od command in Linux is used to output the contents of a file in different formats with the octal format being the default.

This command is especially useful when debugging Linux scripts for unwanted changes or characters.

This article explains how to use od command with some examples.  
  
The basic syntax of this command is :

od [OPTION]... [FILE]...

**1. Display contents of file in octal format using -b option**

The following is the input file used for this example:

$ cat input

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

Now execute od command on this input file.

$ od -b input

0000000 061 012 062 012 063 012 064 012 065 012 066 012 067 012 070 012

0000020 071 012 061 060 012 061 061 012 061 062 012 061 063 012 061 064

0000040 012 061 065 012 061 066 012 061 067 012 061 070 012 061 071 012

0000060 062 060 012

0000063

So we see that output was produced in octal format. The first column in the output of od represents the byte offset in file.

**2. Display contents of file in character format using -c option**

Using the same input file (as in example 1 above).

$ od -c input

0000000 1 \n 2 \n 3 \n 4 \n 5 \n 6 \n 7 \n 8 \n

0000020 9 \n 1 0 \n 1 1 \n 1 2 \n 1 3 \n 1 4

0000040 \n 1 5 \n 1 6 \n 1 7 \n 1 8 \n 1 9 \n

0000060 2 0 \n

0000063

So we see that the output was produced in the character format.

**3. Display the byte offsets in different formats using -A option**

The byte offset can be displayed in any of the following formats :

* Hexadecimal (using -x along with -A)
* Octal (using -o along with -A)
* Decimal (using -d along with -A)

The following are the examples of offsets in different formats :

$ od -Ax -c input

**000000** 1 \n 2 \n 3 \n 4 \n 5 \n 6 \n 7 \n 8 \n

**000010** 9 \n 1 0 \n 1 1 \n 1 2 \n 1 3 \n 1 4

**000020** \n 1 5 \n 1 6 \n 1 7 \n 1 8 \n 1 9 \n

**000030** 2 0 \n

**000033**

$ od -Ad -c input

**0000000** 1 \n 2 \n 3 \n 4 \n 5 \n 6 \n 7 \n 8 \n

**0000016** 9 \n 1 0 \n 1 1 \n 1 2 \n 1 3 \n 1 4

**0000032** \n 1 5 \n 1 6 \n 1 7 \n 1 8 \n 1 9 \n

**0000048** 2 0 \n

**0000051**

$ od -Ao -c input

**0000000** 1 \n 2 \n 3 \n 4 \n 5 \n 6 \n 7 \n 8 \n

**0000020** 9 \n 1 0 \n 1 1 \n 1 2 \n 1 3 \n 1 4

**0000040** \n 1 5 \n 1 6 \n 1 7 \n 1 8 \n 1 9 \n

**0000060** 2 0 \n

**0000063**

So we see that as per the input supplied to the -A option, the first column (that contains the byte offset) is displayed in different formats.

**4. Display no offset information using ‘-An’ option**

Consider the following example :

$ od -An -c input

1 \n 2 \n 3 \n 4 \n 5 \n 6 \n 7 \n 8 \n

9 \n 1 0 \n 1 1 \n 1 2 \n 1 3 \n 1 4

\n 1 5 \n 1 6 \n 1 7 \n 1 8 \n 1 9 \n

2 0 \n

So we see that byte offset related information was not displayed.

**5. Display output after skipping some bytes**

This is achieved by using -j option. Here is an example :

$ od -j9 -c input

0000011 \n 6 \n 7 \n 8 \n 9 \n 1 0 \n 1 1 \n 1

0000031 2 \n 1 3 \n 1 4 \n 1 5 \n 1 6 \n 1 7

0000051 \n 1 8 \n 1 9 \n 2 0 \n

0000063

If we compare the above output with the output in example 2, we can see that initial 9 bytes were skipped from output.

**6. Display limited bytes in output using -N option**

This is the opposite of the -j option discussed in example 5 above. Here is an example :

$ od -N9 -c input

0000000 1 \n 2 \n 3 \n 4 \n 5

0000011

So we see that only 9 bytes were displayed in the output.

**7. Display output as decimal integers using -i option**

Consider the following example :

$ od -i input

0000000 171051569 171182643 171313717 171444791

0000020 808520249 170995978 822751793 875629107

0000040 171258122 822752817 942737975 171520266

0000060 667698

0000063

If we combine -i with -b then its gives more information as to how decimal integers are displayed. Here is an example :

$ od -ib input

0000000 171051569 171182643 171313717 171444791

061 012 062 012 063 012 064 012 065 012 066 012 067 012 070 012

0000020 808520249 170995978 822751793 875629107

071 012 061 060 012 061 061 012 061 062 012 061 063 012 061 064

0000040 171258122 822752817 942737975 171520266

012 061 065 012 061 066 012 061 067 012 061 070 012 061 071 012

0000060 667698

062 060 012

0000063

So the above output shows how octal output is displayed as integer output.

**8. Display output as hexadecimal 2 byte units using -x option**

Consider the following example :

$ od -x input

0000000 0a31 0a32 0a33 0a34 0a35 0a36 0a37 0a38

0000020 0a39 3031 310a 0a31 3231 310a 0a33 3431

0000040 310a 0a35 3631 310a 0a37 3831 310a 0a39

0000060 3032 000a

0000063

So we see that the output was displayed in terms of hexadecimal 2 byte units.

**9. Display the contents as two byte octal units using -o option**

Consider the following example :

$ od -o input

0000000 005061 005062 005063 005064 005065 005066 005067 005070

0000020 005071 030061 030412 005061 031061 030412 005063 032061

0000040 030412 005065 033061 030412 005067 034061 030412 005071

0000060 030062 000012

0000063

Note that the od command displays the same output when run without any option. Here is an example:

$ od input

0000000 005061 005062 005063 005064 005065 005066 005067 005070

0000020 005071 030061 030412 005061 031061 030412 005063 032061

0000040 030412 005065 033061 030412 005067 034061 030412 005071

0000060 030062 000012

0000063

**10. Customize the output width using -w option**

Consider the following example :

$ od -w1 -c -Ad input

0000000 1

0000001 \n

0000002 2

0000003 \n

0000004 3

0000005 \n

0000006 4

0000007 \n

0000008 5

0000009 \n

0000010 6

0000011 \n

0000012 7

0000013 \n

0000014 8

0000015 \n

0000016 9

0000017 \n

0000018 1

0000019 0

0000020 \n

0000021 1

\*

0000023 \n

0000024 1

0000025 2

0000026 \n

0000027 1

0000028 3

0000029 \n

0000030 1

0000031 4

0000032 \n

0000033 1

0000034 5

0000035 \n

0000036 1

0000037 6

0000038 \n

0000039 1

0000040 7

0000041 \n

0000042 1

0000043 8

0000044 \n

0000045 1

0000046 9

0000047 \n

0000048 2

0000049 0

0000050 \n

0000051

So we see that output width was reduced to 1 in the above output.

**11. Output duplicates using -v option**

As can be observed in the output of example 10 above, a \* was printed. This is done to suppress the output of lines that are same or duplicates. But through -v option these lines can also be printed. Here is an example :

$ od -w1 -v -c -Ad input

0000000 1

0000001 \n

0000002 2

0000003 \n

0000004 3

0000005 \n

0000006 4

0000007 \n

0000008 5

0000009 \n

0000010 6

0000011 \n

0000012 7

0000013 \n

0000014 8

0000015 \n

0000016 9

0000017 \n

0000018 1

0000019 0

0000020 \n

**0000021 1**

**0000022 1**

0000023 \n

0000024 1

0000025 2

0000026 \n

0000027 1

0000028 3

0000029 \n

0000030 1

0000031 4

0000032 \n

0000033 1

0000034 5

0000035 \n

0000036 1

0000037 6

0000038 \n

0000039 1

0000040 7

0000041 \n

0000042 1

0000043 8

0000044 \n

0000045 1

0000046 9

0000047 \n

0000048 2

0000049 0

0000050 \n

0000051

**12. Accept input from command line using –**

Consider the following example :

$ od -c -

The Geek Stuff0000000 T h e G e e k S t u f f

0000016

So we see that first the input was given through stdin and then after pressing the ctrl+d a couple of times the od command output was displayed.

**13. Display hidden characters using od command**

Consider the following input :

The Geek ^MStuff

If a file containing the above string is printed using the cat command, following output is seen :

$ cat input

Stuffeek

But as you can see that this is not what exactly file contains.

Now lets use od command with -c option over this :

$ od -c input

0000000 T h e G e e k \r S t u f f \n

0000020

So od command clearly displays that a carriage return without a line feed was inserted between the strings due to which a messed up output was being shown by cat command.