

CIS-387: Digital Forensics (4 credits)

With Dr. Jinhua Guo

Lab 2

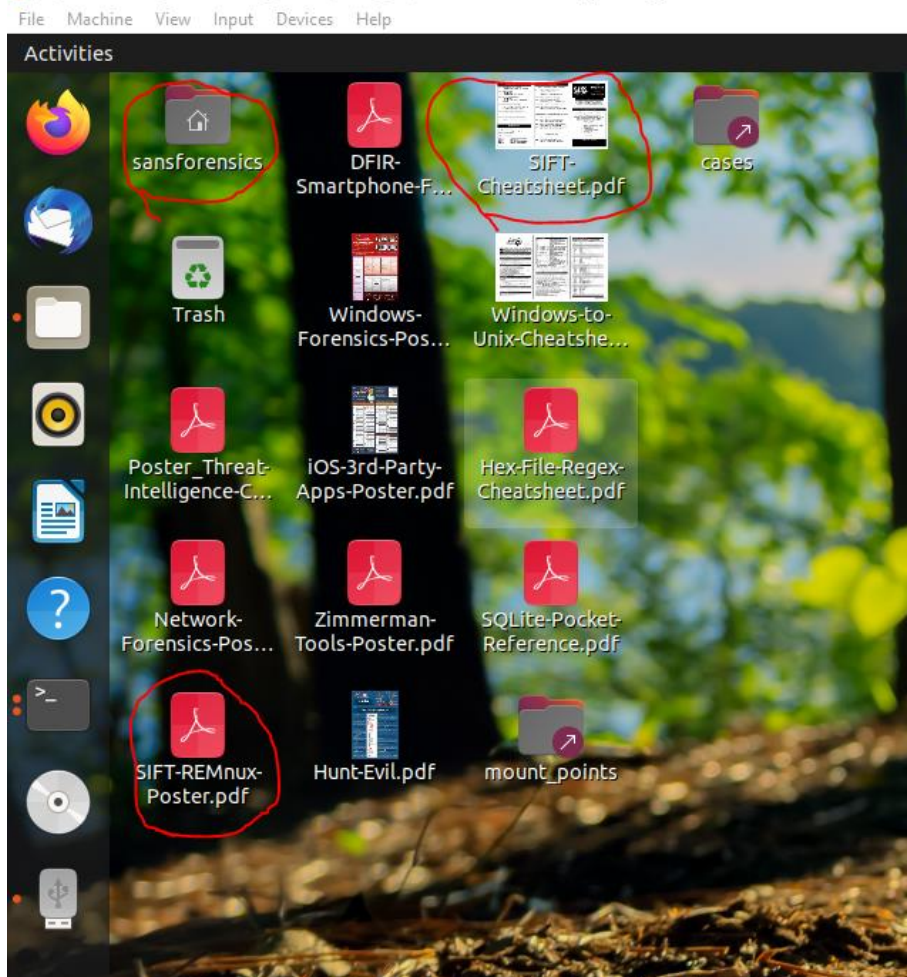
Demetrius Johnson

September 26, 2022

ACTIVITY 1: USING DD TO COPY AND COMPARE FILES

1. Launch SIFT Workstation 3 and open a terminal

DigitalForensics_CIS-387-LAB1_Meech (LAB 1_Snapshot 4 - COMPLETED) [Running] - Oracle VM VirtualBox



2. Use the command `dd` to copy an existing file on your computer. Name the new file `copy.dd`.

```
sansforensics@siftworkstation: ~/host
$ dd if=test.txt of=copy.dd
0+1 records in
0+1 records out
13 bytes copied, 0.002568 s, 5.1 kB/s
sansforensics@siftworkstation: ~/host
$ ls
copy.dd  LiME-master  LiME-master.zip  'New T
sansforensics@siftworkstation: ~/host
$ cat test.txt
cat: test.txt: No such file or directory
sansforensics@siftworkstation: ~/host
$ cat test.txt
Hello World.
sansforensics@siftworkstation: ~/host
$ cat copy.dd
Hello World.
sansforensics@siftworkstation: ~/host
$
```

3. Using `md5sum`, create MD5 hashes of the original file and the copy.

4. Compare the hash of the copy to the hash of the original file; confirm that the hashes are the same.

```
sansforensics@siftworkstation: ~/host
$ man md5sum
sansforensics@siftworkstation: ~/host
$ md5sum test.txt
770b95bb61d5b0406c135b6e42260580  test.txt
sansforensics@siftworkstation: ~/host
$ md5sum copy.dd
770b95bb61d5b0406c135b6e42260580  copy.dd
sansforensics@siftworkstation: ~/host
$
```

5. Repeat Steps 3 and 4 using shasum to generate SHA1 hashes.

```
sansforensics@siftworkstation: ~/host
$ man shasum
sansforensics@siftworkstation: ~/host
$
sansforensics@siftworkstation: ~/host
$ shasum test.txt
b924c2f360b572e17c971f1b1b667e0732944df7 test.txt
sansforensics@siftworkstation: ~/host
$ shasum copy.dd
b924c2f360b572e17c971f1b1b667e0732944df7 copy.dd
sansforensics@siftworkstation: ~/host
$
```

6. Use dd to copy one block of zero from /dev/zero to a file called zero.dd.

(Hint: use the dd option count).

Side note: I am guessing the /dev/zero is a large file with 0s written to it so that blocks of memory can be overwritten with zeroed data. I think this because I did the dd command but did not specify how many blocks to copy to my zero.dd file, then I realized it so I ctrl-c to stop the process, and it wrote 159MB of data to my zero.dd file. Then, When I 'cat zero.dd' to check the contents that was copied to the file, there was a long pause as it was gathering (reading) the file so it could output – but, when cat operation finished, the output was nothing (no characters or anything at all).

See:

```
sansforensics@siftworkstation: ~/host
$ dd if=/dev/zero of=zero.dd
^C310360+0 records in
310360+0 records out
158904320 bytes (159 MB, 152 MiB) copied, 62.9084 s, 2.5 MB/s

sansforensics@siftworkstation: ~/host
$ cat zero.dd
sansforensics@siftworkstation: ~/host
$
```

Also, notice the file size when I issue ls -l command before and after I re-ran the dd command and set count=1 block for copying:

```

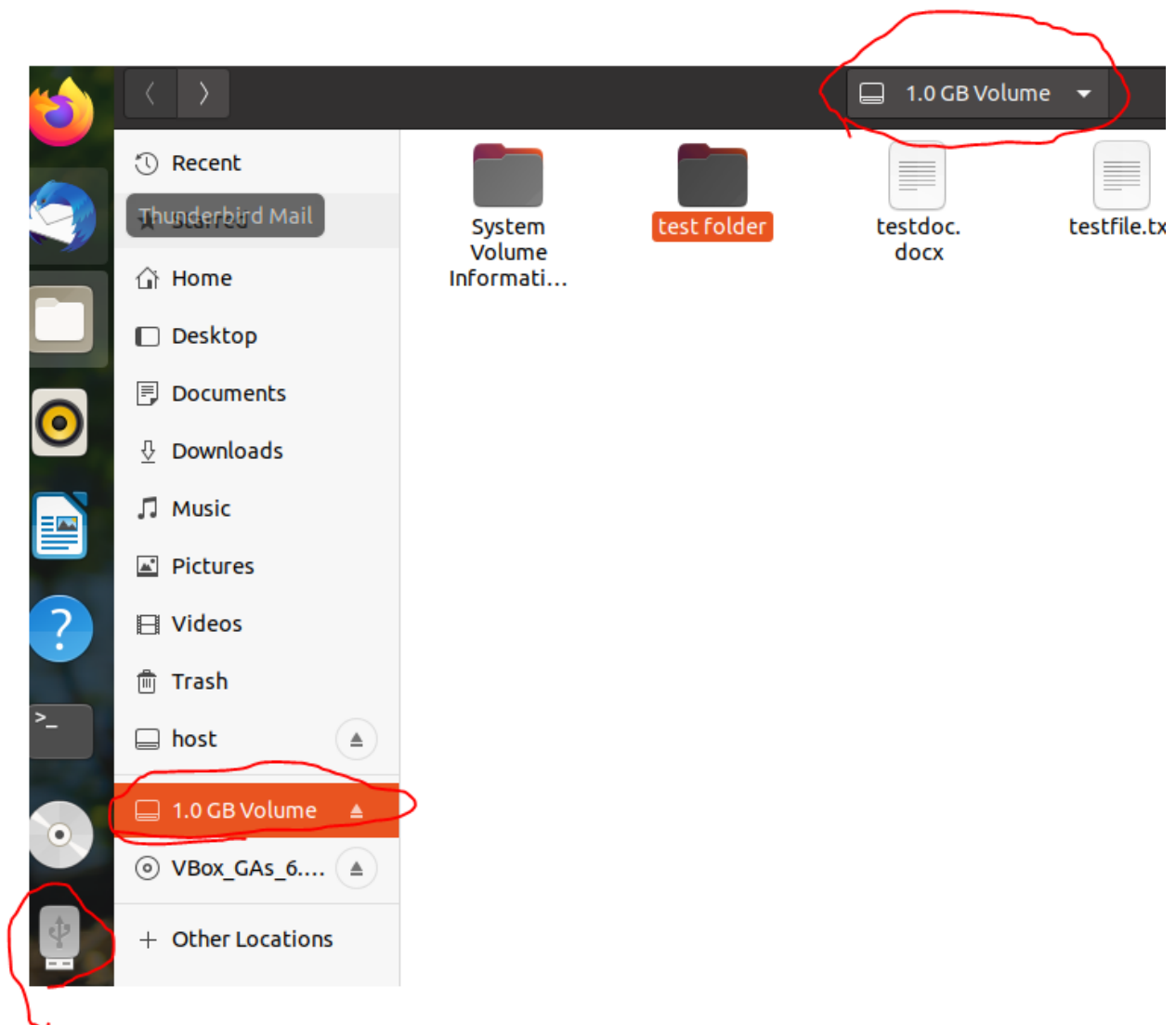
sansforensics@siftworkstation: ~/host
$ ls -l
total 155214
-rwxrwx--- 1 root vboxsf      13 Sep 26 16:51  copy.dd
drwxrwx--- 1 root vboxsf      0 Sep 19 18:27  LiME-master
-rwxrwx--- 1 root vboxsf    30226 Sep 19 18:27  LiME-master.zip
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17  'New Text Document - Copy.t
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17  'New Text Document.txt'
-rwxrwx--- 1 root vboxsf     15 Sep 26 16:32  test_input_file.txt
-rwxrwx--- 1 root vboxsf      0 Sep 26 16:32  test_output_file.txt
-rwxrwx--- 1 root vboxsf     13 Sep 26 16:25  test.txt
-rwxrwx--- 1 root vboxsf 158904320 Sep 26 17:14  zero.dd
sansforensics@siftworkstation: ~/host
$ man ls
sansforensics@siftworkstation: ~/host
$ ls -s
total 155214
      1 copy.dd          32 LiME-master.zip          0 'New Tex
      0 LiME-master      0 'New Text Document - Copy.txt'  1 test_in
sansforensics@siftworkstation: ~/host
$ man ls
sansforensics@siftworkstation: ~/host
$ dd if=/dev/zero of=zero.dd count=1
1+0 records in
1+0 records out
512 bytes copied, 0.00464342 s, 110 kB/s
sansforensics@siftworkstation: ~/host
$ ls -l
total 38
-rwxrwx--- 1 root vboxsf      13 Sep 26 16:51  copy.dd
drwxrwx--- 1 root vboxsf      0 Sep 19 18:27  LiME-master
-rwxrwx--- 1 root vboxsf    30226 Sep 19 18:27  LiME-master.zip
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17  'New Text Document - Copy.txt'
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17  'New Text Document.txt'
-rwxrwx--- 1 root vboxsf     15 Sep 26 16:32  test_input_file.txt
-rwxrwx--- 1 root vboxsf      0 Sep 26 16:32  test_output_file.txt
-rwxrwx--- 1 root vboxsf     13 Sep 26 16:25  test.txt
-rwxrwx--- 1 root vboxsf     512 Sep 26 17:24  zero.dd
sansforensics@siftworkstation: ~/host
$

```

Finally, I note what I learned in class today (9/26/22) that on many systems, the minimum write size is 512 bytes, and the minimum block size is 1 sector (512 bytes= the typical size of 1 sector for the main storage, such as HDD); so writing one block (cluster) meant writing 512 bytes.

7. Insert the USB drive and connect your USB to SIFT Workstation 3.

The USB drive should auto-mount. (NOTE: In a real investigation, you should use a write blocker to prevent the SIFT Workstation from modifying the USB drive.)



8. Run the command `mount` to find the USB device file name.

You will use the device file name in command `dd` to make a full image of your USB. For example, my USB's device file is `/dev/sdb`; it is mounted on `/media/sansforensics/003B-38D3`.

(Hint: using `lsblk` or `sudo fdisk -l` command to find it out)

```
sansforensics@siftworkstation: /dev
$ sudo fdisk -l
Disk /dev/sda: 488.29 GiB, 524288000000 bytes, 1024000000 sectors
Disk model: HARDDISK
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: dos
Disk identifier: 0x60ccc656

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sda1                2048     3999743     3997696    1.9G 82 Linux swap / Solaris
/dev/sda2    *   3999744 1023997951 1019998208 486.4G 83 Linux

Disk /dev/sdb: 960 MiB, 1006632960 bytes, 1966080 sectors
Disk model:
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: dos
Disk identifier: 0x6f20736b

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sdb1                778135908 1919645538 1141509631 544.3G 72 unknown
/dev/sdb2                168689522 2104717761 1936028240 923.2G 65 Novell Netware 386
/dev/sdb3                1869881465 3805909656 1936028192 923.2G 79 unknown
/dev/sdb4                  0 3637226495 3637226496    1.7T  d unknown

Partition table entries are not in disk order.
sansforensics@siftworkstation: /dev
$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda    8:0    0 488.3G  0 disk
├─sda1  8:1    0   1.9G  0 part [SWAP]
└─sda2  8:2    0 486.4G  0 part /
sdb    8:16   1   960M  0 disk /media/sansforensics/B0B0-B854
sr0    11:0   1   58.4M  0 rom  /media/sansforensics/VBox_GAs_6.1.32
sansforensics@siftworkstation: /dev
$
```

From above, I issued both the fdisk and the lsblk commands; I have found the location of my 1 GB (960MiB, where MiB means “mebibytes; prefix mebi = 2^{20}) USB device folder; I also have found the mount point of the drive; I realize I need device file location in order to do the image using the dd command because the mount point is not the root of the drive; I want to copy all contents of the drive bit-for-bit or byte-for-byte including file system formatting data. So, I must use /dev/sdb location.

Comment from professor Guo: It is because we are doing a Disk-to-Image acquisition instead of a logical acquisition.

9. Use dd to make a full image of your USB flash drive. Name the image usb.dd.

(Hint: dd if=/dev/sdb of=usb.dd)

```
sansforensics@siftworkstation: ~/host
$ sudo dd if=/dev/sdb of=usb.dd
1966080+0 records in
1966080+0 records out
1006632960 bytes (1.0 GB, 960 MiB) copied, 432.637 s, 2.3 MB/s
sansforensics@siftworkstation: ~/host
$ ls
copy.dd  LiME-master  LiME-master.zip  'New Text Document - Copy.txt'  'New Text
sansforensics@siftworkstation: ~/host
$ ls -l
total 983078
-rwxrwx--- 1 root vboxsf      13 Sep 26 16:51 copy.dd
drwxrwx--- 1 root vboxsf      0 Sep 19 18:27 LiME-master
-rwxrwx--- 1 root vboxsf    30226 Sep 19 18:27 LiME-master.zip
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17 'New Text Document - Copy.txt'
-rwxrwx--- 1 root vboxsf      0 Sep 12 19:17 'New Text Document.txt'
-rwxrwx--- 1 root vboxsf      15 Sep 26 16:32 test_input_file.txt
-rwxrwx--- 1 root vboxsf      0 Sep 26 16:32 test_output_file.txt
-rwxrwx--- 1 root vboxsf      13 Sep 26 16:25 test.txt
-rwxrwx--- 1 root vboxsf 1006632960 Sep 26 18:12 usb.dd
-rwxrwx--- 1 root vboxsf      512 Sep 26 17:24 zero.dd
sansforensics@siftworkstation: ~/host
```

10. Create both MD5 and SHA1 hashes of the USB flash.

(Hint: md5sum /dev/sdb; shasum /dev/sdb)

11. Create both MD5 and SHA1 hashes of the USB image.

(Hint: md5sum usb.dd; shasum usb.dd)

12. Make sure that:

The md5 hash of the USB flash matches with the md5 hash of the USB image.

The sha1 hash of the USB flash matches with the sha1 hash of the USB image.

(10, 11, and 12 are all under same screenshot):


```
sansforensics@siftworkstation: ~/host
$ md5sum /dev/sdb
md5sum: /dev/sdb: Permission denied
sansforensics@siftworkstation: ~/host
$ sudo md5sum /dev/sdb
8a81125fb5297f126a47f35ff1e25c50 /dev/sdb
sansforensics@siftworkstation: ~/host
$ sudo sha1 /dev/sdb
sudo: sha1: command not found
sansforensics@siftworkstation: ~/host
$ sudo shasum /dev/sdb
85ddc96cc60ce9a51a9b11b84778c56e52725f96 /dev/sdb
sansforensics@siftworkstation: ~/host
$ sudo md5sum usb.dd
8a81125fb5297f126a47f35ff1e25c50 usb.dd
sansforensics@siftworkstation: ~/host
$ sudo shasum usb.dd
85ddc96cc60ce9a51a9b11b84778c56e52725f96 usb.dd
sansforensics@siftworkstation: ~/host
$
```

ACTIVITY 2: IMAGING WITH NETCAT OVER A NETWORK

1. Launch SIFT Workstation 3.
2. Open two terminals on SIFT Workstation 3. One terminal represents a forensic machine; the other represents the suspect machine.

CIS-387-LAB1_Meech (LAB 1_Snapshot 4 - COMPLETED) [Running] - Oracle VM VirtualBox

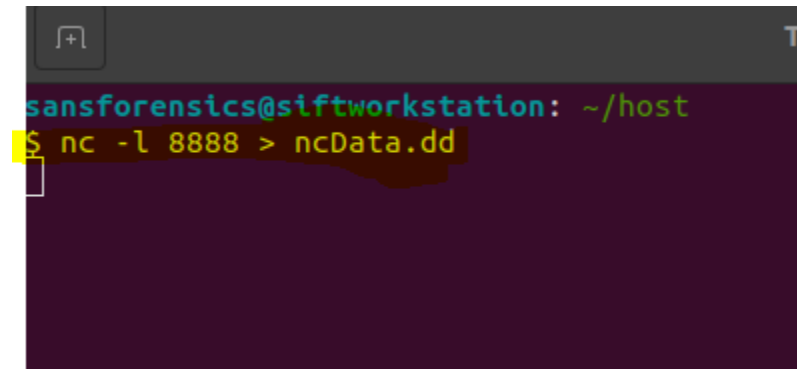
```
Terminal
> 1.0 GB Volume
Sep 26 19:22

sansforensics@siftworkstation: /home
$ cd sansforensics
sansforensics@siftworkstation: ~
$ cd host
sansforensics@siftworkstation: ~/host
$ ls
copy.dd  LiME-master  LiME-master.zip  'New Text Document - Copy.txt'  'New Text Document.txt'  test_input_file.txt  test_output_file.txt  test.txt  zero.dd
sansforensics@siftworkstation: ~/host
$ sudo dd if=/dev/sdb of=usb.dd
1966080+0 records in
1966080+0 records out
1006632960 bytes (1.0 GB, 960 MiB) copied, 432.637 s, 2.3 MB/s
sansforensics@siftworkstation: ~/host
$ ls
copy.dd  LiME-master  LiME-master.zip  'New Text Document - Copy.txt'  'New Text Document.txt'  test_input_file.txt  test_output_file.txt  test.txt  usb.dd  zero.dd
sansforensics@siftworkstation: ~/host
$ ls -l
total 983078
-rwxrwx--- 1 root vboxsf      13 Sep 26 16:51 copy.dd
drwxrwx--- 1 root vboxsf        0 Sep 19 18:27 LiME-master
-rwxrwx--- 1 root vboxsf    30226 Sep 19 18:27 LiME-master.zip

sansforensics@siftworkstation: ~
$ ls
Desktop  Documents  Downloads  host  Music  Pictures  Public  Templates
sansforensics@siftworkstation: ~
$ cd ..
sansforensics@siftworkstation: /home
$ ls
sansforensics
sansforensics@siftworkstation: /home
$
```

3. On the forensic machine terminal, use `nc -l` to listen on port 8888 for the incoming data. Save the received data as `ncData.dd`.

(Hint: `nc -l 8888 > ncData.dd`)

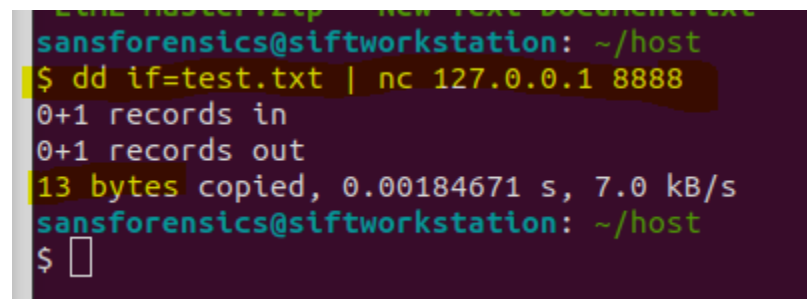


```
sansforensics@siftworkstation: ~/host
$ nc -l 8888 > ncData.dd
```

4. On the suspect machine terminal, use `dd` to copy an existing file and pipe (`|`) to netcat (`nc`), sending copy of the file to the forensic machine terminal.

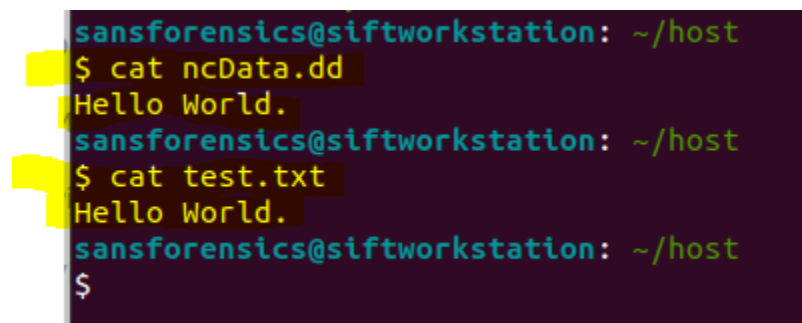
Since we are sending data to the same machine, we use local host's loopback IP address 127.0.0.1. If you send data to a networked machine, replace 127.0.0.1 with the receiving machine's IP address.

(Hint: In our case, we run `dd if=test-original-file | nc 127.0.0.1 8888`)



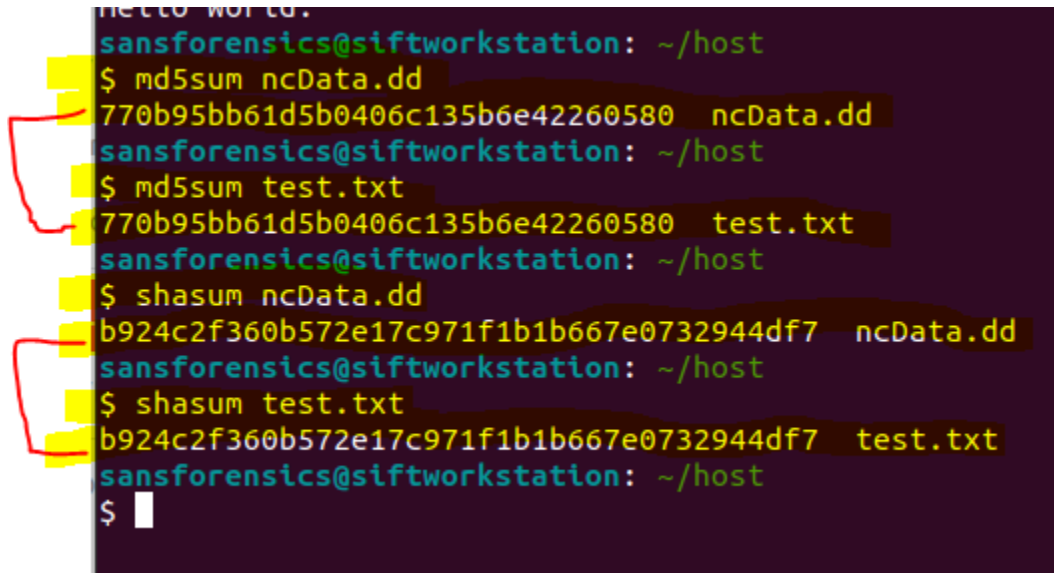
```
sansforensics@siftworkstation: ~/host
$ dd if=test.txt | nc 127.0.0.1 8888
0+1 records in
0+1 records out
13 bytes copied, 0.00184671 s, 7.0 kB/s
sansforensics@siftworkstation: ~/host
$
```

Notice file contents are the same:



```
sansforensics@siftworkstation: ~/host
$ cat ncData.dd
Hello World.
sansforensics@siftworkstation: ~/host
$ cat test.txt
Hello World.
sansforensics@siftworkstation: ~/host
$
```

5. Generate MD5 and SHA1 hashes of ncData.dd and compare them with the original file's MD5 and SHA1 hashes



```
net to world.  
sansforensics@siftworkstation: ~/host  
$ md5sum ncData.dd  
770b95bb61d5b0406c135b6e42260580  ncData.dd  
sansforensics@siftworkstation: ~/host  
$ md5sum test.txt  
770b95bb61d5b0406c135b6e42260580  test.txt  
sansforensics@siftworkstation: ~/host  
$ shasum ncData.dd  
b924c2f360b572e17c971f1b1b667e0732944df7  ncData.dd  
sansforensics@siftworkstation: ~/host  
$ shasum test.txt  
b924c2f360b572e17c971f1b1b667e0732944df7  test.txt  
sansforensics@siftworkstation: ~/host  
$
```

A terminal window screenshot with a dark purple background. It shows a series of commands and their outputs. The prompt is 'sansforensics@siftworkstation: ~/host'. The first command is '\$ md5sum ncData.dd', which outputs '770b95bb61d5b0406c135b6e42260580 ncData.dd'. The second command is '\$ md5sum test.txt', which outputs '770b95bb61d5b0406c135b6e42260580 test.txt'. The third command is '\$ shasum ncData.dd', which outputs 'b924c2f360b572e17c971f1b1b667e0732944df7 ncData.dd'. The fourth command is '\$ shasum test.txt', which outputs 'b924c2f360b572e17c971f1b1b667e0732944df7 test.txt'. The prompt returns to '\$' at the end. Red brackets on the left side of the terminal window group the MD5 and SHA1 command pairs together.

Notice above, hash output is the same for both hash algorithms, signifying that we have made an exact copy of test.txt file.

Summary/Reflection

Overall, I learned the power of dd and netcat commands. I also understand better how file systems work, and why drives need to be formatted so that the OS can read and understand the connected device. Also, I learned better how to use the redirect operators > and |, where > is used to send output to a file, and | is used to send output to another function (process) to be used as input. I also am more familiar with the practicality of using hash algorithms to check data integrity.

Now, if I ever need to make sure that some copy I made of a very important file is an exact copy, and if later I want to make sure that my file did not get corrupted, I can compute the hash of the original file, save that hash value somewhere (like write it down), then later compute the hash to check its integrity and the integrity of any other copies I have made of the file to see if even the slightest change to any of the file data contents were changed. This is a very powerful tool and can actually save a lot of time, and gives you the best possible data integrity check in even the most practical sense.