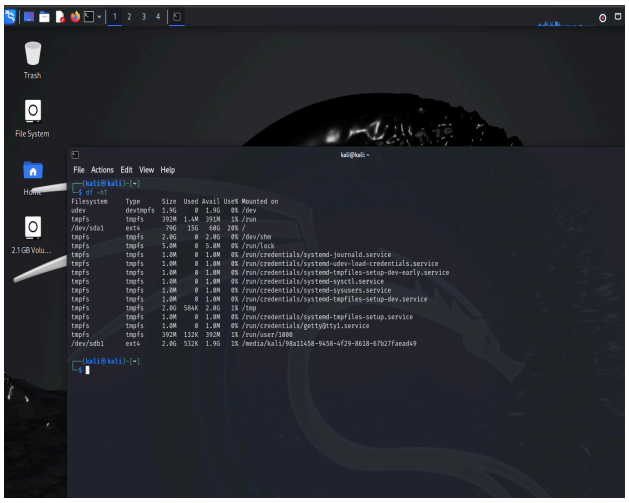
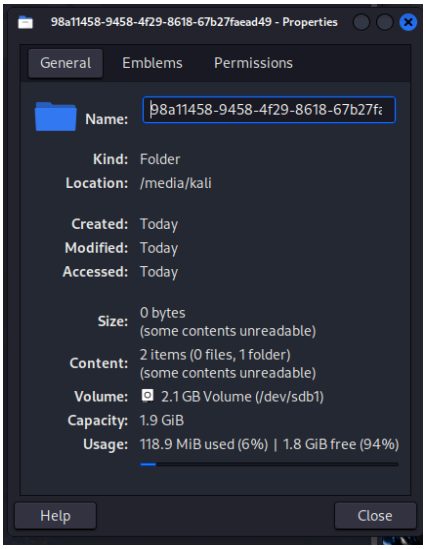
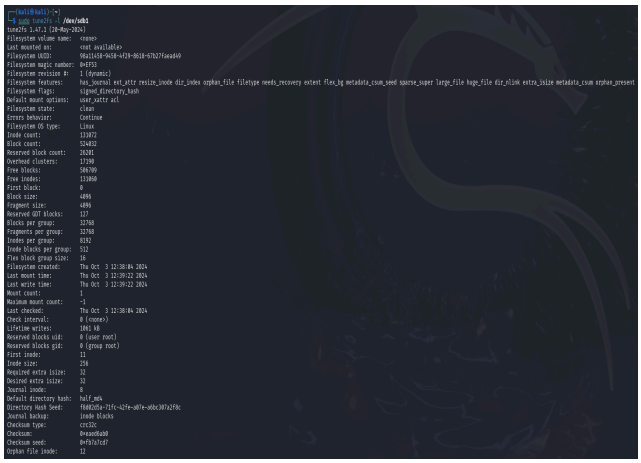


In this step of the lab, I ran the df -hT command to check the disk space usage. This command gives a clear view of how much space is used and available, along with the types of file systems. The output lists various file systems, their types, total sizes, used and available space, the percentage of space used, and where they are mounted. First, there's the udev file system, which is devtmpfs type with a size of 1.9G, all of it free, and it's mounted on /dev. The tmpfs file system at /run is 392M, with just 1.4M used. Another tmpfs at /dev/shm is 2.0G, fully available. My main partition, /dev/sda1, is an ext4 file system with 79G in total, 15G used, and 60G free, mounted on /. There are several tmpfs entries for different systemd services like systemd-journal, udev-load-credentials, tmpfiles-setup-dev, etc., each around 1.0M or 5.0M in size, and most are barely used. I added an extra 2G on my virtual machine mounted on the very bottom.

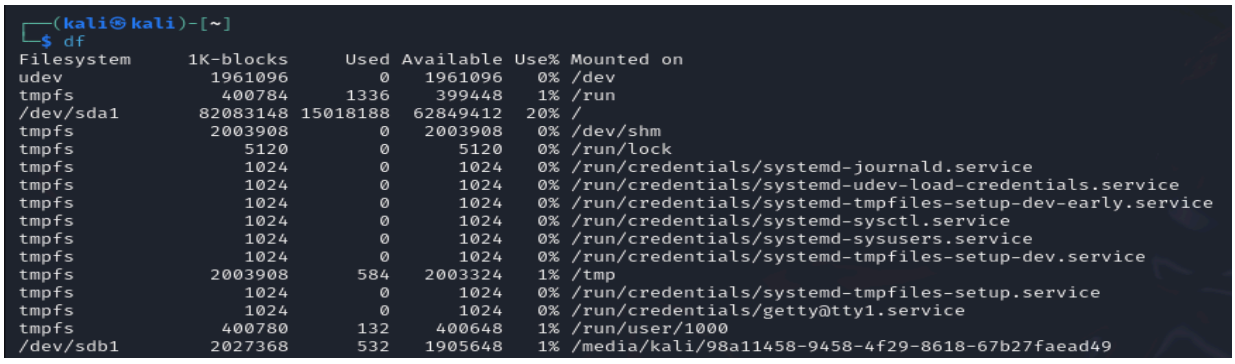
After running the first command, I ran the sudo lshw -class volume -short command on my Kali Linux system. This command lists hardware details focusing on storage volumes in a summarized format. The output shows two volumes: one on /dev/sda1, an 80GiB EXT4 file system, and another on /dev/sdb1, a 2047MiB EXT4 file system. In the real world, this command is useful for quickly identifying and verifying the types and sizes of storage volumes on a system, which can be essential for tasks such as system audits, troubleshooting storage issues, or preparing for system upgrades and maintenance.



This screenshot shows the properties of the folder, located at /media/kali. The folder was created, modified, and accessed today. It has a size of 0 bytes, containing 2 items (0 files, 1 folder), with some contents unreadable. The folder is on a volume labeled 2.1 GB Volume (/dev/sdb1), which has a total capacity of 1.9 GiB, with 118.9 MiB used (6%) and 1.8 GiB free (94%). The general tab shows this information, while the emblems and permissions tabs are also available but not shown.

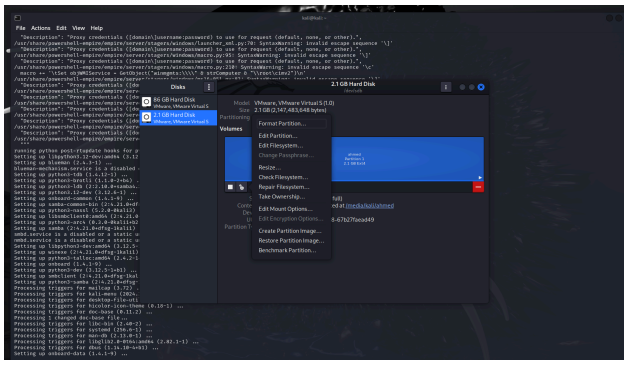


This image shows the output of the sudo tune2fs -l /dev/sdb1 command run on the system, which provides a detailed look at the ext2, ext3, or ext4 filesystem on the partition /dev/sdb1. This command basically gives a rundown of various details about the filesystem's setup and condition. For instance, it shows the volume name, which in this case is <none>, meaning that no specific name has been given to it yet. It also lists the filesystem's unique identifier (UUID), which can be useful for identifying the filesystem in mount settings instead of using the usual device names. Then, it lists the features supported by the filesystem, such ashas_journal, indicating that it uses journaling to help maintain consistency and support data recovery if something goes wrong. Other features like ext_attr, resize_inode, and dir_index are about extending attributes, resizing, and optimizing directory storage. Some other features, like orphan_file, help track orphaned files if there's a crash. The flags section includes signed_directory_hash, which helps guard against hash collisions that could mess up data integrity. Moving on, it also provides some numbers, like the total count of inodes and blocks. The filesystem was created on October 3, 2024, and has both a block size and fragment size of 4096 bytes. This means that 4096 bytes is the smallest amount of disk space that can be allocated to a file. The reserved blocks section shows how many blocks are set aside for the superuser, which is a bit of a safeguard to help with system recovery. It also mentions that the filesystem state is clean, meaning it was unmounted properly and doesn’t need any urgent recovery action. Lastly, it points out that crc32c is the checksum type used to verify the integrity of the filesystem. The`tune2fs -l command is a handy way to get a full picture of what’s going on with a filesystem. This info can be really helpful for managing the filesystem, solving problems, or even doing data recovery if needed.



The df command gives a quick look at how much disk space is being used and how much is left on different parts of the system. Here's a simple rundown of what it shows. The Filesystem column lists all the mounted filesystems. Some of these, like udev and tmpfs, are virtual filesystems used by the system for things like devices and temporary files. In this list, /dev/sda1 is the main disk partition, which is about 78 GB in size. Out of that, around 15 GB is being used, and about 63 GB is still free, meaning it’s 20% full. This partition is mounted at /, which is the

root of the whole file system where most of the operating system files are stored. You'll notice several tmpfs entries, which are temporary storage spaces in the system's RAM. These are used for things like shared memory and temporary files during system operation. For example, /dev/shm has about 2 GB available. The Mounted on column shows where each filesystem can be accessed within the directory structure, like /run, /run/lock, and /tmp, which are for system-related tasks that need quick, temporary storage. At the bottom, you see /dev/sdb1, which is another disk partition with a size of around 2 GB. It's barely used, with only 532 blocks taken up and about 1.9 GB still free, making it just 1% full. This partition is mounted and identified by its UUID.



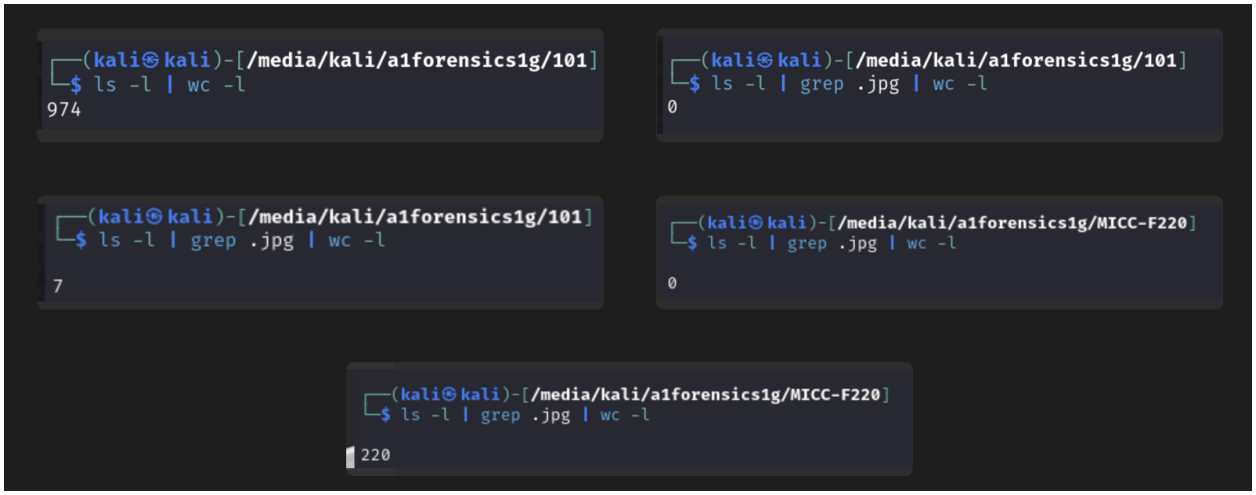
This part of the lab shows the gnome disk utility. This tool provides an easy-to-use graphical interface for handling tasks like formatting, resizing, and mounting partitions, which is much simpler than using command-line tools. On the left side, there are two disk devices listed: an 86 GB hard disk, which is likely the main virtual drive used by the operating system, and a smaller 2.1 GB hard disk (which I added). The 2.1 GB disk is currently selected, and in the middle section, you can see detailed information about this disk. It's formatted with the Ext4 filesystem and is labeled as ahmed. The disk has a total capacity of 2.1 GB, and its only partition is mounted at /media/kali/ahmed. You can also see the disk's unique identifier, or UUID, which the system uses to manage and access it. Just below this, there is a graphical bar that shows the partition, which is fully allocated and formatted as Ext4, represented by the blue color. To the right, there's an options menu with several actions you can take on this partition, like formatting, resizing, and checking or repairing the filesystem. One important option here is Take Ownership, which gives the current user full permissions to read, write, and manage the files on this partition. This is especially useful if you're preparing the partition for tasks like forensic analysis or data recovery. This part of the lab, which corresponds to Step 14, involves using the gnome-disk-utility to make sure the partition is properly set up and accessible. The tool provides a clear and straightforward way to interact with the disk, making complex disk operations more manageable for everyone, whether you're a beginner or more experienced.

```
┌──(kali㉿kali)-[~]
└─$ sudo dd if=/dev/random of=/dev/sdb1 bs=1M status=progress
2065694720 bytes (2.1 GB, 1.9 GiB) copied, 12 s, 172 MB/s
dd: error writing '/dev/sdb1': No space left on device
2048+0 records in
2047+0 records out
2146435072 bytes (2.1 GB, 2.0 GiB) copied, 12.3551 s, 174 MB/s
```

This is Step 15 of the lab, where the dd command is used to wipe the partition clean by filling it with random data. In this step, the command sudo dd if=/dev/random of=/dev/sdb1 bs=1M status=progress is run to overwrite the whole /dev/sdb1 partition. This is important when you're

making the disk ready for things like data recovery or secure deletion because it makes sure that any old data is completely covered up, making it much harder to recover anything that was there before. Let's break down what's happening here: if=/dev/random tells the command to use random data as the input. of=/dev/sdb1 sets the target partition where this data will be written. bs=1M specifies the block size as 1 Megabyte, which means it writes data in chunks of 1 MB at a time, speeding up the process. The status=progress part shows how much data has been copied so far while the command runs. In the output, it shows that 2.1 GB of random data was copied at a speed of about 172 MB/s, and it took around 12 seconds to fill up the partition. The message "No space left on device" pops up once the partition is completely full, which is exactly what we want because it means the entire space has been overwritten. The lines "2048+0 records in" and "2047+0 records out" show how many 1 MB blocks were written to the disk, matching the total size copied. Doing this is a key step in forensics and data recovery since it wipes out traces of old data and gives you a clean slate to work with. In this lab, it's all about making sure the disk is fully sanitized before using it for the next part of the process.

The image shows the dd command being used to wipe the /dev/sdb1 partition by filling it with zeros, just like in Step 15 of the lab. The command sudo dd if=/dev/zero of=/dev/sdb1 bs=1M is basically copying a bunch of zeros to the target partition /dev/sdb1, using 1 Megabyte blocks to speed things up. In the output, you see that it keeps writing until the partition is full, which is confirmed by the "No space left on device" message. The "2048+0 records in" and "2047+0 records out" lines show how many 1 MB blocks were written, adding up to 2.1 GB of data. This took just under 3 seconds at a speed of 733 MB/s. By overwriting everything with zeros, this step wipes out any old data on the disk, making it nearly impossible to recover and giving you a clean slate for data recovery or other forensic work.



The sequence of commands shows a process of listing and counting the total files in two different directories 101 and MICC-F220, with a focus on .jpg files. The counts are first captured before any deletion. Then, after a deletion operation which is not shown in the image but inferred, the commands are re-run to verify that the .jpg files have been removed. The commands involving ls, grep, and wc provide a straightforward way to list and filter files in directories, especially when managing large numbers of files, as is common in digital forensics work.

Executive summary:
In this lab, the goal was to learn how to create and manage virtual disks, clean them of old data, and attempt to recover deleted files. First, I practiced using hash functions, which are like digital fingerprints, to verify files and ensure they haven't been altered. Then, I set up a virtual hard drive within VMware, where I created, formatted, and prepared the disk for analysis. A key part of the lab was to clean the disk by overwriting it with random data or zeros, which makes sure that all old information is erased. I used different Linux commands to do this. Next, I focused on data recovery by using tools designed to find deleted files, like images, on the cleaned disk. The lab provided hands-on experience with recovering files that had been removed, which is a valuable skill in digital forensics.

Even though I ran into some virtual machine issues and wasn’t able to finish the lab to its entirety, the tasks I did complete helped me understand the basics of managing and investigating digital data in a controlled virtual environment.