NAME

basenc, **base64**, **base64url**, **base32**, **base32hex**, **base16**, **base2msbf**, **base2lsbf**, **z85** — transcode RFC 4648, binary, or ZeroMQ data

SYNOPSIS

base64 [**−w** *wrap*] [*file*]
base64url [**−w** *wrap*] [*file*]
base32 [**−w** *wrap*] [*file*]
base32hex [**−w** *wrap*] [*file*]
base16 [**−w** *wrap*] [*file*]
base2msbf [**−w** *wrap*] [*file*]
base2lsbf [**−w** *wrap*] [*file*]
z85 [**−w** *wrap*] [*file*]
basenc −−{**base64**|**base64url**|**base32**|**base32hex**|**base16**|**base2msbf**|**base2lsbf**|**z85**}
    [**−w** *wrap*] [*file*]

base64 −d [**−i**] [*file*]
base64url −d [**−i**] [*file*]
base32 −d [**−i**] [*file*]
base32hex −d [**−i**] [*file*]
base16 −d [**−i**] [*file*]
base2msbf −d [**−i**] [*file*]
base2lsbf −d [**−i**] [*file*]
z85 −d [**−i**] [*file*]
basenc −−{**base64**|**base64url**|**base32**|**base32hex**|**base16**|**base2msbf**|**base2lsbf**|**z85**}
    −d [**−i**] [*file*]

DESCRIPTION

Without **−d**, encode `file` (standard input stream if "**-**", the default), mapping consecutive bits to the indices into one of the following RFC 4648 alphabets:

base64	`ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/`
base64url	`ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_`
base32	`ABCDEFGHIJKLMNOPQRSTUVWXYZ234567`
base32hex	`0123456789ABCDEFGHIJKLMNOPQRSTUV`
base16	`0123456789ABCDEF`

If the input is not long enough ($\frac{8 len(input)}{\log_2(len(alphabet))}$ is not an integer), it's padded with null bytes, expressed as "**=**"s.

base2?sbf use a different algorithm:

base2msbf	yields the bits of each byte (as `01`) in natural (most-significant-bit-first) order
base2lsbf	likewise but in reverse (least-significant-bit-first) order

z85 uses the ZeroMQ 32/Z85 algorithm, mapping consecutive **4**-byte chunks onto **5** output bytes by casting them to big-endian 32-bit unsigned integers, then repeatedly dividing by **85**, with the remainders used as an index into the alphabet (output in reverse order):

z85	`0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.-:+=^!/*?&<>()[]{}@%$#`

Non-**4**-byte-multiple input is a hard error.

If `wrap` (default **76**) is non-zero, newlines are inserted every `wrap` bytes, as well as a final one if the output wasn't empty.

With **−d**, decode it, mapping input bytes from the alphabet to their indices' bits. Padding, if any, is required to decode the full message.

base2?sbf decode in the expected way. **z85** does the inverse operation, treating each **5** bytes as a base-**85** big-endian number, and outputting the resulting integer in big-endian order. The encoded input not being a multiple of **8** or **5** is a hard error.

If bytes from outside the alphabet (except the newline) are encountered and **-i** wasn't specified, a diagnostic is issued.

By definition, sequentially applying either two inverse transformations yields the same data: encoding into any of these alphabets, carefully composed of bytes which universally have no special meaning, allows lossless transmission of binary data as plain text at only a minor increase in size, equal to the amount of alphabet text required to express one byte of input: $\frac{8}{\log_2(len(alphabet))} \Rightarrow$ **4/3=1.(3)**, **1.6**, **2**, **8** and **1.25** respectively.

base16 is equivalent to a hexadecimal listing and **base2msbf** to a binary one. **base32hex** has the useful property of retaining the sort ordering of the input (i.e. **base32hex**(data) < **base32hex**(dat2) ⇔ data < dat2). The **base64url** alphabet is safe to use in paths (and URLs), and may be converted into from the **base64** alphabet by a simple `tr '+/' '-_'` invocation.

OPTIONS

-d, **--decode**	Decode the input.
-i, **--ignore-garbage**	Don't produce diagnostics and return successfully when a non-alphabet byte is encountered while decoding.
-w, **--wrap**=*wrap*	Wrap the encoded output at *wrap* columns and terminate it with a newline. Defaults to **76**.

basenc requires a **--** *algorithm* flag to select the algorithm, but is otherwise equivalent to invoking *algorithm* directly with the same arguments.

EXIT STATUS

1 if *file* couldn't be opened or read, contained garbage and **-i** wasn't specified, or didn't contain a multiple of **4** bytes in **z85** mode.

SEE ALSO

RFC 4648: **https://tools.ietf.org/html/4648**
ZeroMQ 32/Z85: **https://rfc.zeromq.org/spec/32/Z85**

STANDARDS

Compatible with the GNU system, which only contains **base64**, **base32**, and **basenc**. A compatible **base64** also appears in NetBSD 9.0.

HISTORY

4BSD introduced uuencode(1C) (and uudecode(1C)): a structured, whole-file approach including the name and permissions in the encoded output, also using a 6-bits-per-byte encoding (but a fundamentally different one), an enhancement to the uucp(1)/uusend(1) suite. IEEE Std 1003.1-2001 ("POSIX.1") added **-m**, using the **basenc** encoding instead.

coreutils 6.1 (2006-08-19) adds **base64**, largely as present-day. **base32** joins it in coreutils 8.25 (2016-01-20), and **basenc** in coreutils 8.31 (2019-03-10).