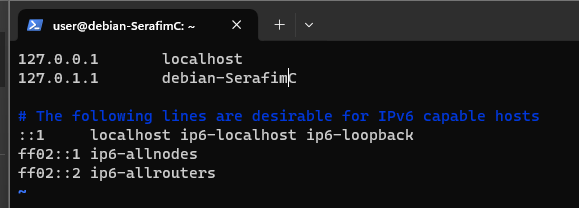
* Execute the command: hostname debian-DaanP



* Change the content of the /etc/hostname file to debian-DaanP



* Change the ‘debian’ text in /etc/hosts for 127.0.1.1 to ‘debian-DaanP’



* Logout and login again

reboot or do su —login

Now, all your prompts (in tty, xfce uxterm, ssh etc) should include your name. When screenshots are asked for, ensure that your name is visible in the prompt (if not, it will be considered void).

# ANALYSIS

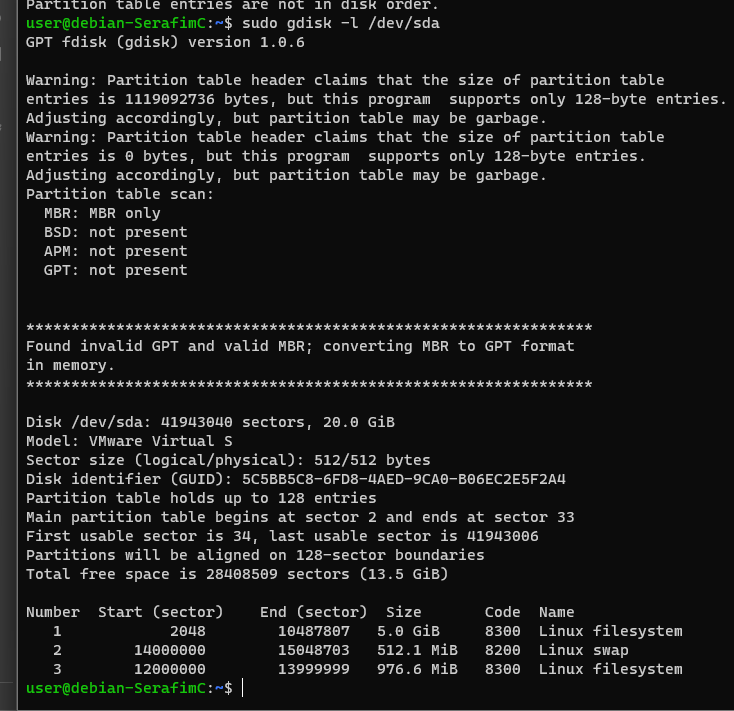
### Analyze the current partitions

In this section, we will make sure to take the time to obtain a good understanding of how the partition table works, where it is located etc, before changing the partitions in the next sections.

As you know from previous lab, **/dev/sda** is the representation of our SCSI hard disk (a block device) in Linux. **/dev/sda1, /dev/sda2**, etc are representations of partitions on that disk. These are considered as ‘**block devices**’ in Linux: a continuous series of bits, ordered in blocks/sectors, representing the hard disk. You’ll learn more about Linux’ view on ‘block devices’ later in this course.

For basic partitioning operation in Linux, we use the fdisk tool for disks with the **MBR (Master Boot Record)** partition style or the gdisk tool for disks with the **GPT (GUID partition table)** partition style. Note: there also other partition styles, such as APM (Apple Partition Map) and BSD (Berkeley Software Distribution)

* Run sudo gdisk -l /dev/sda to know whether the disk has MBR or GPT partition style.



It says it is a MBR

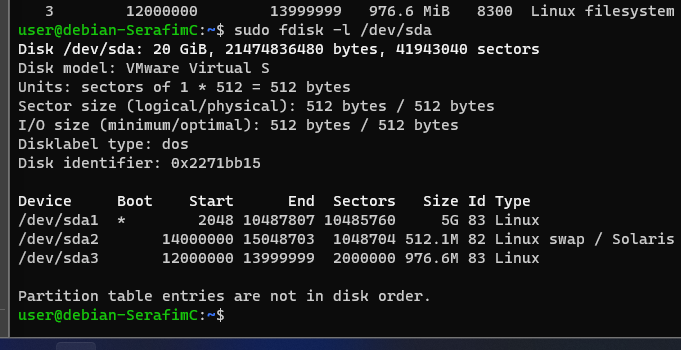
Using fdisk, included in about every linux distro, you can manipulate the partition table of a single MBR partitioned disk. The command offers two much used options:

sudo fdisk –l /dev/sda # displays the partition table of the hard disk

sudo fdisk /dev/sda # opens a CLI wizard to manipulate the partition

(fdisk was named that way originally for ‘fixed disk’. Some argue it stands for ‘f\*\*\*ing disk’, especially when you write the partition on the wrong disk ).

* Use fdisk to display the partition table of the hard disk of the VM



* Are the partitions closely following each other? Hint: look at the last line of fdisk output.

They are not in disk order

* What is the actual order of the partitions on the disk: which one comes first on the disk, middle and last? Hint: Look at the start and end sectors.

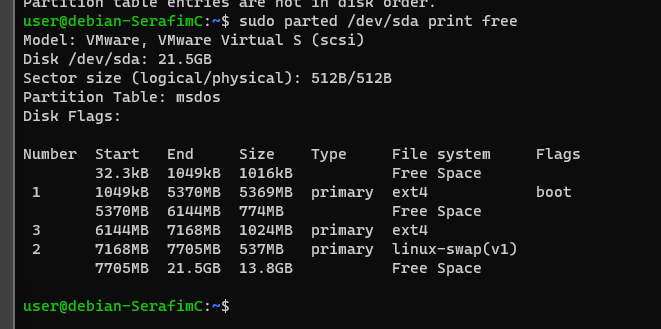
**/dev/sda1**, then **/dev/sda3** and then **/dev/sda2**

fdisk is a very basic tool, a more elaborated tool is parted, which also displays the free space available.

sudo parted /dev/sda print free

# lists the partitions and the unallocated space

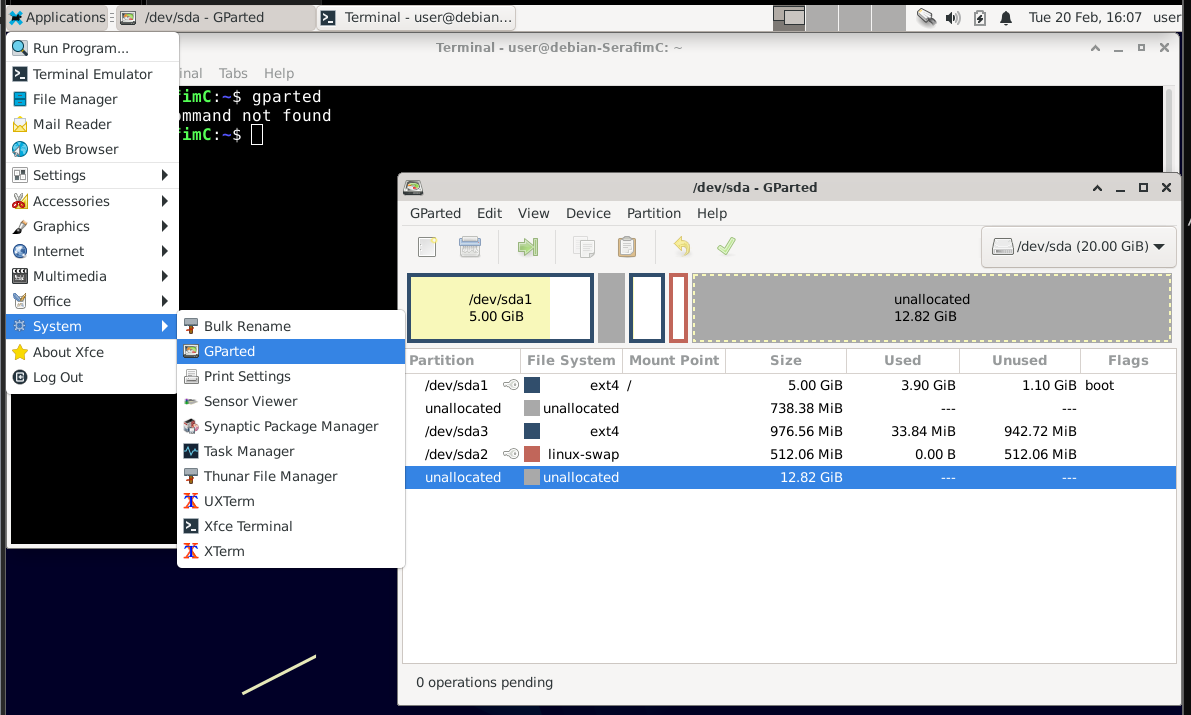
* Execute the parted command above. You can now better see the order of the partitions on the disk and the unallocated space in between. How much free space is there at the end?



Seems like it is 13.8GB of free space

We can also use the graphical tool gparted, which visualizes this partition scheme.

* Run the gparted tool.



Let’s take it one step further and understand the partition tables in which each partition is described. To this end we use the mmls tool (Media Manager Listing) of the sleuthkit suite. We’ll use the various tools of the sleuthkit (TSK) suite quite often in this course.

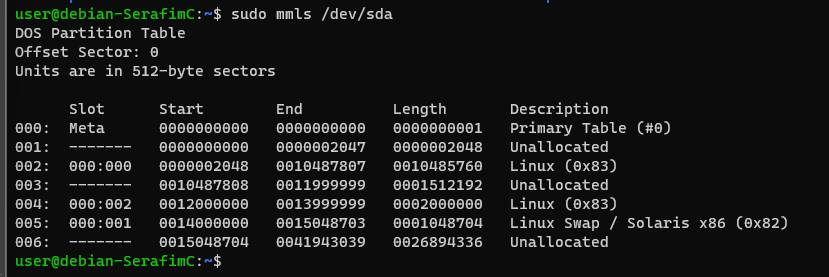
sudo mmls /dev/sda

# lists the partitions and the partition table entries

* Install the sleuthkit package (using apt)

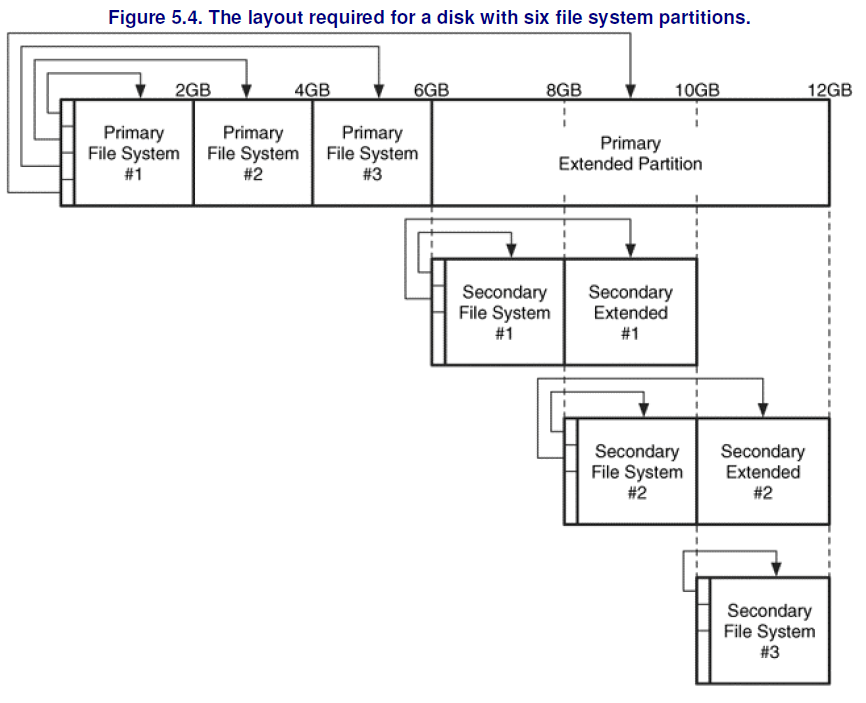
sudo apt install sleuthkit

* Execute the mmls command above.



In the ‘slot’ column, you’ll see something of this format 000:001 . The first number (eg 000 in 000:001) shows which table is used (cfr theory slides):

* 000 is the primary partition table, located at the MBR
* 001 would be the second partition table, located at the beginning of a primary extended partition,
* 002 would be the third partition table, located at the beginning of a secondary extended partition.
* 003 would be the fourth partition table, located at the beginning of a secondary extended partition.
* Etc. See the figure below (from the File System Forensic Analysis ebook) and theory slides



The second number, (eg 001 in 000:001) indicates the entry in the partition table. In MBR disks, there are maximum 4 entries in a partition table.

* 000 is the first entry in the partion table
* 001 is the second entry in the partition table
* 002 is the third entry in the partion table
* 003 is the fourth (and last) entry in the partion table

Now, back to our VM and the output of mmls:

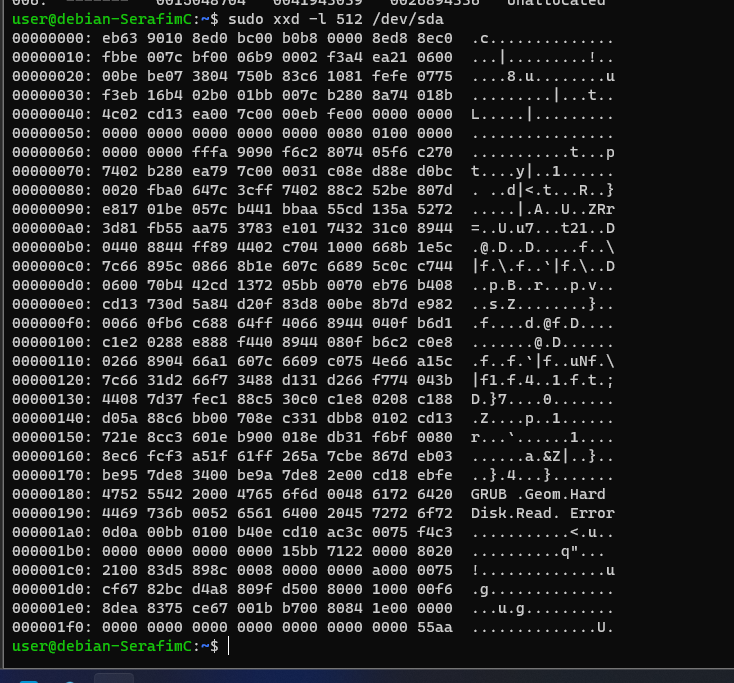
* Are all the partitions included in the primary partition table (the one in the MBR)?

Apparently they are, as all of them start with 000

Thus all the tools you used so far, are actually just different ways of visualizing the partition table inside the MBR. The MBR itself has a size of 512 bytes and is (typically) located at the beginning of the hard disk. We can use a hex editor, such as xxd, to view these bytes:

sudo xxd -l 512 /dev/sda # hex view of first 512 bytes (= the MBR)

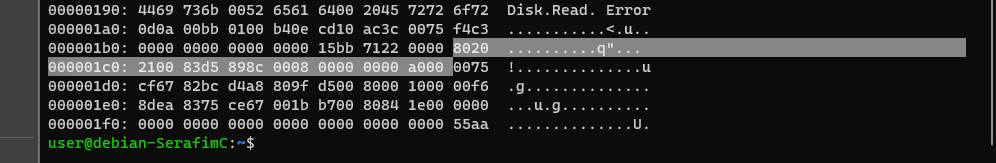
* Run the command above to visualize the MBR.



* The first bytes of the MBR are boot loader code. The last bytes are the partition table and the boot signature. What is the size of the partition table inside the MBR? Have a look at the theory slides. What is the value of the first byte in this partition table?

The size of the partition table inside the MBR is genuiely 4\*16 bytes, or 8 spaces of 4 hexes (since you can have 4 partition entrties). In our case, we have only 3 partitions, hence the last 16 bytes are missing (before the 55aa).

The first byte (meaning partition 1, consisting of 16 bytes as in the screenshot) seems to be **80**

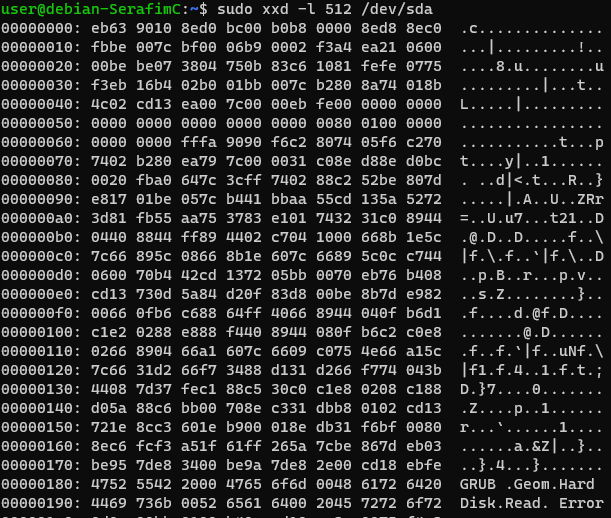


* What are the last 2 bytes of the MBR? This is the boot signature.

The last 2 bytes are the **55aa, and they are situated in the right bottom corner**

* Have a look at the ASCII representation of the boot loader code inside the MBR. What boot loader do you recognize here?

GRUB is our bootloader. Fisrt you can understand that while booting, but in the MBR code aswell.



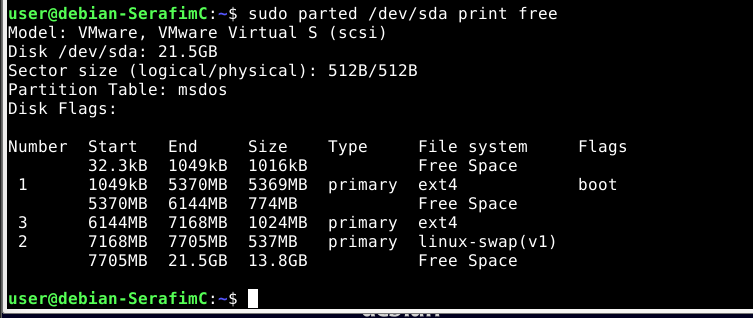
# CREATION

### creating a new Primary partition

We will create new partitions to learn working with fdisk.

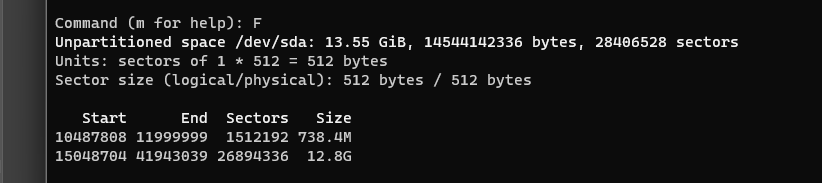
First, we want to create 1 extra primary partition of 6 GiB. To this end, follow the following steps:

* Beware: you will need to have a good look at how the partition table is structured until now. Where can you fit in a 6GiB partition? Before/after which current partition? Rerun the parted tool to find out where to create this new partition.



You can make it after 7705MB, as the start is there.

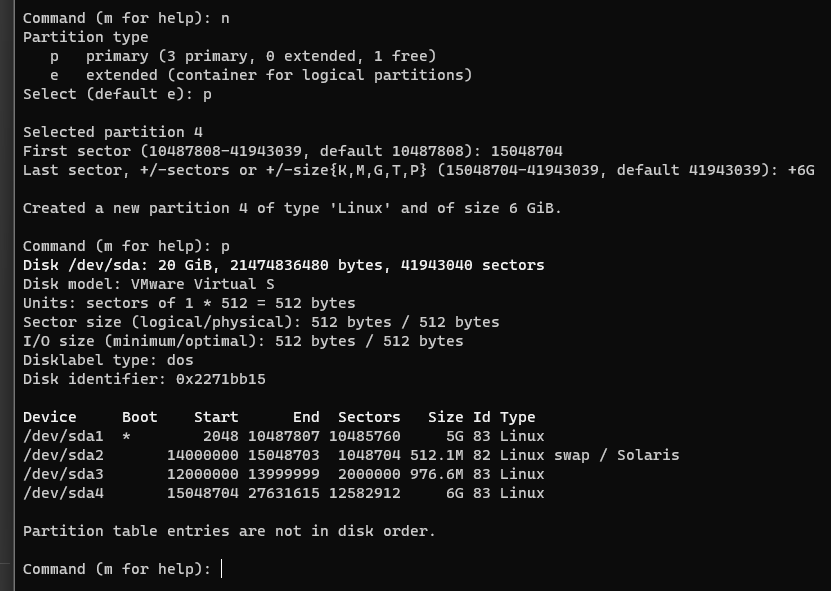
* Run the fdisk tool to find out what the first sector of the new partition will be, to create the 6GiB partition immediately after the partition of the previous question.



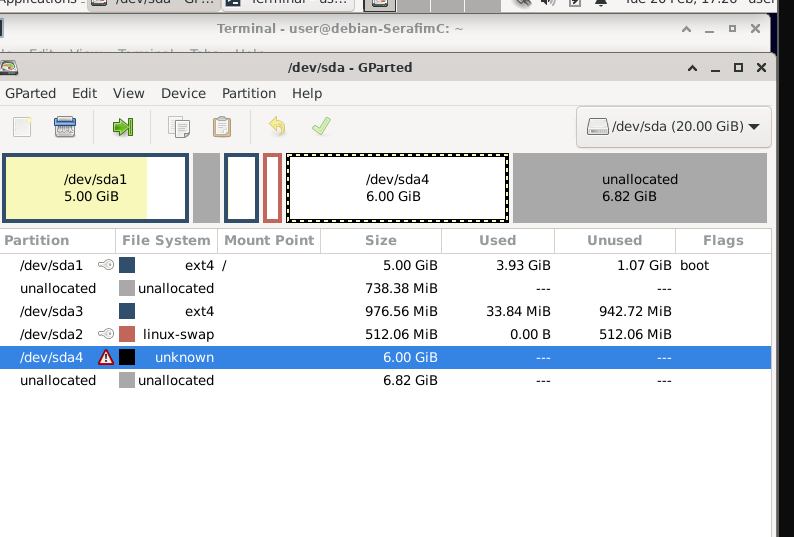
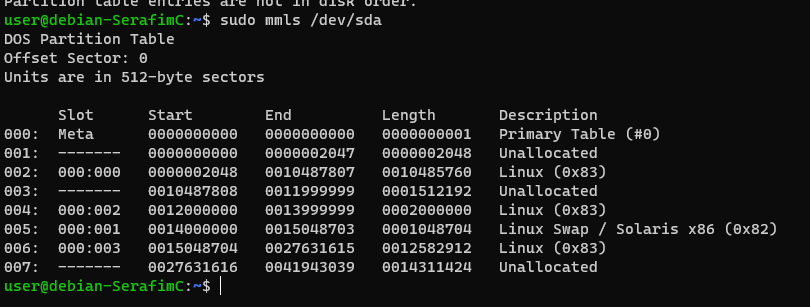
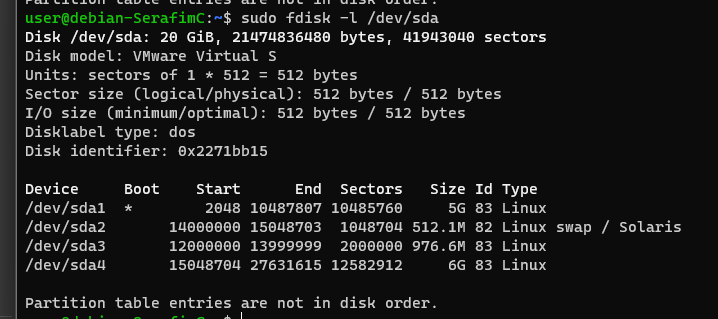
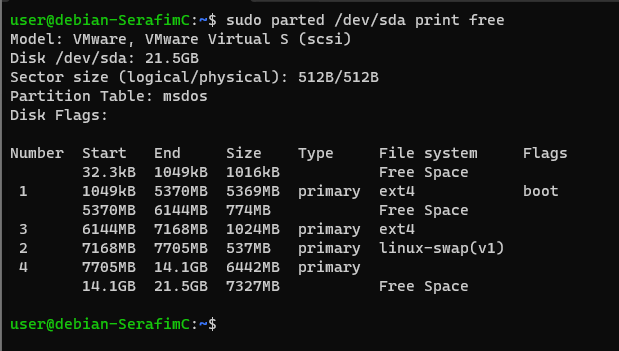
It will start at **15048704, as the partition availability is shown right here**

Parted shows you 2 different sectors of file types or something. As to create the partition afterwards

* Use the fdisk tool (in interactive mode) to create the new 6GiB partition. Use the ‘m’ command for help within fdisk. Create a new **primary (p)** partition. You already identified the first sector in the previous question. For the last sector, you could calculate the number of the last sector manually or use the easy way: use the “+” syntax, e.g.: “+100M” means 100 MiB further.



* Don’t forget to tell fdisk to write your newly proposed partition effectively also into the partition table. (Use ‘m’ for help.)
* After writing this change, you’ll read “Syncing disks.” which involves the running kernel to be aware of the changes.
  + If this fails, you’ll get a warning about re-reading the partition table which failed. In that case, run the following (as suggested by fdisk): partprobe -s /dev/sda
* Check your new disk layout with fdisk, parted, mmls and gparted after the successful creation of this 6GiB partition. Can you identify the new partition in the different tools?



GParted does not show the type of the partition, but over utilities – do. That might be a reboot issue, because it should work.

### Creating extended (logical) partitions

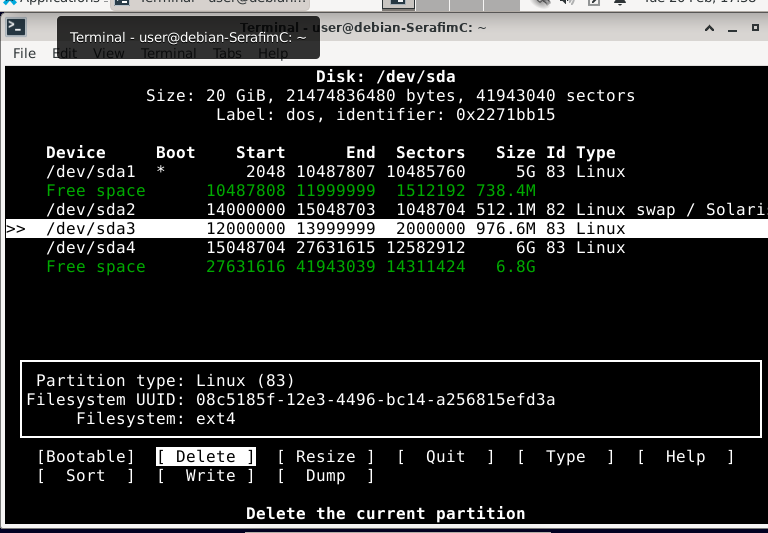
Now, we want to create two **logical** partitions in the free space at the end of the disk drive:

* 1 logical partition, sized 2 GiB
* 1 logical partition, sized 800 MiB

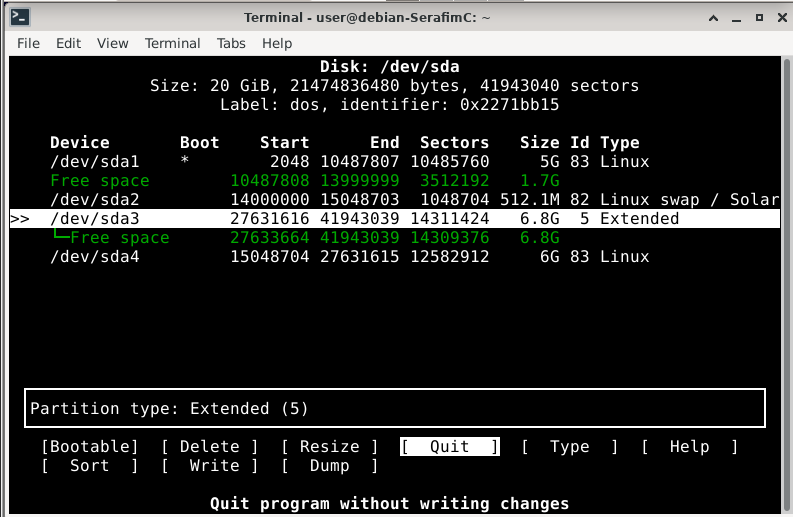
Follow these steps to do so:

* Are you able to create a new partition with fdisk? Why is this (not) possible? If not: there is a 1 GB partition currently not in use. You can erase this one … You can delete it via fdisk or you could also try the cfdisk tool instead, which has a little friendlier user interface.

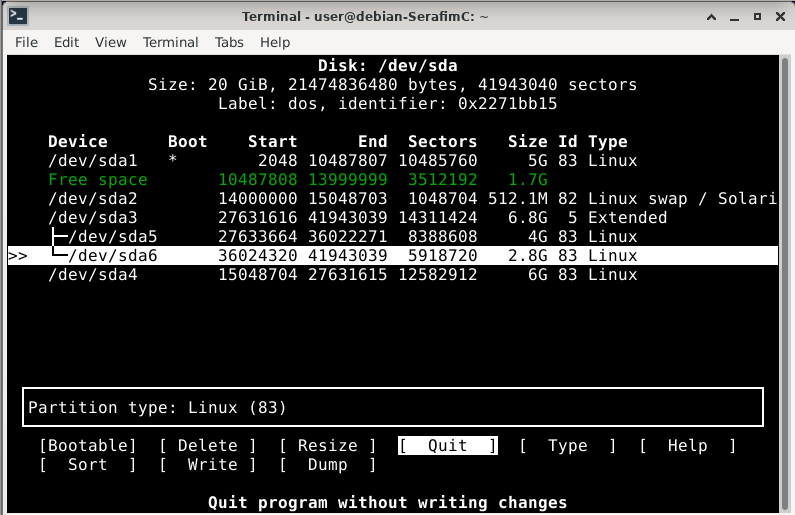
sudo cfdisk /dev/sda



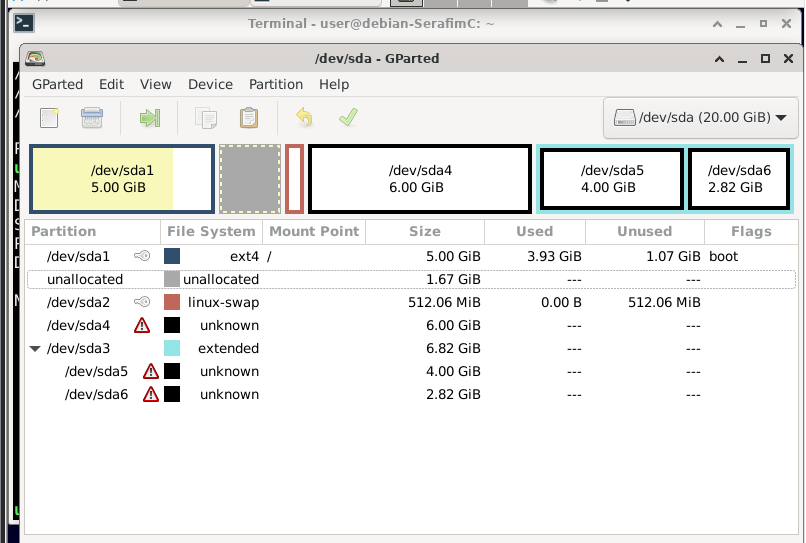
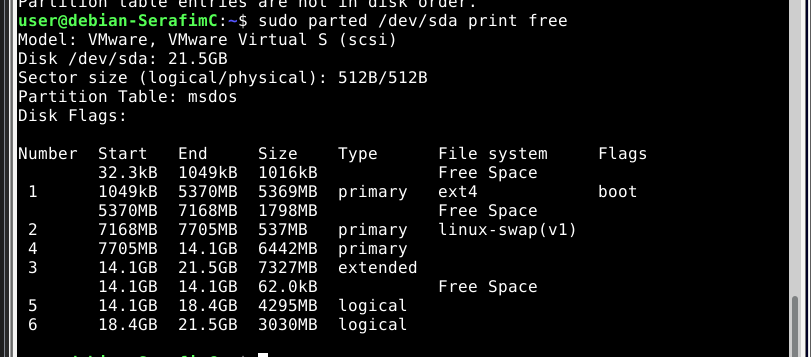
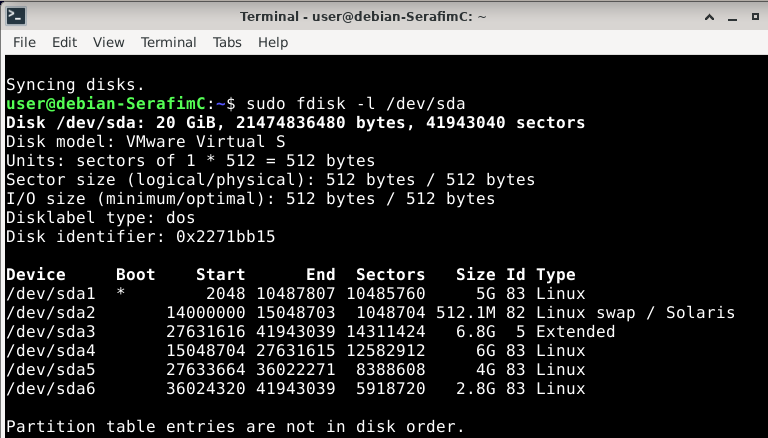
* Create an **extended partition** after the last partition on your disk (i.e. after your previously created 6GiB partition), which uses all free space at the end of the disk. Hint: Use the correct start sector!



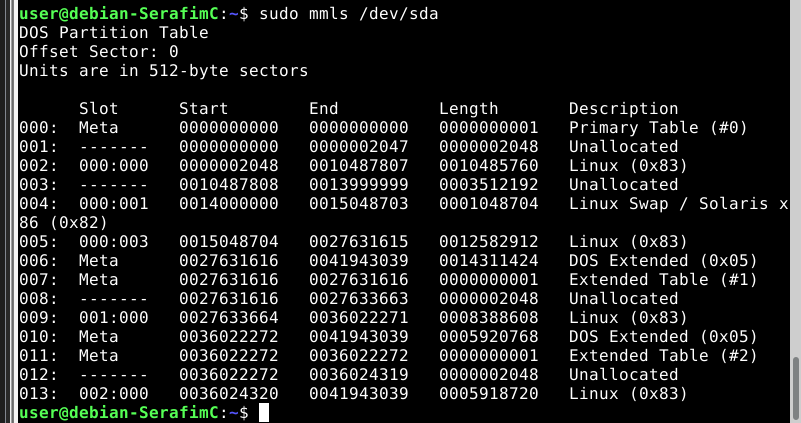
* Now, **within** this extended partition, create a **logical partition** of 4 GiB (you can use the proposed default start sector and use the “+4G” for the last sector in fdisk) **and another logical partition** which takes the rest of the extended partition.



* Have a look at your current disk layout with the fdisk, parted and gparted tools. (The warning signs for the new partitions mean they aren’t formatted with a file system yet.)



* **How many partition tables are currently present on your system?** Remember that logical partitions are using partition tables in extended boot records (EBR) as a kind of linked list, see theory slides. Use the mmls tool to gain insight and you’ll know the answer…



According to the information from the slides and what I can see here, I can say that we have one main partition table (Primary Table #0), and then I can see 2 more partition table (so in sum 3 partition tables), labeled Extrended Table #2 and #3. The slides say:

“This extended partition then has own partition table to includes multiple ‘logical’ partitions”

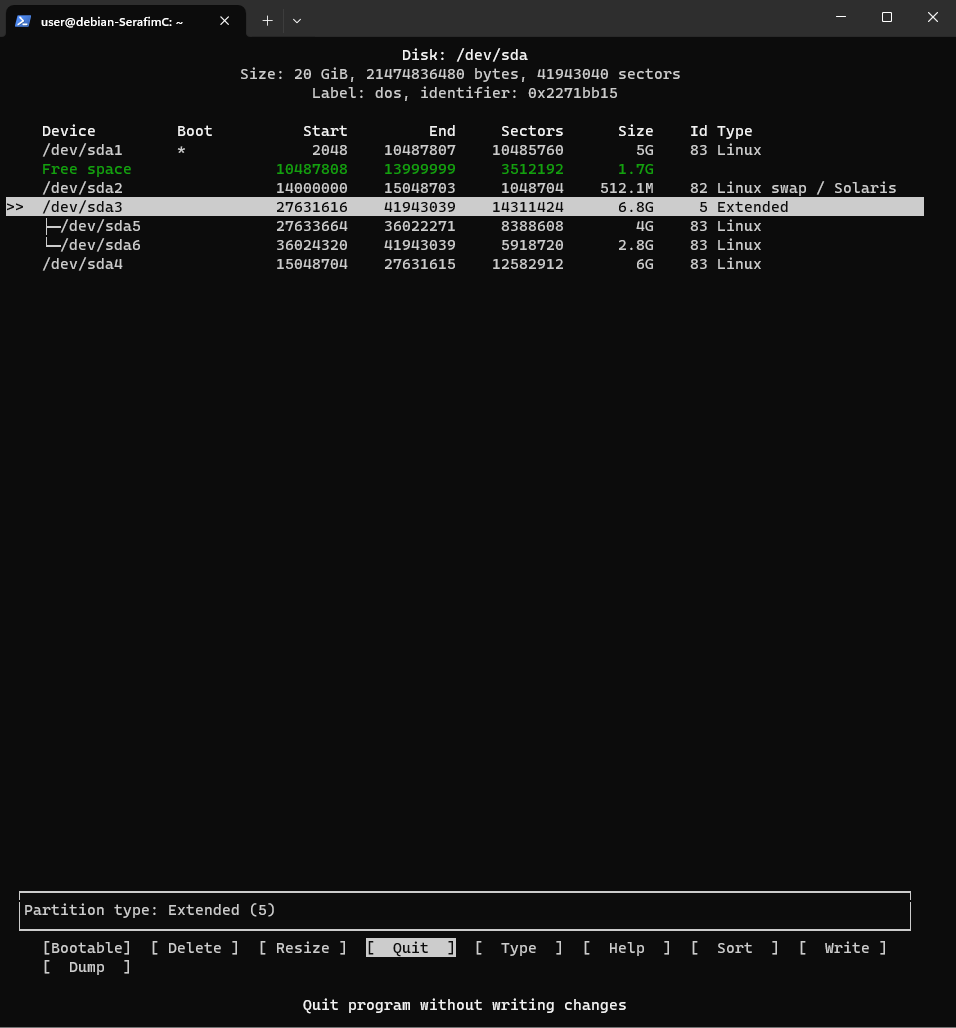
“Extended partition:

special kind of primary partition that cannot be bootable or hold data, but contains a link to a partition with its own partition table/boot record”

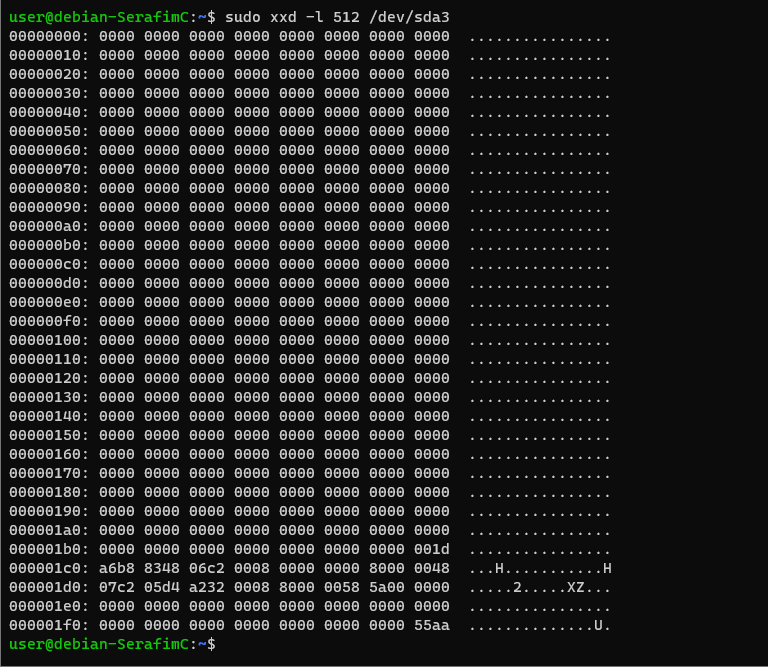
Hence, I consider this to be **3 partition tables.**

* The second partition table lays in an EBR at the beginning of our extended partition.
  + Which sdaX partition is this?

**/dev/sda3, is the extended partition**



* + Use xxd to inspect its first sector (= the EBR). It has the same structure as the MBR with the partition table at the end. What are the last two bytes, do you recognize these?



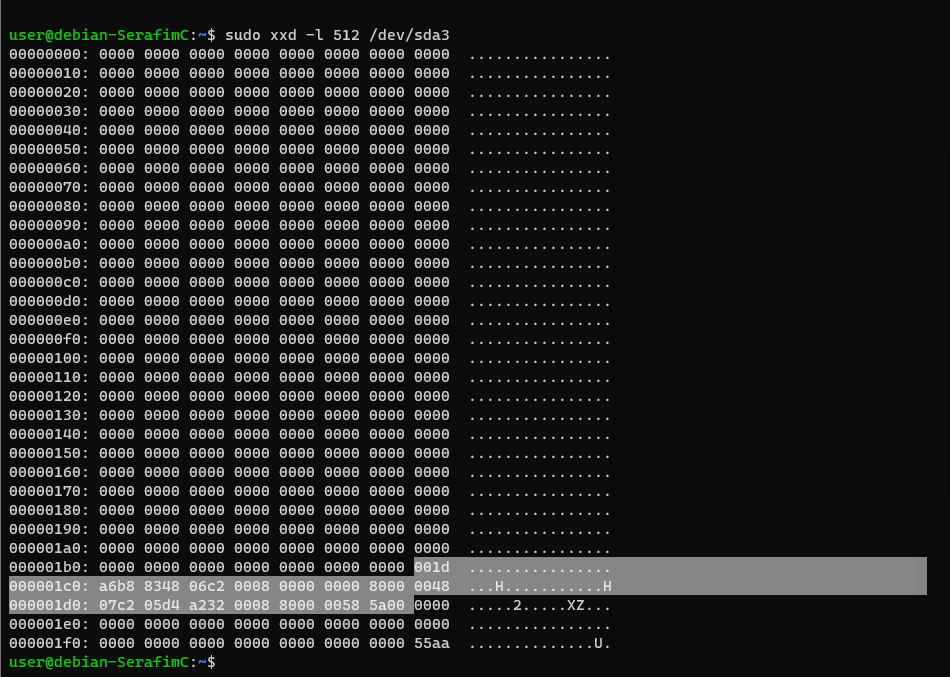
The last two bytes are **55aa, which is the same as in MBR**

* + Is there any boot code present in the first bytes?

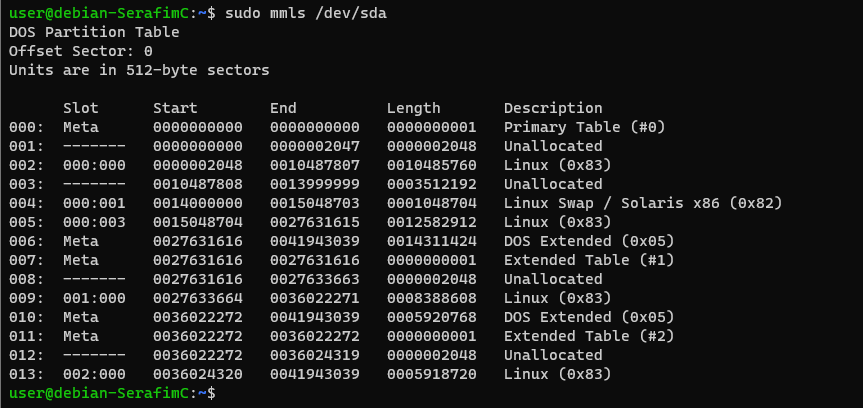
None, because it is not a bootable partition, and it should act like a routing mechanism

* + How many entries of the partition table in the EBR are in use?

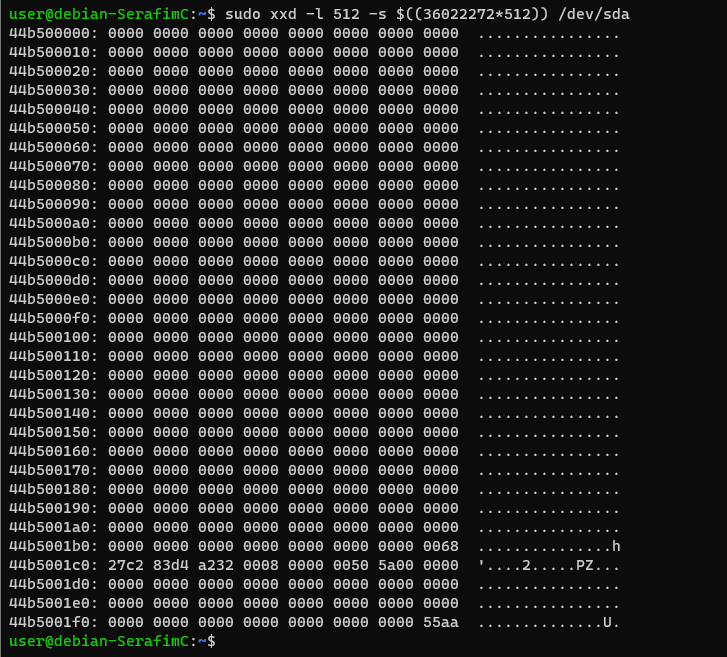
Only 2 entries are present and in use



* To see the second EBR which includes the third partition table, we can't inspect the first sector of a sdaX partition, that trick only worked for the 'extended partition'. To see this EBR, we need to specify explicitly which sector we want to examine. You can use the -s argument of xxd to ‘skip’ a number of bytes from the start of /dev/sda .
  + How many sectors should you skip (cfr mmls output)? Thus, how many bytes?



sudo xxd –l 512 –s $((36022272\*512)) /dev/sda (\*512 so we skip bytes and not simply sectors or something)

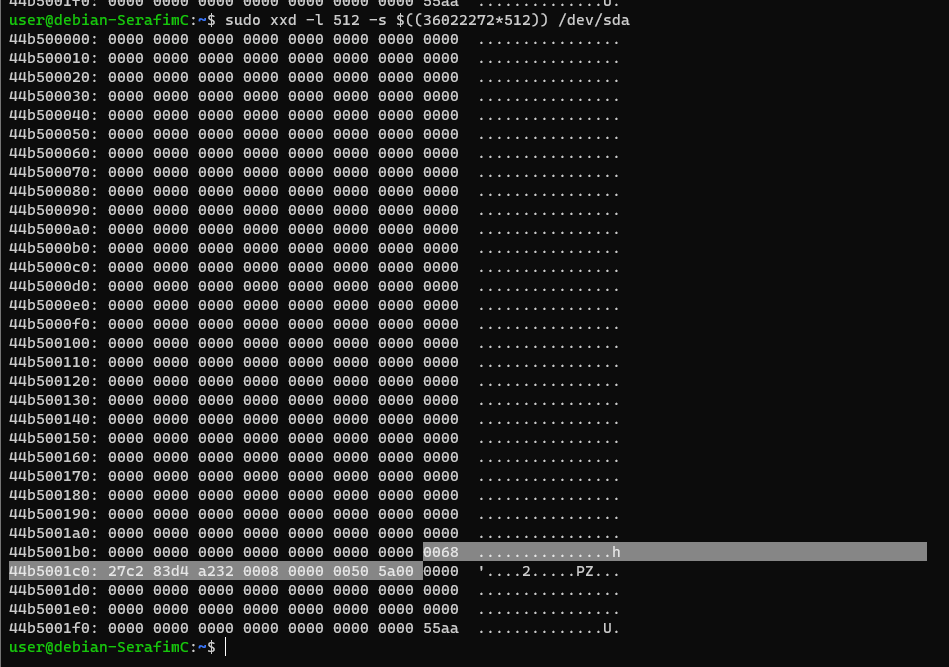


* + Is there any boot code present in the first bytes?\

Again, this is not bootable, so there is not.

* + How many entries of the partition table in the EBR are in use?

One partition is in use



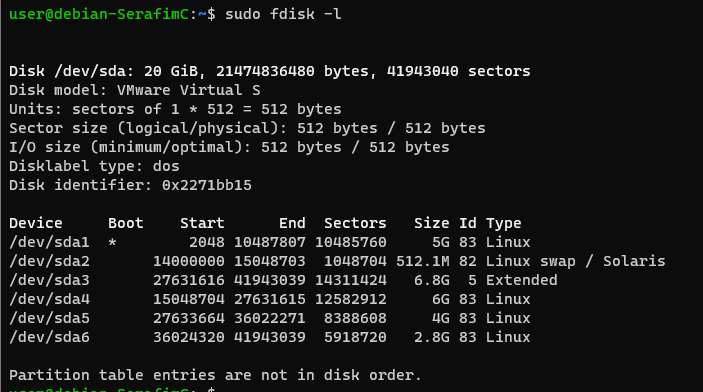
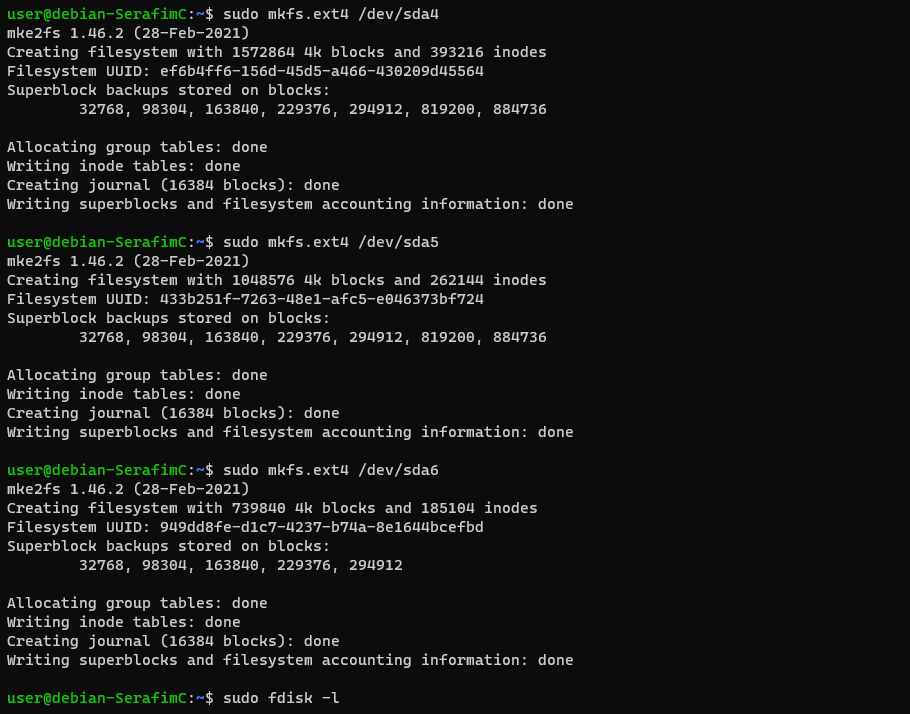
# Formatting the new partitions

Partitions, created on the disk, are created as empty space (allocated to a partition, but empty). They don’t have a file system yet. Formatting – the basic structures within the allocated space – still needs to be done. You’ll learn more about file systems in the next lessons and labs. You can format a partition with the ext4 file system as follows:

sudo mkfs.ext4 /dev/*<name of the partition>*

* Format all three new partitions (the primary one of 6GiB and the logical ones of 4GiB and the last one of about 2.8 GiB) with the ext4 file system. Be careful to format the right partitions. If all was done correctly, these should be sda4, sda5 and sda6

sudo mkfs.ext4 /dev/sda4 5 6



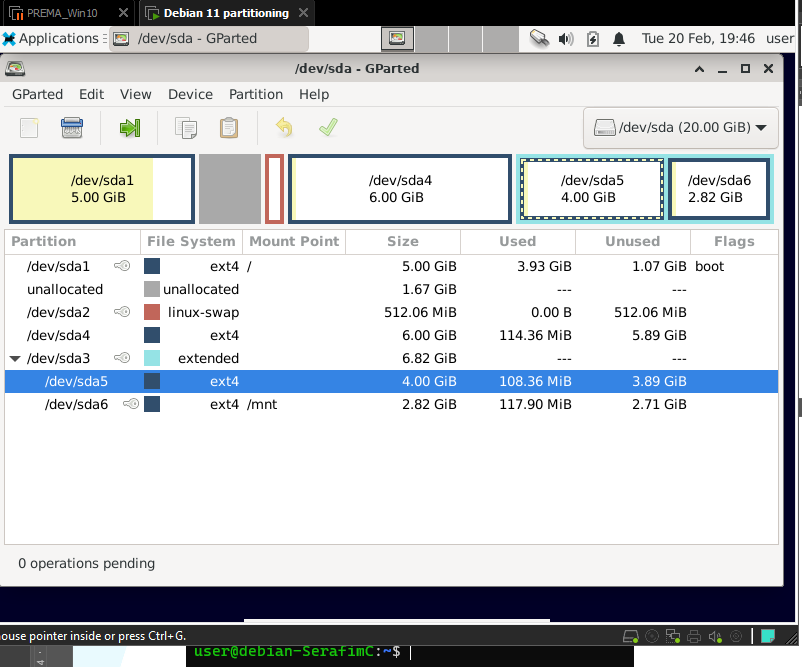
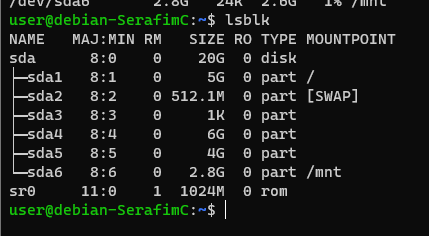
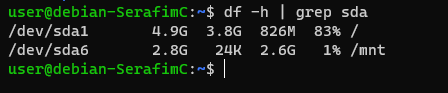
To use a formatted partition, you need to mount it.

* Mount the last partition to the /mnt directory:

mount -t ext4 /dev/sda6 /mnt/

* With df you can see the disk usage on mounted partitions.

df –h | grep sda # disk usage of mounted sda partitions



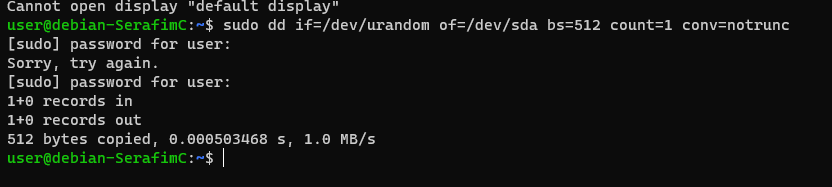
# ADVANCED

### destroy & recover the MBR

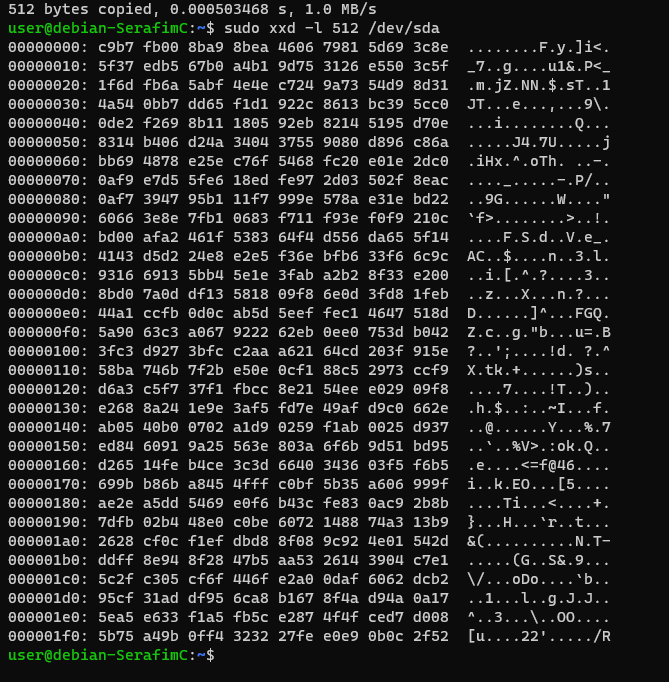
Suppose your MBR is wasted, how to recover? Using a tool as testdisk might help you to try and recover the partition table of the MBR. After the partition table has been restored, you can try to re-build the (GRUB) boot loader code of the MBR using a Debian install cd.

* First, take a snapshot of your VM!!!
* Waste your MBR by overwriting it with random data, using the following command

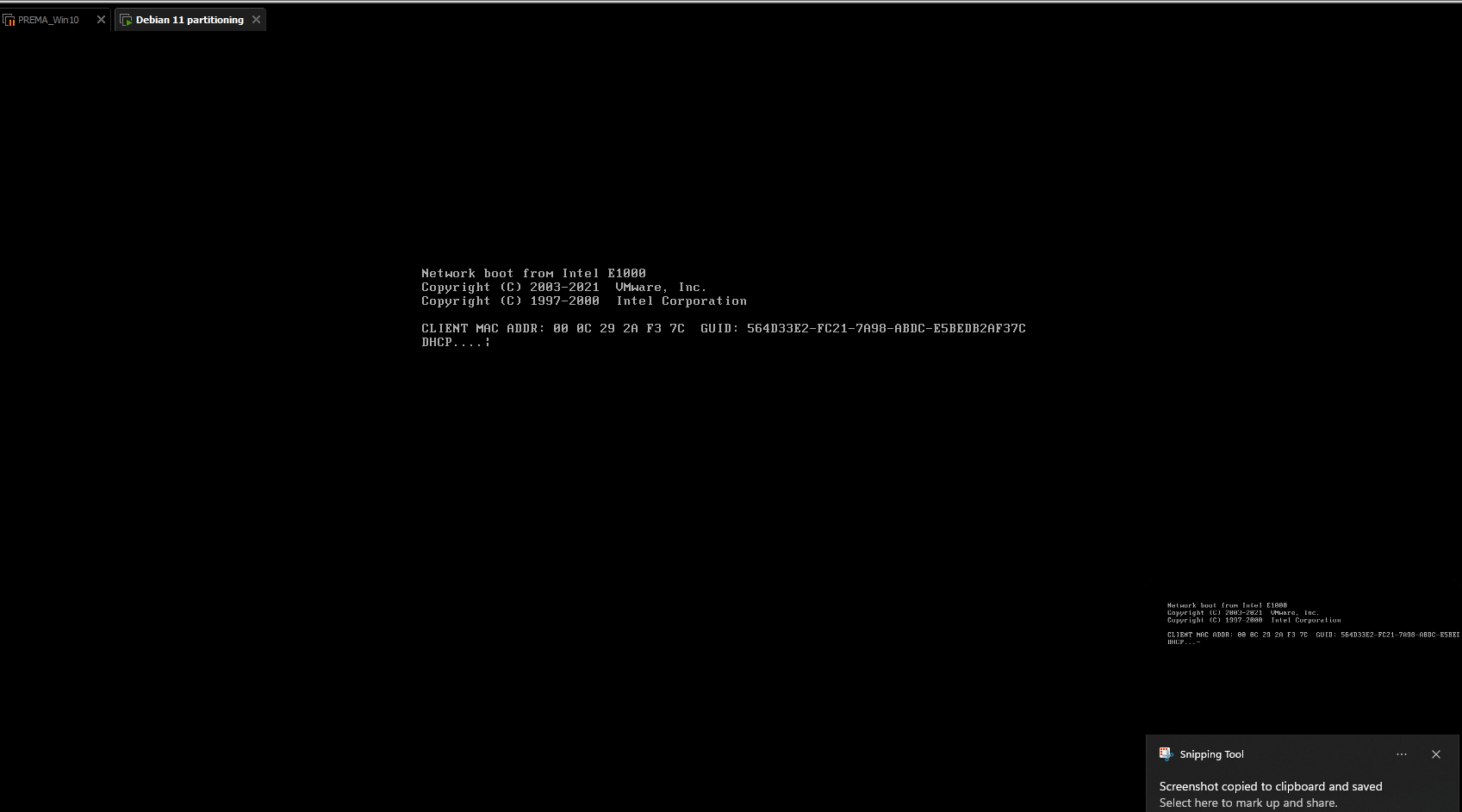
sudo dd if=/dev/urandom of=/dev/sda bs=512 count=1 conv=notrunc



* Verify with xxd that the MBR is randomized.

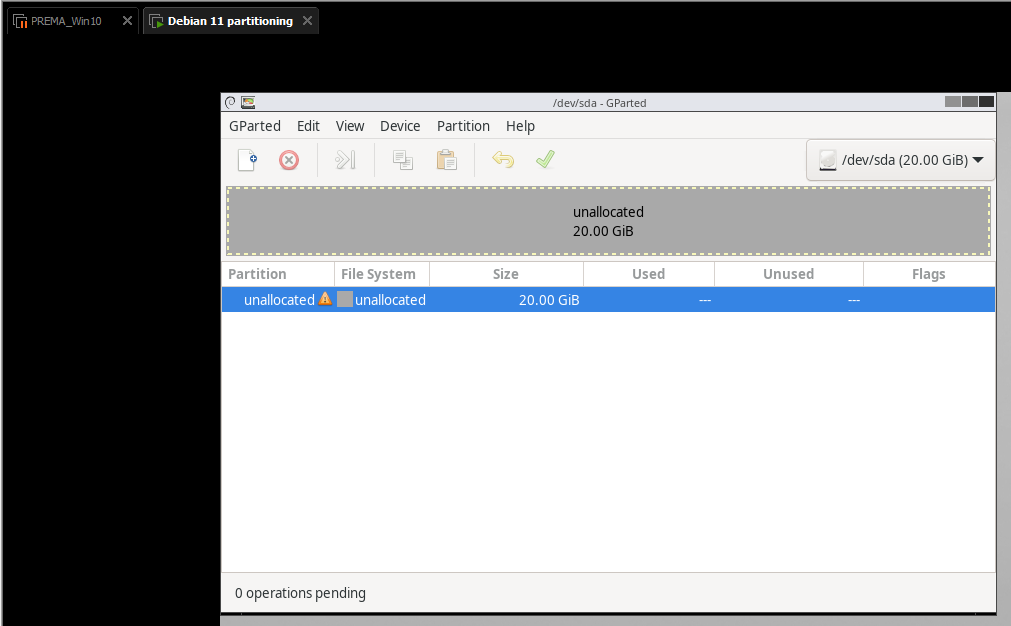


* Reboot and find your VM in trouble. Congratulations, you have just rendered your VM unbootable (by writing just 512 bytes of random data ).

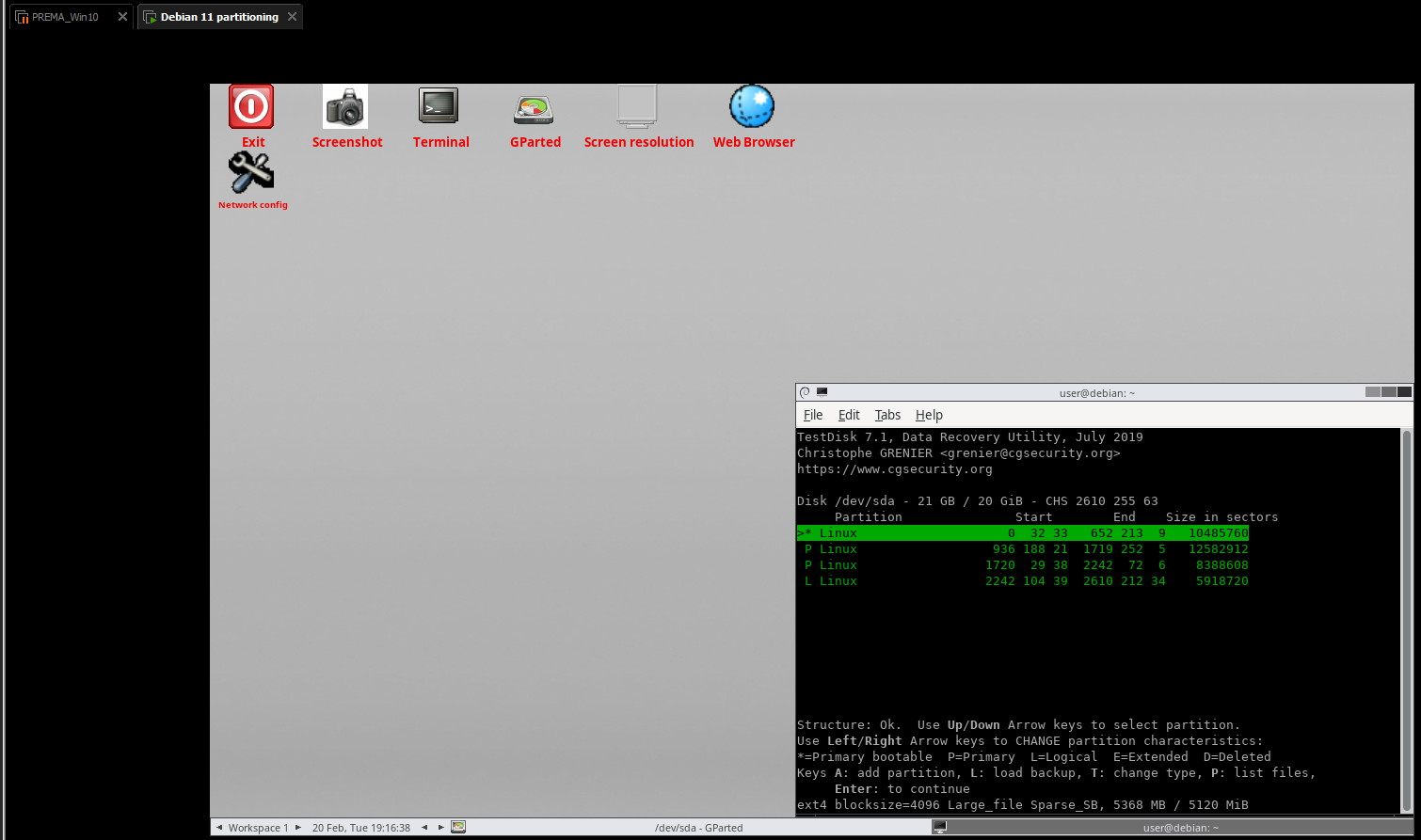


Let’s get that fixed! We’ll use a live boot cd:

* Insert the GParted Live cd and boot from it. Accept the defaults to boot into the graphical environment.

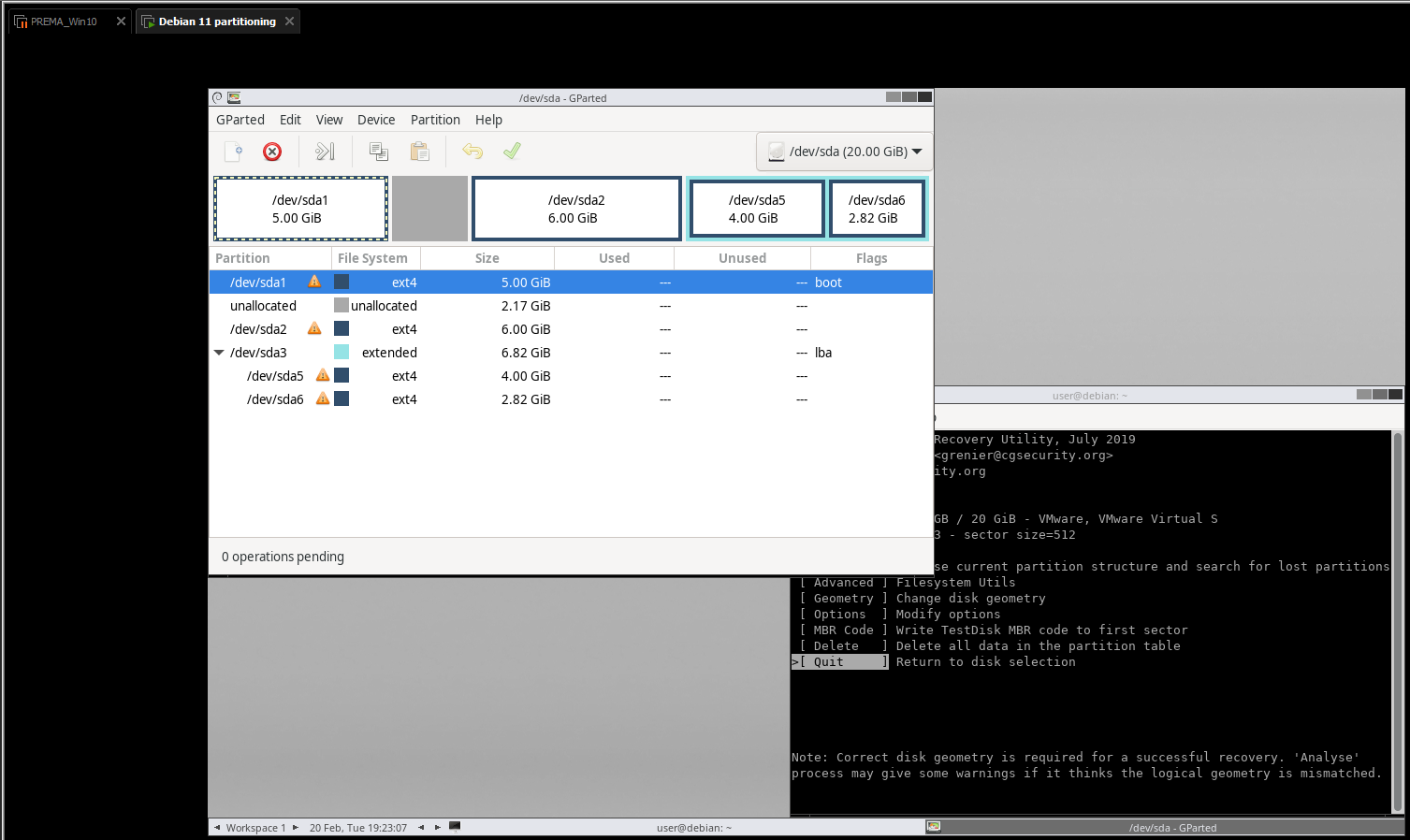


* You’ll notice that GParted doesn’t recognize any partition.
* Minimize or close gparted and start a terminal
* We’ll use the testdisk tool. Start it with sudo testdisk .
* Use ‘No Log’ and select your hard disk
* Use the ‘Intel’ option and analyze the disk

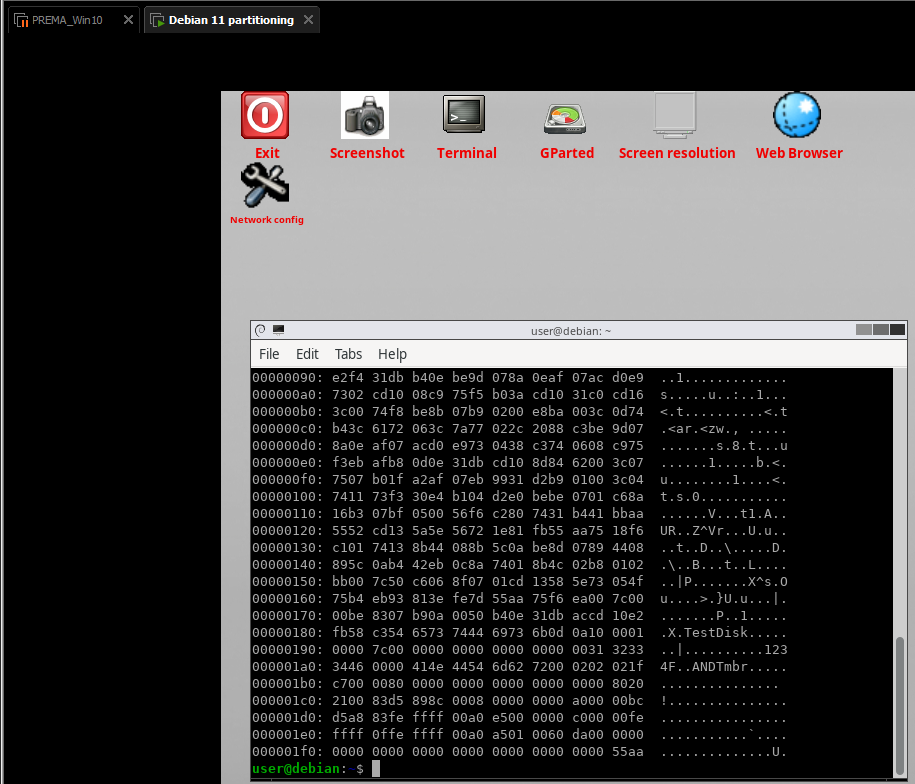


You’ll see testdisk recognizes different partitions (also deleted ones). With right/left arrow you can mark a partition to be put back as primary or logical. The bottom line gives some information, such as the size in MiB (= size in sectors times 512 bytes).

* Select the partitions to be put back in the partition table, as you’ve just created.
  + Notice that in this specific case, the swap partition is not detected/recovered by testdisk
  + You need to select the appropriate ones to be put back as primary or logical. The structure needs to be OK (green), which e.g. means no more than 4 primary partitions (or 3 primary + multiple logical ones). One of your logical partitions is wrongly indicated to be recovered as primary partition. Ensure your 4GiB and 2.8GiB partition are restored as logical partitions
* Hit enter to confirm and choose to ‘write’ to the partition table
* Rerun the gparted tool and you’ll see your partitions back in place!



* Run xxd (using sudo) in the terminal in the live CD to see the MBR. Is GRUB also back?

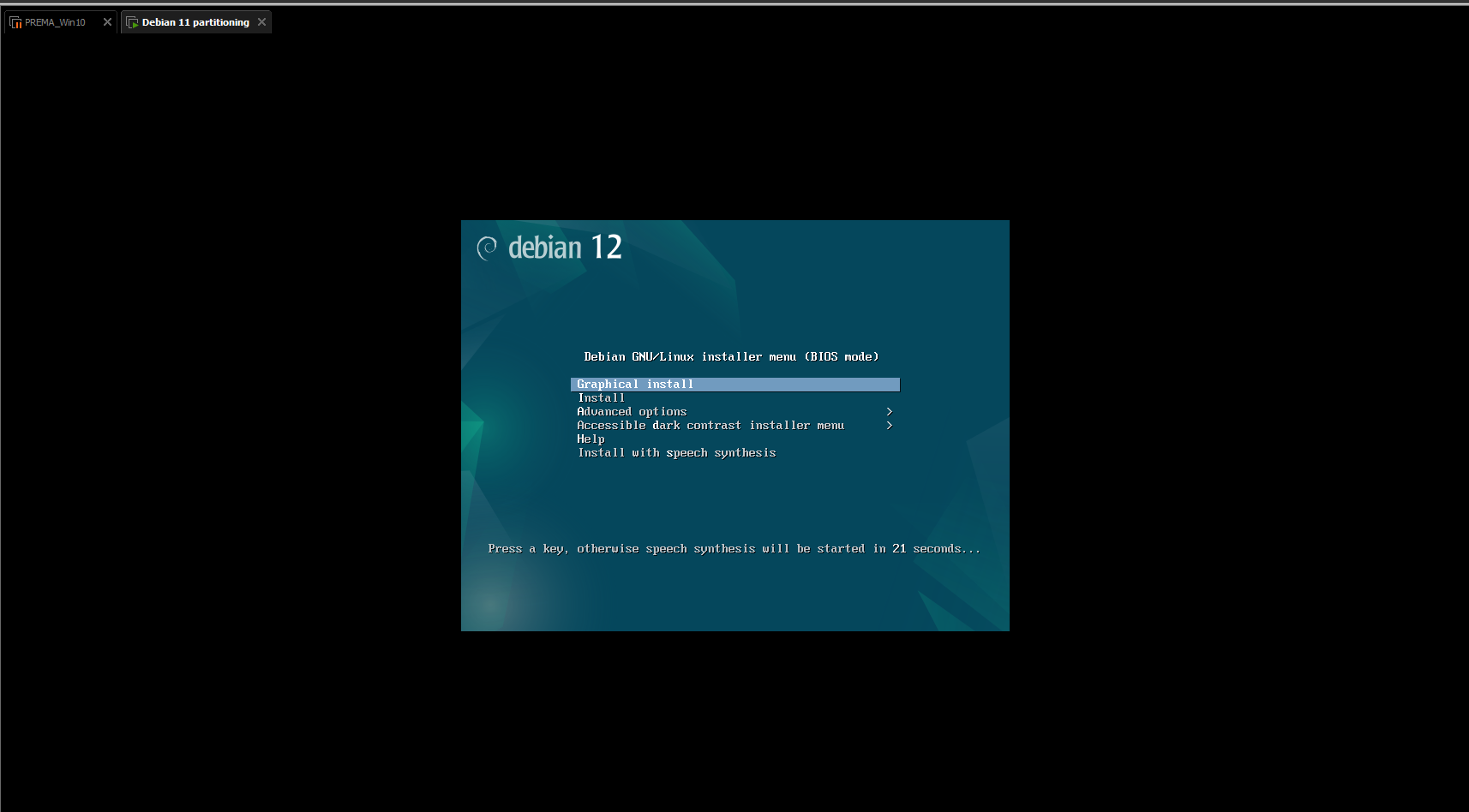


The GRUB does not seem to be there (as we override the bytes with information about it), but the MBR is there.

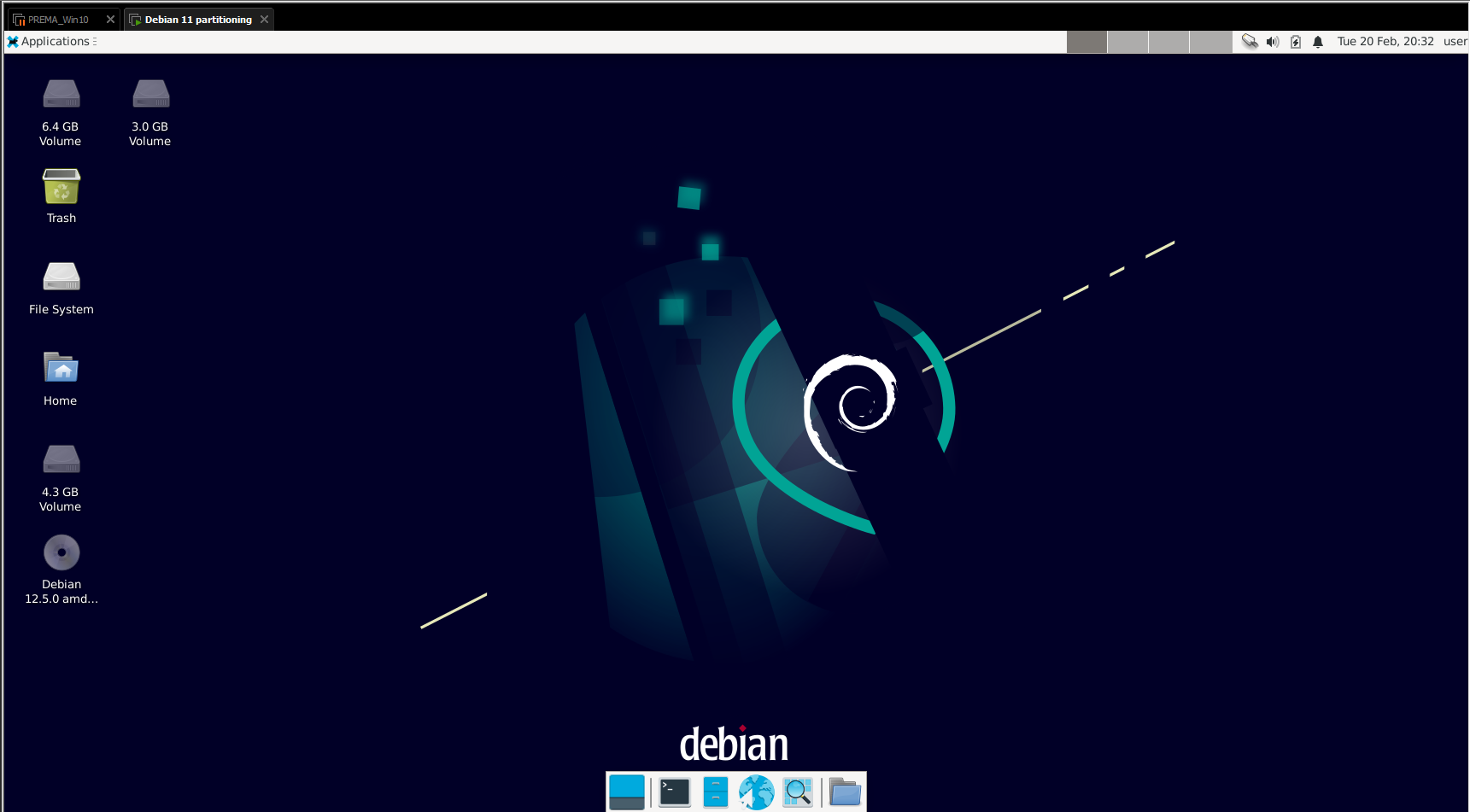
* Reboot the system. If the live cd hangs during reboot, perform a ‘Reset’ of your VM in your VMWare application (this simulates a ‘hard’ power off and on again)

You’ll be presented with “1234F”. This is the boot loader code of testdisk which allows you to boot from partition 1,2,3,4 or floppy by typing that number. However, we don’t have boot code at our partitions. We used the GRUB boot loader. We want to restore that now in the second phase of our recovery.

* Insert the Debian install cd (iso) at your VM. Make sure it is ‘connected’ in your VM.
* You can boot from your (virtual) CD instead of your (virtual) hard disk by hitting ESC once (!) during the BIOS boot screen and choosing the CD drive to boot from. The BIOS boot screen on your VM however disappears in the blink of an eye, so need to be very fast (and lucky) or you might want to extend the boot screen duration by adding a line ‘bios.bootdelay=3000’ to your VM’s vmx file to make it last for e.g. 3 seconds.



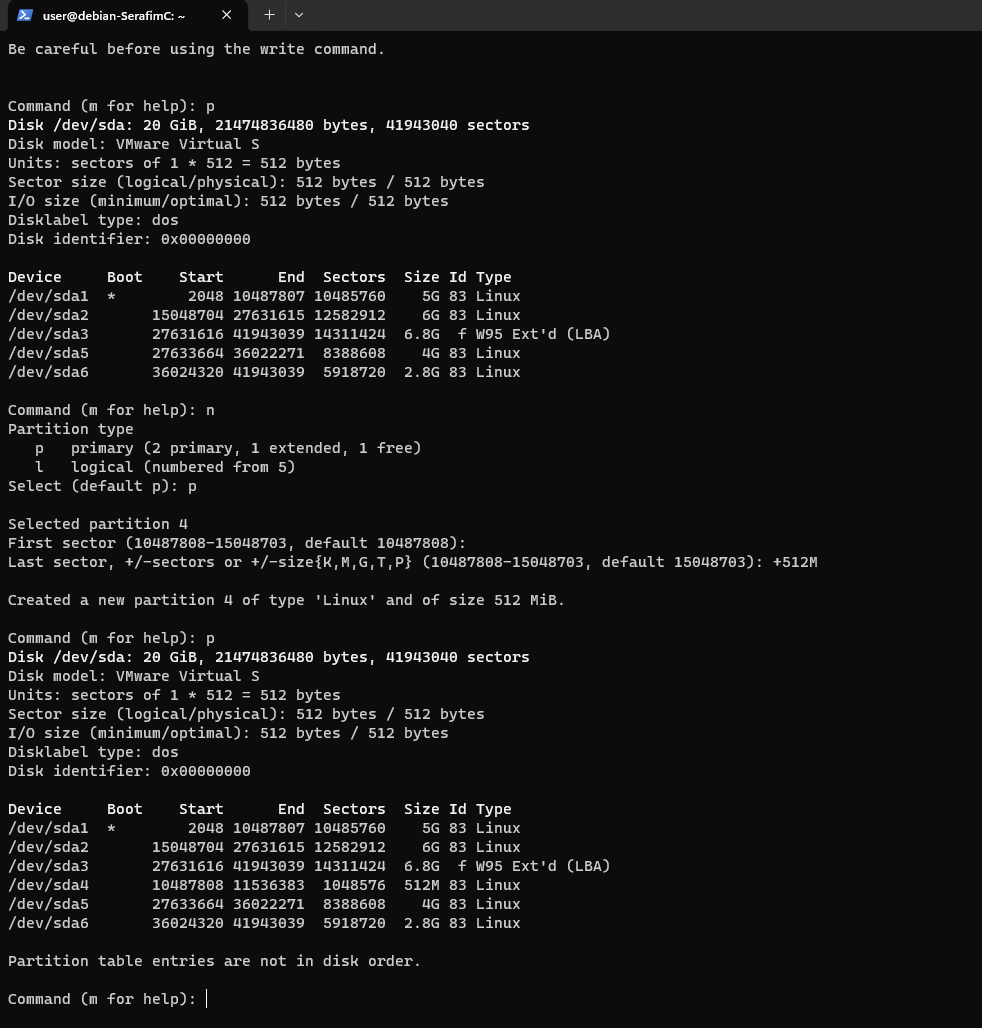
* Choose Advanced – Graphical Rescue mode
* Choose default settings and the keymap of your laptop
* Select the partition on which debian is installed. This was /dev/sda1
* Choose to ‘Reinstall GRUB boot loader’. You won’t see a lot, but it should be OK.
* Choose to Reboot afterwards.
* You’ll notice the linux boot process waits for a minute for the missing swap partition, but then continues booting. E voila, you’re in your debian again !!!



Its alive!

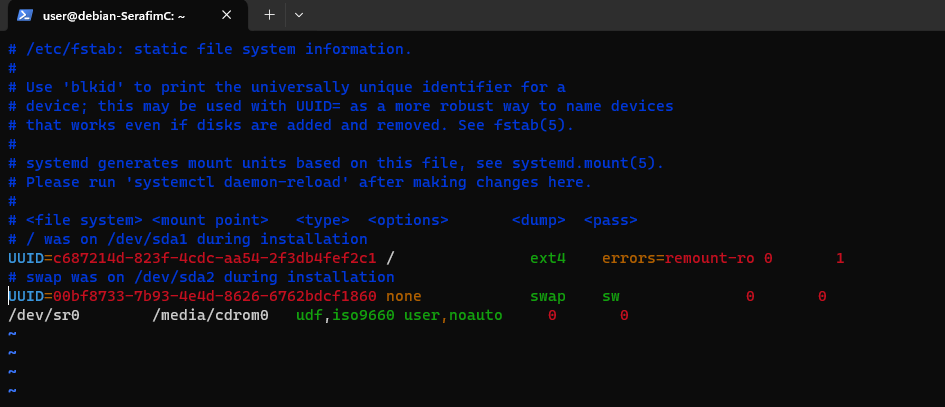
Now finalize recovery by recreating a swap partition (intended for offloading files from computer memory to hard disk):

* Create a new primary partition of 512MiB in the unallocated free space using fdisk (starting at the first proposed sector). It you did all previous steps correctly, it should be created as sda4

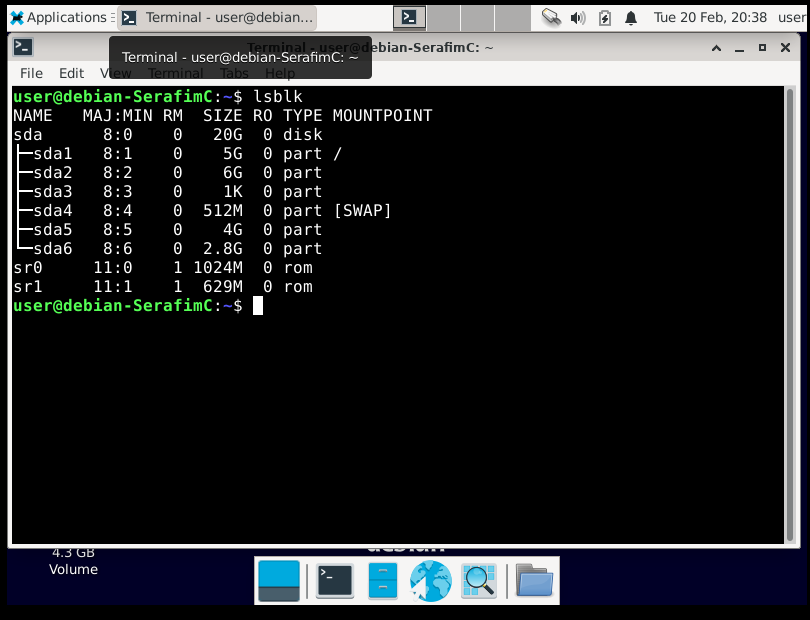


* Make it available for swap by the following command: sudo mkswap /dev/sda4
* The previous command provides you with the UUID of that new partition. In your /etc/fstab file (which dictates the partitions that are automatically mounted), replace the UUID of the swap with this new UUID of your newly created swap partition.

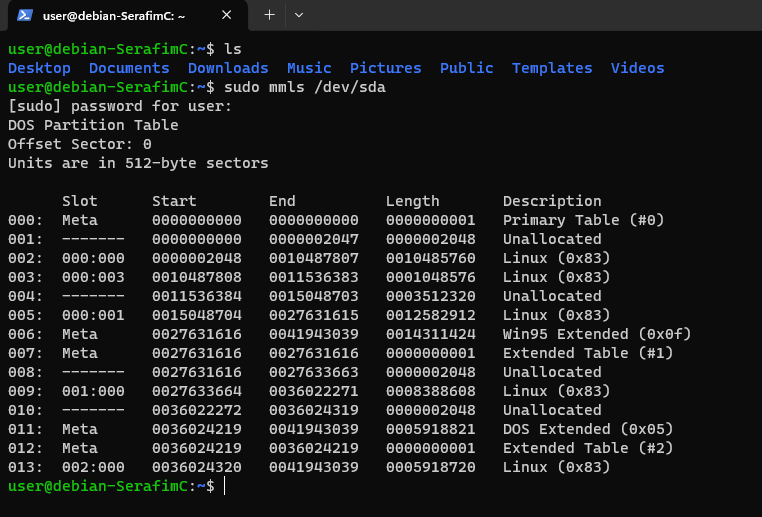
UUID=00bf8733-7b93-4e4d-8626-6762bdcf1860



* Reboot your debian and find out you restored your debian successfully!



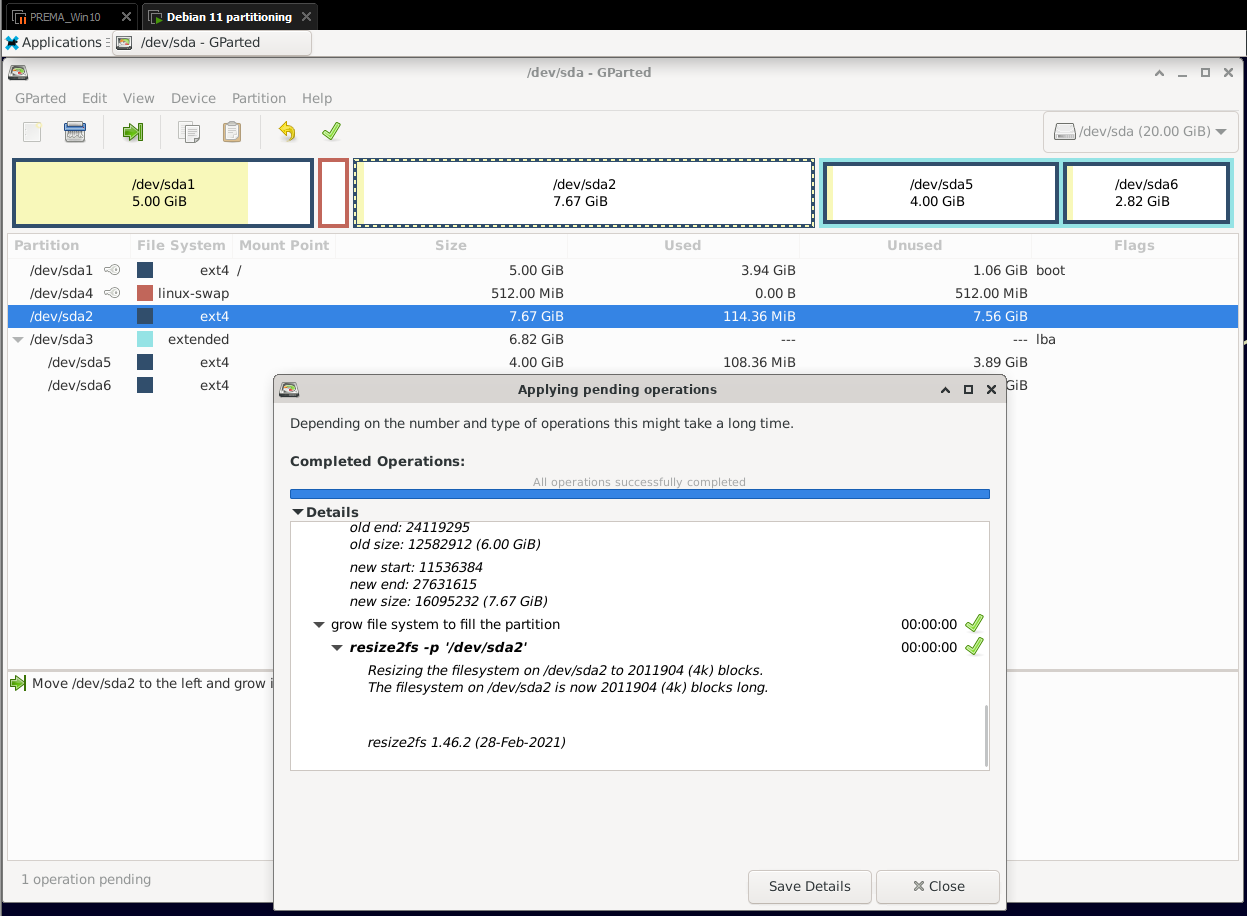
* Use the mmls tool once more to inspect the partitioning. Take a screenshot of the command and its output.



### Resizing partitions

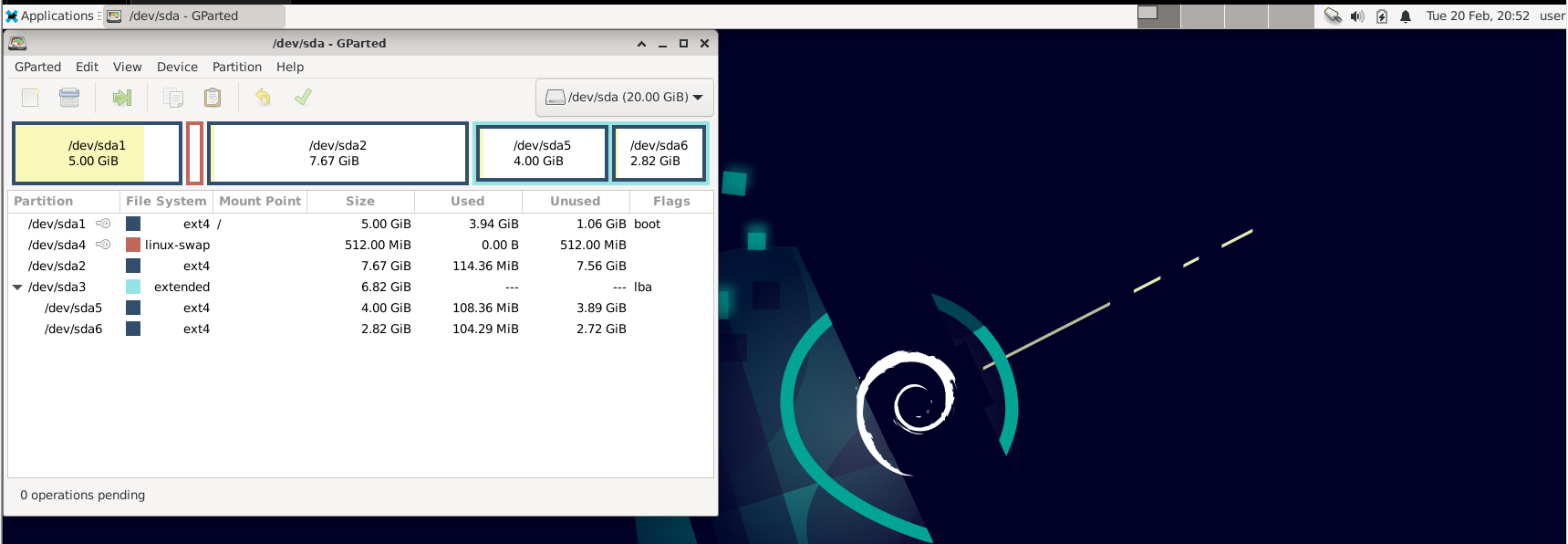
We still have some unallocated disk space, which would be a pity to waste. We will use gparted to easily ‘grow’ a partition to include that disk space.

* Start gparted. You can do this on your running Debian (or via gparted live cd, but not required).
* Select your 6GiB partition and make it use all preceding space via the GUI.
* Apply the change and unfold ‘Details’ to see all involved operations. Notice that apart from growing your partition (i.e. simply editing the entry in the partition table), the file system itself on the partition also needs to be inflated. You can unfold each operation individually. What’s the command used to grow the file system?



Seems like the command is **resize2fs**

* Reboot your debian and verify everything still works.



Everything seems to work