NAME

hexdump, **hd** — ASCII, decimal, hexadecimal, octal dump

SYNOPSIS

hexdump [**-bcCdovx**] [**-e** *format_string*] [**-f** *format_file*] [**-n** *length*] [**-s** *skip*] *file ...*
hd [**-bcdovx**] [**-e** *format_string*] [**-f** *format_file*] [**-n** *length*] [**-s** *skip*] *file ...*

DESCRIPTION

The **hexdump** utility is a filter which displays the specified files, or the standard input, if no files are specified, in a user specified format.

The options are as follows:

-b *One-byte octal display.* Display the input offset in hexadecimal, followed by sixteen space-separated, three column, zero-filled, bytes of input data, in octal, per line.

-c *One-byte character display.* Display the input offset in hexadecimal, followed by sixteen space-separated, three column, space-filled, characters of input data per line.

-C *Canonical hex+ASCII display.* Display the input offset in hexadecimal, followed by sixteen space-separated, two column, hexadecimal bytes, followed by the same sixteen bytes in %_p format enclosed in "|" characters.

Calling the command **hd** implies this option.

-d *Two-byte decimal display.* Display the input offset in hexadecimal, followed by eight space-separated, five column, zero-filled, two-byte units of input data, in unsigned decimal, per line.

-e *format_string*
Specify a format string to be used for displaying data.

-f *format_file*
Specify a file that contains one or more newline separated format strings. Empty lines and lines whose first non-blank character is a hash mark (**#**) are ignored.

-n *length*
Interpret only *length* bytes of input.

-o *Two-byte octal display.* Display the input offset in hexadecimal, followed by eight space-separated, six column, zero-filled, two byte quantities of input data, in octal, per line.

-s *offset*
Skip *offset* bytes from the beginning of the input. By default, *offset* is interpreted as a decimal number. With a leading **0x** or **0X**, *offset* is in terpreted as a hexadecimal number, otherwise, with a leading **0**, *offset* is interpreted as an octal number. Appending the character **b**, **k**, or **m** to *offset* causes it to be interpreted as a multiple of `512`, `1024`, or `1048576`, respectively.

-v Cause **hexdump** to display all input data. Without the **-v** option, any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the input offsets), are replaced with a line comprised of a single asterisk.

-x *Two-byte hexadecimal display.* Display the input offset in hexadecimal, followed by eight, space separated, four column, zero-filled, two-byte quantities of input data, in hexadecimal, per line.

For each input file, **hexdump** sequentially copies the input to standard output, transforming the data according to the format strings specified by the **-e** and **-f** options, in the order that they were specified.

Formats

A format string contains any number of format units, separated by whitespace. A format unit contains up to three items: an iteration count, a byte count, and a format.

The iteration count is an optional positive integer, which defaults to one. Each format is applied iteration count times.

The byte count is an optional positive integer. If specified it defines the number of bytes to be interpreted by each iteration of the format.

If an iteration count and/or a byte count is specified, a single slash must be placed after the iteration count and/or before the byte count to disambiguate them. Any whitespace before or after the slash is ignored.

The format is required and must be surrounded by double quote ( ) marks. It is interpreted as a fprintf-style format string (see , -- fprintf(3)) with the following exceptions:

- An asterisk (*) may not be used as a field width or precision.
- A byte count or field precision *is* required for each "s" conversion character (unlike the `fprintf`(3) default which prints the entire string if the precision is unspecified).
- The conversion characters "%", "h", "l", "n", "p" and "q" are not supported.
- The single character escape sequences described in the C standard are supported:

  NUL 0
  <alert character> a
  <backspace> b
  <form-feed> f
  <newline> n
  <carriage return> r
  <tab> t
  <vertical tab> v

The **hexdump** utility also supports the following additional conversion strings:

_a[**dox**] Display the input offset, cumulative across input files, of the next byte to be displayed. The appended characters **d**, **o**, and **x** specify the display base as decimal, octal or hexadecimal respectively.

_A[**dox**] Identical to the **_a** conversion string except that it is only performed once, when all of the input data has been processed.

_c Output characters in the default character set. Nonprinting characters are displayed in three character, zero-padded octal, except for those representable by standard escape notation (see above), which are displayed as two character strings.

_p Output characters in the default character set. Nonprinting characters are displayed as a single "**.**".

_u Output US ASCII characters, with the exception that control characters are displayed using the following, lower-case, names. Characters greater than 0xff, hexadecimal, are displayed as hexadecimal strings.

000 NUL001 SOH 002 STX 003 ETX004 EOT005 ENQ
006 ACK007 BEL008 BS 009 HT 00A LF 00B VT
00C FF 00D CR 00E SO 00F SI 010 DLE011 DC1
012 DC2 013 DC3014 DC4 015 NAK016 SYN017 ETB
018 CAN019 EM 01A SUB 01B ESC01C FS01D GS
01E RS 01F US 07F DEL

The default and supported byte counts for the conversion characters are as follows:

%_c, %_p, %_u, %c	One byte counts only.
%d, %i, %o, %u, %X, %x	Four byte default, one, two and four byte counts supported.
%E, %e, %f, %G, %g	Eight byte default, four and twelve byte counts supported.

The amount of data interpreted by each format string is the sum of the data required by each format unit, which is the iteration count times the byte count, or the iteration count times the number of bytes required by the format if the byte count is not specified.

The input is manipulated in ‘‘blocks’’, where a block is defined as the largest amount of data specified by any format string. Format strings interpreting less than an input block’s worth of data, whose last format unit both interprets some number of bytes and does not have a specified iteration count, have the iteration count incremented until the entire input block has been processed or there is not enough data remaining in the block to satisfy the format string.

If, either as a result of user specification or **hexdump** modifying the iteration count as described above, an iteration count is greater than one, no trailing whitespace characters are output during the last iteration.

It is an error to specify a byte count as well as multiple conversion characters or strings unless all but one of the conversion characters or strings is **_a** or **_A**.

If, as a result of the specification of the **-n** option or end-of-file being reac hed, input data only partially satisfies a format string, the input block is zero-padded sufficiently to display all available data (i.e., any format units overlapping the end of data will display some number of the zero bytes).

Further output by such format strings is replaced by an equivalent number of spaces. An equivalent number of spaces is defined as the number of spaces output by an **s** conversion character with the same field width and precision as the original conversion character or conversion string but with any “+”, “ ”, “#” conversion flag characters removed, and referencing a NULL string.

If no format strings are specified, the default display is equivalent to specifying the **-x** option.

EXIT STATUS

The **hexdump** and **hd** utilities exit 0 on success, and >0 if an error occurs.

EXAMPLES

Display the input in perusal format:

```
"%06.6_ao " 12/1 "%3_u "
"\t\t" "%_p "
"\n"
```

Implement the -x option:

```
"%07.7_Ax\n"
"%07.7_ax " 8/2 "%04x " "\n"
```

Some examples for the -e option:

```
# hex bytes
% echo hello | hexdump -v -e '/1 "%02X "' ; echo
68 65 6C 6C 6F 0A

# same, with ASCII section
% echo hello | hexdump -e '8/1 "%02X ""\t"" "' -e '8/1 "%c""\n"'
68 65 6C 6C 6F 0A hello

# hex with preceding 'x'
% echo hello | hexdump -v -e '"x" 1/1 "%02X" " "' ; echo
x68 x65 x6C x6C x6F x0A

# one hex byte per line
% echo hello | hexdump -v -e '/1 "%02X\n"'
68
65
6C
6C
6F
0A

# a table of byte#, hex, decimal, octal, ASCII
% echo hello | hexdump -v -e '/1 "%_ad# "' -e '/1 "%02X hex"' -e '/1 " = %03i dec"' -e '/
0# 68 hex = 104 dec = 150 oct = _h_
1# 65 hex = 101 dec = 145 oct = _e_
2# 6C hex = 108 dec = 154 oct = _l_
3# 6C hex = 108 dec = 154 oct = _l_
4# 6F hex = 111 dec = 157 oct = _o_
5# 0A hex = 010 dec = 012 oct = _
_

# byte# & ASCII with control chars
% echo hello | hexdump -v -e '/1 "%_ad# "' -e '/1 " _%_u\_\n"'
0# _h_
1# _e_
2# _l_
3# _l_
4# _o_
5# _lf_
```

SEE ALSO

gdb(1), od(1)