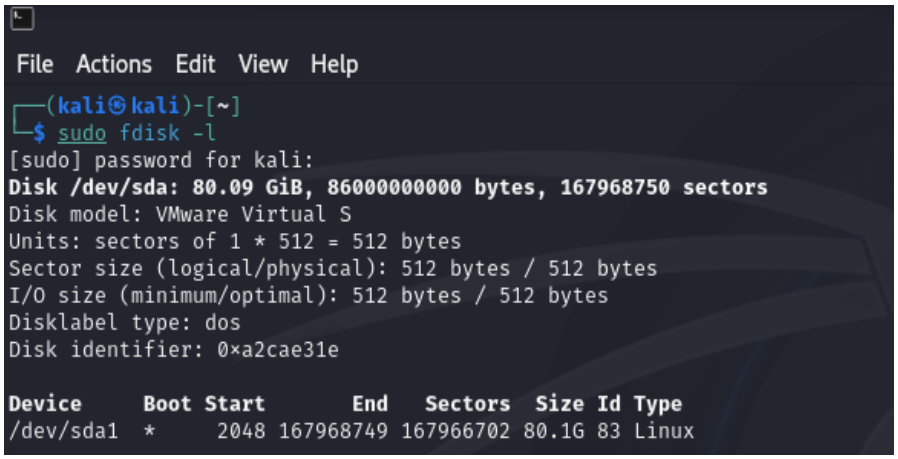
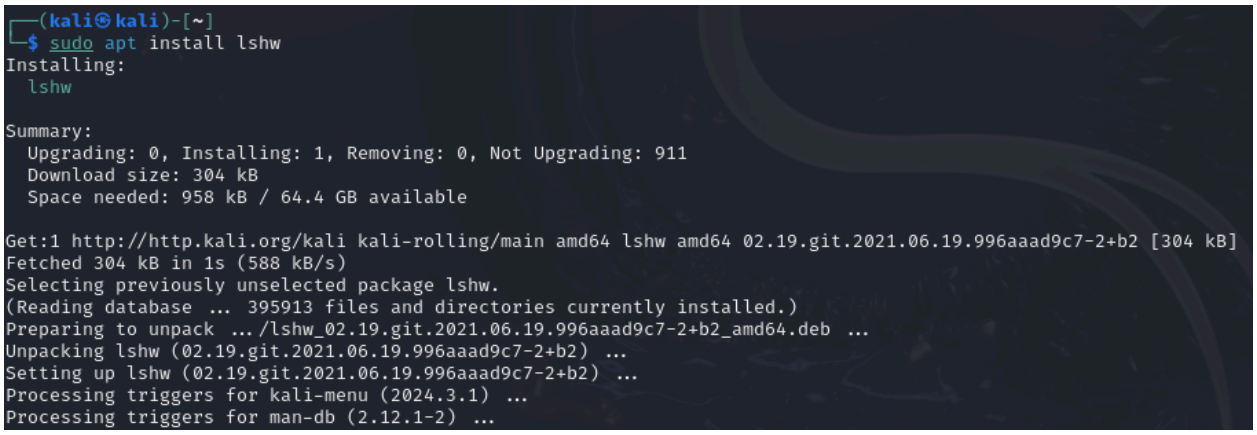


This screenshot shows the use of the fdisk -l command on a Kali Linux terminal. It displays the partition details of /dev/sda, which is an 80.09 GiB disk. It shows sector sizes of 512 bytes, both logically and physically. The partition table is of the dos type, with the disk's identifier. There's one partition (/dev/sda1), marked as bootable, starting at sector 2048 and ending at 167968749, with a size of 80.1 GiB and a Linux file system (Id 83).

This screenshot shows the Kali Linux terminal after navigating to the /dev directory and listing its contents using the ls command. The /dev directory holds device files, representing various hardware components and devices recognized by the Linux system. It shows a range of device files including sda, sda1, and sg0, which correspond to disk and partition devices, as well as other system devices like tty, null, random, and more.

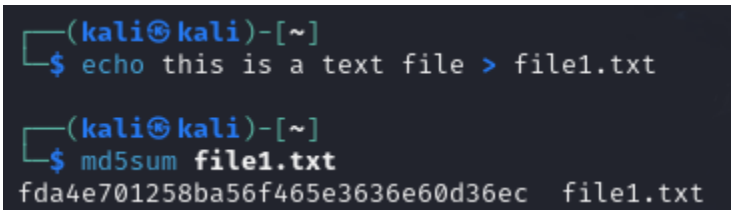


The sudo apt install lshw command is being executed in the terminal to install the lshw tool, which is used to list hardware components of the system. The output shows the process of installing lshw, including fetching package information, unpacking, and setting it up. The installation is successful, with a summary indicating no errors.

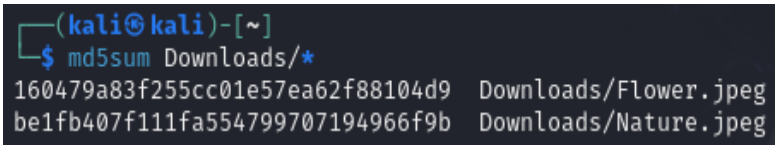
```
┌──(kali㉿kali)-[~]
└─$ sudo lshw -class disk -short
H/W path           Device     Class          Description
=========================================================
/0/100/10/0.0.0    /dev/sda   disk           86GB VMware Virtual S
```

The sudo lshw -class disk -short command is used to generate a concise overview of the disk devices recognized by the system. The output lists one disk under the hardware path /0/100/10/0.0.0, identified as /dev/sda. This disk is classified as disk with a description showing its capacity of 86GB, further indicating the virtualized environment. The output provides a quick summary of the storage hardware without delving into specific partitions or volumes, which is useful for verifying the physical or virtual disks connected to the system.

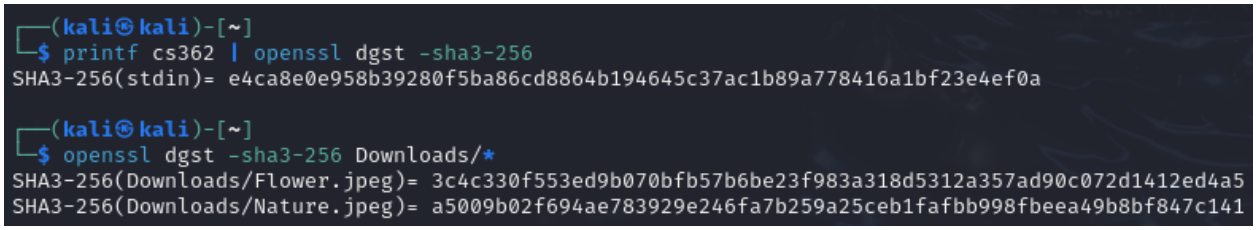
This screenshot displays the use of three different hashing commands in Linux: sha1sum, md5sum, and sha512sum. The command printf cs362 outputs the string cs362, which is then piped into each hashing command to generate a unique hash for the input string. The output shows three different hash values: a 160-bit SHA-1 hash, a 128-bit MD5 hash, and a 512-bit SHA-512 hash, which is a much longer hexadecimal string. This exercise illustrates how various hashing algorithms produce unique digests for a given input, showing the different hash lengths and complexity, which play an important role in digital forensics for verifying data integrity.



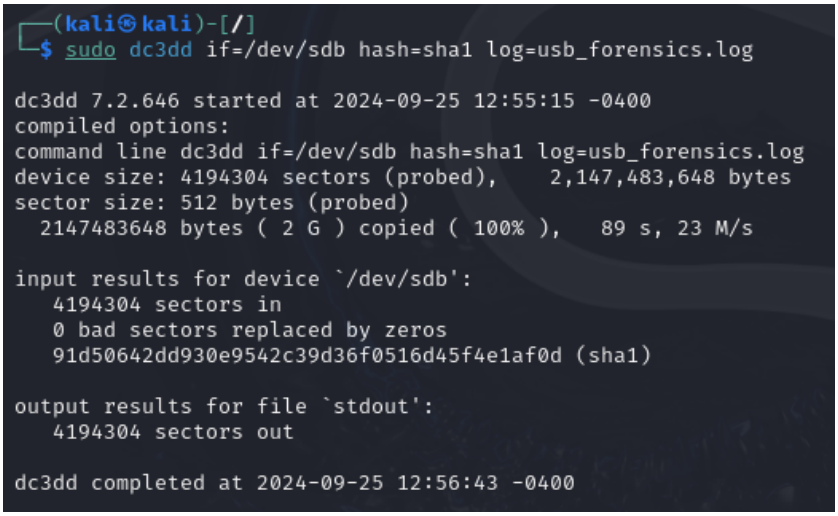
This shows the creation of a simple text file named file1.txt using the echo command with the content "this is a text file." The next command, md5sum file1.txt, generates an MD5 hash for this file, resulting in a hash. This process demonstrates how to create a file and compute its hash, a common practice in digital forensics to verify that the contents of a file have not been altered. The MD5 hash acts as a digital fingerprint for the file's current state.



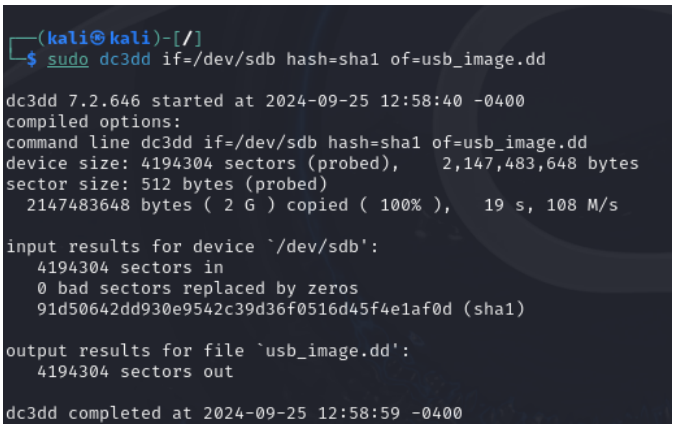
The md5sum command is used to calculate the MD5 hashes for all files in the Downloads directory, indicated by Downloads/*. Two JPEG files, Flower.jpeg and Nature.jpeg, have their hash values displayed. The purpose of generating these hashes is to ensure the integrity of the files, which is especially useful when monitoring files for any unauthorized changes. This method provides a quick way to hash all files within a specific directory.



This screenshot shows the use of the openssl command to generate SHA3-256 hashes for a string and for files in the Downloads directory. The command printf cs362 | openssl dgst -sha3-256 outputs the SHA3-256 hash for the string cs362. The command calculates the SHA3-256 hashes for Flower.jpeg and Nature.jpeg, resulting in two unique hash values. SHA-3 is part of a newer family of cryptographic hash functions, offering enhanced security compared to previous SHA versions. This demonstrates how to use OpenSSL for advanced hashing operations.

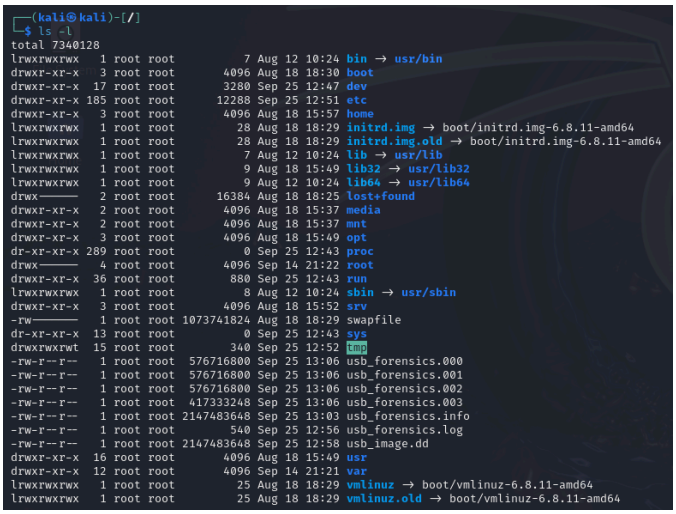


This shows us the use of the dc3dd command to create a forensic image of the /dev/sdb drive. The command specifies the input file (/dev/sdb), the hashing algorithm (SHA-1), and the log file (usb_forensics.log). The output shows that the command successfully copied 2 GiB of data with a sector size of 512 bytes, generating a SHA-1 hash. The log captures details about the imaging process, including the number of sectors read (4,194,304) and confirms that there were no bad sectors. This process is essential in digital forensics to ensure the integrity of the copied data and to create a bit-by-bit copy of the disk for further analysis. The final line indicates the completion of the imaging process, which took about 89 seconds.

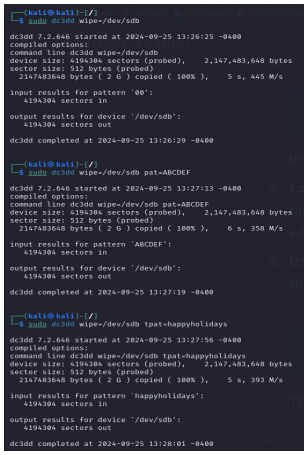


This screenshot shows the use of the dc3dd command to create a forensic image of the /dev/sdb drive and save it as usb_image.dd. The tool reads the drive, generating a SHA-1 hash to ensure data integrity. It successfully copied 2 GiB of data in 19 seconds at a speed of 108 MB/s, confirming that there were no bad sectors. The operation completed successfully, creating a bit-by-bit copy of the drive for forensic analysis.

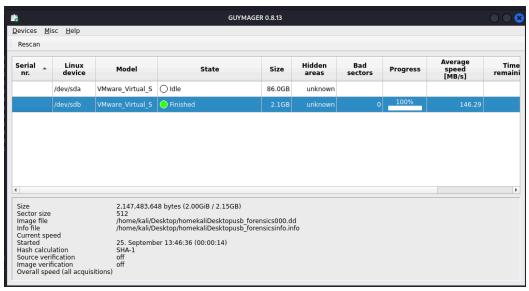
```
┌──(kali㉿kali)-[/]
└─$ ls
bin   dev  home        initrd.img.old  lib32  lost+found  mnt  proc  run   srv       sys  usb_forensics.log  usr  vmlinuz
boot  etc  initrd.img  lib             lib64  media       opt  root  sbin  swapfile  tmp  usb_image.dd       var  vmlinuz.old
```

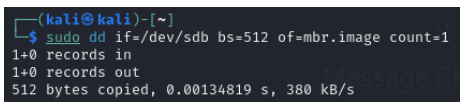


This screenshot shows the output of the ls -l command in the root directory (/), providing a detailed list of files and directories along with their permissions, ownership, sizes, and modification dates. Standard system directories like bin, boot, dev, and etc are present, each with root ownership. here are files named usb_forensics.000, usb_forensics.001, usb_forensics.002, usb_forensics.003, and usb_forensics.info, suggesting the forensic image was either split into parts or accompanied by additional metadata files. This listing provides an overview of the filesystem's current state, highlighting the forensic files recently added or modified.



This step shows the use of the dc3dd tool to wipe the contents of the /dev/sdb drive with different patterns. The first command overwrites the drive with zeros (00), processing 4,194,304 sectors at a speed of 445 MB/s, completing in 5 seconds. The second command uses the pattern "ABCDEF" for wiping, completing in 6 seconds at 358 MB/s. The third command uses the custom text pattern "happyholidays," finishing in 5 seconds at 393 MB/s. Each operation confirms that all sectors were successfully overwritten, ensuring that the data on the drive was securely erased.





The Screenshot on the left displays the interface of Guymager, a forensic disk imaging tool. It shows two devices: /dev/sda and /dev/sdb. The imaging process for /dev/sdb has finished, as indicated by the progress bar and status. The details at the bottom reveal that the disk was successfully imaged with a sector size of 512 bytes and a total size of 2,147,483,648 bytes. The average speed during the process was 146.29 MB/s, and SHA-1 hashing was used for data integrity. The resulting image file is saved, along with an associated info file. This overview confirms that the disk imaging

process was completed successfully and securely. The screenshot on the right shows the use of the dd command to create an image of the Master Boot Record (MBR) of the /dev/sdb drive. The command sudo dd if=/dev/sdb bs=512 of=mbr.image count=1 copies the first 512 bytes (which includes the MBR) from the drive into a file named mbr.image. The output indicates that 512 bytes were successfully copied in a fraction of a second at a speed of 380 kB/s. This operation is commonly used in disk forensics to extract the MBR for analysis.