‘dd’ command in Linux

* Difficulty Level : [Medium](https://www.geeksforgeeks.org/medium/)
* Last Updated : 15 May, 2019

**dd** is a command-line utility for Unix and Unix-like operating systems whose primary purpose is to convert and copy files.

* On Unix, device drivers for hardware (such as hard disk drives) and special device files (such as /dev/zero and /dev/random) appear in the file system just like normal files.
* dd can also read and/or write from/to these files, provided that function is implemented in their respective drivers
* As a result, dd can be used for tasks such as backing up the boot sector of a hard drive, and obtaining a fixed amount of random data.
* The dd program can also perform conversions on the data as it is copied, including byte order swapping and conversion to and from the ASCII and EBCDIC text encodings.

**Usage :**The command line syntax of dd differs from many other Unix programs, in that it uses the syntax *option=value* for its command line options, rather than the more-standard *-option value* or *–option=value* formats. By default, dd reads from stdin and writes to stdout, but these can be changed by using the if (input file) and of (output file) options.

**Some practical examples on ddcommand :**

1. **To backup the entire harddisk :** To backup an entire copy of a hard disk to another hard disk connected to the same system, execute the dd command as shown. In this dd command example, the UNIX device name of the source hard disk is /dev/hda, and device name of the target hard disk is /dev/hdb.
2. **# dd if = /dev/sda of = /dev/sdb**
   * *“if”* represents inputfile, and *“of”* represents output file. So the exact copy of */dev/sda* will be available in */dev/sdb*.
   * If there are any errors, the above command will fail. If you give the parameter *“conv=noerror”* then it will continue to copy if there are read errors.
   * Input file and output file should be mentioned very carefully. Just in case, you mention source device in the target and vice versa, you might loss all your data.
   * To copy, hard drive to hard drive using dd command given below, sync option allows you to copy everything using synchronized I/O.
   * **# dd if = /dev/sda of = /dev/sdbconv=noerror, sync**
3. **To backup a Partition :**You can use the device name of a partition in the input file, and in the output either you can specify your target path or image file as shown in the dd command.
4. **# dd if=/dev/hda1 of=~/partition.img**
5. **To create an image of a Hard Disk :**Instead of taking a backup of the hard disk, you can create an image file of the hard disk and save it in other storage devices. There are many advantages of backing up your data to a disk image, one being the ease of use. This method is typically faster than other types of backups, enabling you to quickly restore data following an unexpected catastrophe.It creates the image of a hard disk /dev/hda.
6. **# dd if = /dev/hda of = ~/hdadisk.img**
7. **To restore using the Hard Disk Image :**To restore a hard disk with the image file of an another hard disk, the following dd command can be used
8. **# dd if = hdadisk.img of = /dev/hdb**

The image file hdadisk.imgfile, is the image of a /dev/hda, so the above command will restore the image of /dev/hda to /dev/hdb.

1. **To create CDROM Backup :**dd command allows you to create an iso file from a source file. So we can insert the CD and enter dd command to create an iso file of a CD content.
2. **# dd if = /dev/cdrom of = tgsservice.isobs = 2048**

dd command reads one block of input and process it and writes it into an output file. You can specify the block size for input and output file. In the above dd command example, the parameter *“bs”* specifies the block size for the both the input and output file. So dd uses ***2048bytes*** as a block size in the above command.

**Linux DD Command - 15 Examples with All Options**

By [Leena](https://linoxide.com/author/leena/" \o "View all posts by Leena)

 | Updated March 18, 2021 | Categories[Commands](https://linoxide.com/category/commands/)

The Linux dd command is one of the most powerful utility which can be used in a variety of ways. This tool is mainly used for copying and converting data, hence it stands for data duplicator. This tool can be used for:

* Backing up and restoring an entire hard drive or a partition.
* Creating virtual filesystem and backup images of CD or DVDs called ISO files
* Copy regions of raw device files like backing up MBR (master boot record).
* Converting data formats like ASCII to EBCDIC.
* Converting lowercase to uppercase and vice versa.

Only the superuser can execute this command. You should be very careful while using this command as improper usage may cause huge data loss. So, some people consider this tool as data destroyer.

Syntax of dd command

The basic use of the dd command is rather easy because it takes just two arguments: *if*= to specify the input file and *of*= to specify the output file. The arguments to those options can be either files or block devices. I would, however, not recommend using dd to [copy files](https://linoxide.com/linux-cp-command/) because cp does that in a much simpler way. However, you can use it to clone a hard disk. The syntax is

dd if=<source file name> of=<target file name> [Options]

We will learn various options while going through dd command examples.

**1. Backing up and restoring an entire disk or a partition**

It is possible to save all the data from an entire disk/partition to another disk/partition. Not a simple copy as cp command but a block size copy.

a. Backup entire disk to disk

You can copy all the data (entire disk) from the disk /dev/sda to /dev/sdb. dd doesn’t know anything about the filesystem or partitions; it will just copy everything from /dev/sda to /dev/sdb. You need to indicate the block size to be copied at time with bs option. So, this will clone the disk with the same data on the same partition.

# dd if=/dev/sda of=/dev/sdbbs=4096 conv=noerror,sync

97281+0 records in

97280+0 records out

99614720 bytes (100 MB) copied, 2.75838 s, 36.1 MB/s

This works only if the second device is as large as or larger than the first. Otherwise, you get truncated and worthless partitions on the second one. Here, if stands for input file , of stands for output file and bs stands for the block size (number of bytes to be read/write at a time). Make sure you use block sizes in multiples of 1024 bytes which is equal to 1KB. If you don't specify block size, dd use a default block size of 512 bytes. The conv value parameter *noerror* allows the tool to continue to copy the data even though it encounters any errors. The sync option allows to use synchronized I/O.

b. Creating dd disk image (file image)

You can create an image of a disk or a file image. Backing up a disk to an image will be faster than copying the exact data. Also, disk image makes the restoration much easier.

# dd if=/dev/sda of=/tmp/sdadisk.img

You can store the output file where you want but you have to give a filename ending with .img extension as above. Instead of /tmp/sdadisk.img, you could store it for example at /sdadisk.img if you want.

c. Creating a compressed disk image

Because dd creates the exact content of an entire disk, it means that it takes too much size. You can decide to compress the disk image with the command below

# dd if=/dev/vda | gzip -c >/tmp/vdadisk.img.gz

The pipe | operator makes the output on the left command become the input on the right command. The -c option writes output on standard output and keeps original files unchanged.

d. Backup a partition or clone one partition to another

Instead of an entire disk, you can only backup a simple partition. You just need to indicate the partition name in input file as below

# dd if=/dev/sda1 of=/dev/sdb1 bs=4096 conv=noerror,sync

This will synchronize the partition /dev/sda1 to /dev/sdb1. You must verify that the size of /dev/sdb1 should be larger than /dev/sda1. Or you can create a partition image as below

# dd if=/dev/sda1 of=/tmp/sda1.img

e. Restoring a disk or a partition image

# Save a disk or a partition helps to restore all the data, if there is any problem with our original drive. To restore, you need to inverse the input file with the output file indicated during backup operation as below.

# dd if=/tmp/sdadisk.img of=/dev/sda

You will retrieve data that were present before the backup operation and not after the operation

e. Restoring compressed image

You need to first indicate the [compressed file](https://linoxide.com/16-tar-commands-compress-extract-files-linux/) and the output file which is the disk compressed before.

# gzip -dc /tmp/vdadisk.img.gz | dd of=/dev/vda

The -d option here is to uncompress. Note the output file. You can mount the restored disk to see the content. Note that you will data added after the last compression backup operation.

**2. Creating virtual filesystem/Backup images of CD or DVDs as iso files**

You can need to create a virtual filesystem on Linux for some reasons as creating a virtual machine on your Linux host. You can also need to create a backup [iso image](https://linoxide.com/how-to-create-cd-iso-image-from-linux/" \t "_blank) of a CD or DVD

a. Creating a virtual filesystem

A virtual filesystem is a filesystem that exists in a file, which in turn exists on a physical disk. You can need it to create for example an additional swap or loop device or a virtual machine. We need /dev/zero which is a file used to [create a file](https://linoxide.com/learn-how-create-file-linux-terminal/) with no data but with required size (a file with all zero’s). In other words, this will create a data file with all zeros in the file which will give the size to a file.

# dd if=/dev/zero of=/file bs=1024K count=500

500+0 records in

500+0 records out

524288000 bytes (524 MB) copied, 1.21755 s, 431 MB/s

The option count refers to the number of input blocks to be copied. Combined with block size value, it indicates the total size to copy. For example bs=1024k and count=500 give a *size=1024K\*500 =524288000 bytes =524MB*

Now let's check the size of our file

# ls -lh /file

-rw-r--r-- 1 root root 500M May 17 18:57 /file

You can see that we have our virtual filesystem created with the size indicated. You can now use it to create loop device or a virtual disk or anything else.

b. Modify the first 512 bytes of a file with null data

If during the operation you indicate an existing output file, you will lose its data. For some reasons, you can need to replace a block size of the output file.

dd if=/dev/zero of=file1 bs=512 count=1 conv=notrunc

The notrunc option refers to do not truncate the file, only replace the first 512 bytes, if it exists. Otherwise, you will get a 512 byte file

c. Creating a backup iso image of CD or DVD

You may wonder why not just copy the contents of your CD to a directory. How would you handle the boot sector of a CD? You can’t find that as a file on the device because it’s just the first sector. Because dd copies sector by sector, on the other hand, it will copy that information as well.

# dd if=/dev/cdrom of=/mycd.iso

You need to know that you have to use the -o loop option, which allows you to mount a file like any normal device. So, to mount /mycd.iso on the /mnt/cd directory, do as below

# mount -o loop /mycd.iso /mnt/cd

d) Create a bootable USB Drive

Lets assume we have downloaded centos iso image here to this directory "~/Downloads/".

if=~/Downloads/CentOS-8-x86\_64-1905-dvd1.iso - input file is in the path '~/Downloads/CentOS-8-x86\_64-1905-dvd1.iso'

of=/dev/sdb - output file is in the path '/dev/sdb'

bs=1M - Read from '~/Downloads/CentOS-8-x86\_64-1905-dvd1.iso' and write to '/dev/sdb' 1 Megabytes of data at a time.

# dd if=~/Downloads/CentOS-8-x86\_64-1905-dvd1.iso of=/dev/sdb bs=1M

**3. Backing up and restoring MBR**

The [GRUB bootloader](https://linoxide.com/best-difference-between-linux-grub-and-grub2-bootloader/) is most commonly stored in the MBR of the bootable drive. The MBR makes up the first 512 bytes of the disk, allowing up to 466 bytes of storage for the bootloader. The additional space will be used to store the partition table for that drive.  If MBR gets corrupted, we will not be able to boot into Linux.

a. Backing up MBR

Because the MBR makes up the first 512 bytes of the disk, we just need to copy that block size

# dd if=/dev/sda of=/tmp/sdambr.imgbs=512 count=1

With the count=1 and bs=512, only 512 bytes will be copied which correspond to the size of our MBR.

You can display the saved MBR with the od command which dump files in octal and other formats as below

# od -xa /tmp/sdambr.img

0000000 bf52 81f4 8b66 832d 087d 0f00 e284 8000

R ? tsoh f vt - etx } bsnulsieot b nulnul

0000020 ff7c 7400 6646 1d8b 8b66 044d 3166 b0c0

| delnul t F fvtgs f vt M eot f 1 @ 0

-a option  selects named characters and -x selects hexadecimal 2-byte units

b. Backing up the boot data of MBR excluding the partition table

The MBR 512 bytes data is located at the first sector of the hard disk. It consists of 446 bytes bootstrap, 64 bytes partition table and 2 bytes signature. It means that we can exclude the partition table and bytes signature while backing up the MBR with conserving only a block size equal to the bootstrap size.

# dd if=/dev/sda of=/tmp/sdambr2.img bs=446 count=1

c. Restoring MBR from MBR image

You can restore your MBR as shown on the previous commands with

# dd if=/tmp/sdambr.img of=/dev/sda

**3. Converting data formats**

If an input file uses a character set that is not the native character set of the host computer, the import operator must perform a conversion. For example, if ASCII is the native format for strings on your host computer, but the input data file represents strings using EBCDIC, you must convert EBCDIC to ASCII and vice versa.

a. Convert the data format of a file from EBCDIC to ASCII

If there’s an ebcdic file with you, mostly retrieved from mainframe systems, then, you would like to convert them to ASCII for making modifications using text editors on UNIX servers

# dd if=textfile.ebcdic of=textfile.asciiconv=ascii

The conv value parameter now is *ascii* because we convert from EBCDIC to ASCII

b. Convert the data format of a file from ASCII to EBCDIC

After modifying the ASCII version and once done, you may convert it back to EBCDIC to be used by your application.

# dd if=textfile.ascii of=textfile.ebcdicconv=ebcdic

The conv value parameter now is *ebcdic* because we convert from ASCII to EBCDIC. If you’re just replacing particular number of bytes with an equivalent number of bytes having different characters, the conversion would be smooth and application reading the file should not have any issues.

**4. Converting case of a file**

DD command can be also used for an amazing thing. It can convert all text (alphabets) in a [file to upper or lower case](https://linoxide.com/linux-tr-command/) and vice versa. For the example below, we will have a file for the tests.

# cat file10

testdd convert

a. Converting a file to uppercase

Because our text file example is on lowercase, we will convert it to uppercase

# dd if=~/file10 of=~/file20 conv=ucase

The command will create the new file indicated. See that now conv option takes *ucase* value. Let's check the result

# cat file20

TEST DD CONVERT

b. Converting a file to lowercase

Now we will do the reverse operation which will convert to lowercase

# dd if=~/file20 of=~/file30 conv=lcase

See that we use *lcase*of conv option to convert from upper case to lower case.

# cat file30

testdd convert

dd command does not convert the file names, only its content.

**Conclusion**

The dd command, although not technically an archiving command, is similar in some ways because it can copy an entire partition or disk into a file and vice versa.