



Linux Ubuntu 26.04 Manual Pages on command 'iconv.3'

\$ man iconv.3

iconv(3) Library Functions Manual iconv(3)

NAME

iconv - perform character set conversion

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <iconv.h>

size_t iconv(iconv_t cd,
             char **restrict inbuf, size_t *restrict inbytesleft,
             char **restrict outbuf, size_t *restrict outbytesleft);
```

DESCRIPTION

The `iconv()` function converts a sequence of characters in one character encoding to a sequence of characters in another character encoding. The `cd` argument is a conversion descriptor, previously created by a call to `iconv_open(3)`; the conversion descriptor defines the character encodings that `iconv()` uses for the conversion. The `inbuf` argument is the address of a variable that points to the first character of the input sequence; `inbytesleft` indicates the number of bytes in that buffer. The `outbuf` argument is the address of a variable that points to the first byte available in the output buffer; `outbytesleft` indicates the number of bytes available in the output buffer.

The main case is when `inbuf` is not `NULL` and `*inbuf` is not `NULL`. In this case, the `iconv()` function converts the multibyte sequence starting at `*inbuf` to a multibyte sequence starting at `*outbuf`. At most `*inbytesleft` bytes, starting at `*inbuf`, will be read. At most `*outbytesleft` bytes, starting at `*outbuf`, will be written.

The `iconv()` function converts one multibyte character at a time, and for each character conversion it increments `*inbuf` and decrements `*inbytesleft` by the number of converted input bytes, it increments `*outbuf` and decrements `*outbytesleft` by the number of converted output bytes, and it updates the conversion state contained in `cd`. If the character encoding of the input is stateful, the `iconv()` function can also convert a sequence of input bytes to an update to the conversion state without producing any output bytes; such input is called a shift sequence. The conversion can stop for five reasons:

- ? An invalid multibyte sequence is encountered in the input. In this case, it sets `errno` to `EILSEQ` and returns `(size_t) -1`. `*inbuf` is left pointing to the beginning of the invalid multibyte sequence.
- ? A multibyte sequence is encountered that is valid but that cannot be translated to the character encoding of the output. This condition depends on the implementation and on the conversion descriptor. In the GNU C library and GNU `libiconv`, if `cd` was created without the suffix `//TRANSLIT` or `//IGNORE`, the conversion is strict: lossy conversions produce this condition. If the suffix `//TRANSLIT` was specified, transliteration can avoid this condition in some cases. In the `musl` C library, this condition cannot occur because a conversion to `'*'` is used as a fallback. In the `FreeBSD`, `NetBSD`, and `Solaris` implementations of `iconv()`, this condition cannot occur either, because a conversion to `'?'` is used as a fallback. When this condition is met, `iconv()` sets `errno` to `EILSEQ` and returns `(size_t) -1`. `*inbuf` is left pointing to the beginning of the unconvertible multibyte sequence.
- ? The input byte sequence has been entirely converted, that is, `*inbytesleft` has gone down to 0. In this case, `iconv()` returns the number of nonreversible conversions performed during this call.
- ? An incomplete multibyte sequence is encountered in the input, and the input byte sequence terminates after it. In this case, it sets `errno` to `EINVAL` and returns `(size_t) -1`. `*inbuf` is left pointing to the beginning of the incomplete multibyte sequence.
- ? The output buffer has no more room for the next converted character. In this case, it sets `errno` to `E2BIG` and returns `(size_t) -1`.

A different case is when `inbuf` is `NULL` or `*inbuf` is `NULL`, but `outbuf` is not `NULL` and `*outbuf` is not `NULL`. In this case, the `iconv()` function attempts to set `cd`'s conversion state to the initial state and store a corresponding shift sequence at `*outbuf`. At most `*outbytesleft` bytes, starting at `*outbuf`, will be written. If the output buffer has no more

room for this reset sequence, it sets `errno` to `E2BIG` and returns `(size_t)-1`. Otherwise, it increments `*outbuf` and decrements `*outbytesleft` by the number of bytes written.

A third case is when `inbuf` is `NULL` or `*inbuf` is `NULL`, and `outbuf` is `NULL` or `*outbuf` is `NULL`.

In this case, the `iconv()` function sets `cd`'s conversion state to the initial state.

RETURN VALUE

The `iconv()` function returns the number of characters converted in a nonreversible way during this call; reversible conversions are not counted. In case of error, `iconv()` returns `(size_t)-1` and sets `errno` to indicate the error.

ERRORS

The following errors can occur, among others:

E2BIG There is not sufficient room at *outbuf.

EILSEQ An invalid multibyte sequence has been encountered in the input.

EINVAL An incomplete multibyte sequence has been encountered in the input.

ATTRIBUTES

For an explanation of the terms used in this section, see `attributes(7)`.

[illegible]

? Interface	? Attribute	? Value	?
-------------	-------------	---------	---

[illegible]

? iconv() ? Thread safety ? MT-Safe race:cd ?

[illegible]

The `iconv()` function is MT-Safe, as long as callers arrange for mutual exclusion on the `cd` argument.

STANDARDS

POSIX.1-2008.

HISTORY

glibc 2.1. POSIX.1-2001.

NOTES

In each series of calls to `iconv()`, the last should be one with `inbuf` or `*inbuf` equal to `NULL`, in order to flush out any partially converted input.

Although `inbuf` and `outbuf` are typed as `char **`, this does not mean that the objects they point can be interpreted as C strings or as arrays of characters: the interpretation of character byte sequences is handled internally by the conversion functions. In some encodings, a zero byte may be a valid part of a multibyte character.

The caller of `iconv()` must ensure that the pointers passed to the function are suitable for accessing characters in the appropriate character set. This includes ensuring correct alignment on platforms that have tight restrictions on alignment.

SEE ALSO

`iconv_close(3)`, `iconv_open(3)`, `iconvconfig(8)`

Linux man-pages 6.17

2026-02-08

`iconv(3)`