

## dcfldd - Latest version 1.3.4-1

1.

```
root@siftworkstation:/home/sansforensics/Documents# mkdir usb
```

First create a folder to stroage the image files.

## 2.

```

root@siftworkstation: /tmp
root@siftworkstation: /tmp
5860.008770] sd 3:0:0:0: [sdb] Mode Sense: 1f 00 00 00
5860.008770] sd 3:0:0:0: [sdb] Write cache: disabled, read cache: enabled, does not support DPO or FUA
5860.091694] sdb: sdb1
5860.141813] sd 3:0:0:0: [sdb] Attached SCSI removable disk
5860.671295] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
5880.837757] sd 3:0:0:0: Attached scsi generic sg2 type 0
5880.841560] sd 3:0:0:0: [sdb] 15223808 512-byte logical blocks: (7.79 GB/7.26 GiB)
5880.848547] sd 3:0:0:0: [sdb] Write Protect is off
5880.848548] sd 3:0:0:0: [sdb] Mode Sense: 43 00 00 00
5880.855337] sd 3:0:0:0: [sdb] Write cache: disabled, read cache: enabled, does not support DPO or FUA
5880.912226] sdb: sdb1
5880.963927] sd 3:0:0:0: [sdb] Attached SCSI removable disk
5881.390500] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
5886.168989] blk_update_request: I/O error, dev sdb, sector 13558376 op 0x1:(WRITE) flags 0x0 phys_seg 3 prio class 0

```

Using the command "dmesg | grep" sd this command will focus on an SD drive  
the picture above shows that sdb1 is an FAT-fs it the drive name

### 3.

```
root@siftworkstation:/home/sansforensics/Documents/usb# dcflddd if=/dev/sdb1 conv
=sync,noerror hash=sha256 hashlog=hash.log of=/usb/usb.img
170240 blocks (5320Mb) written.
```

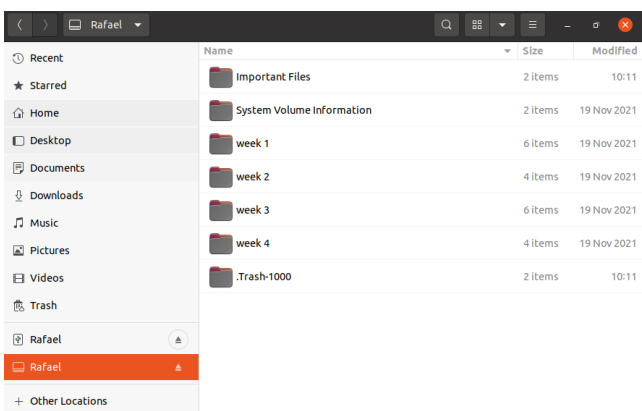
Now I'm aware of the drive's name. I can generate an image of the drive without modifying the actual drive. Imaging is the process of duplicating bits by bits of the original drive; depending on the size of the hdd, this procedure may take a long time.

```
dcflddd if=/dev/sdb1 conv=sync,noerror hash=sha256
hashlog=hash.log of=/usb/usb.img of=/usb/usb.img
of=/usb/usb.img of=/usb/usb.img of=/usb/usb.img
of=/usb/usb.img of=/usb
```

**4.**

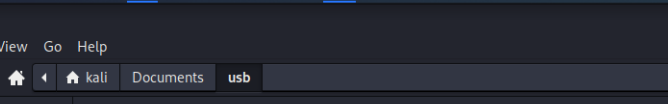
```
root@siftworkstation: /home/sansforensics/Documents/usb
root@siftworkstation: /home/sansforensics/Documents/usb# mount -r /usb/usb.img /media/usb-image
root@siftworkstation: /home/sansforensics/Documents/usb#
```

Once I finish creating the image i can mount the .img file in the media folder.  
The image below shows the mounted .img



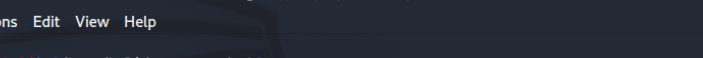
**1.**

# Bulk-Extractor



Copy and paste the image file created using dcfldd to somewhere accessible and easy to remember

## 2.



The screenshot shows a terminal window with the title bar 'root@kali: /home/kali/Documents/usb'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal prompt is '(root@kali)-[/home/kali/Documents/usb]'. The command being executed is '# bulk\_extractor -o bulk-out usb.img'.

```
root@kali: /home/kali/Documents/usb
File Actions Edit View Help
(root@kali)-[/home/kali/Documents/usb]
# bulk_extractor -o bulk-out usb.img
```

### 3.

The screenshot displays the Kali Linux desktop environment. The terminal window is open, showing the output of the 'usb' command. The output includes details about a USB device (ID 1040187392) and its usage statistics. The file manager window is also open, showing the contents of the '/home/kali/Documents/usb' directory, which includes files 'bulk-out', 'hash.log', and 'usb.img'. The desktop background is a dark blue gradient with a white 'Kali' logo.

once the the command finish running, there will be a output folder in the folder where the .img is stored

4

5

[illegible]

For example, running the command "nano email.txt" will display all of the email addresses saved in the.img file, eliminating the need for the user to go through the file one by one. Same goes for different information such as credit card number.