NAME

iconv − character set conversion

SYNOPSIS

iconv [**-c**] [**-s**] [**-f** *encoding*] [**-t** *encoding*] [*inputfile* ...]
iconv **-l**

DESCRIPTION

The **iconv** program converts text from one encoding to another encoding. More precisely, it converts **from** the encoding given for the **-f** option **to** the encoding given for the **-t** option. Either of these encodings defaults to the encoding of the current locale. All the *inputfile*s are read and converted in turn; if no *inputfile* is given, the standard input is used. The converted text is printed to standard output.

When option **-c** is given, characters that cannot be converted are silently discarded, instead of leading to a conversion error.

When option **-s** is given, error messages about invalid or unconvertible characters are omitted, but the actual converted text is unaffected.

The encodings permitted are system dependent. For the libiconv implementation, they are listed in the iconv_open(3) manual page.

The **iconv -l** command lists the names of the supported encodings, in a system dependent format. For the libiconv implementation, the names are printed in upper case, separated by whitespace, and alias names of an encoding are listed on the same line as the encoding itself.

SEE ALSO

iconv_open(3), **locale**(7)

NAME

iconv − perform character set conversion

SYNOPSIS

#include <iconv.h>

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

DESCRIPTION

The argument *cd* must be a conversion descriptor created using the function **iconv_open**.

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*. The conversion can stop for four reasons:

1. 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.

2. The input byte sequence has been entirely converted, i.e. **inbytesleft* has gone down to 0. In this case **iconv** returns the number of non-reversible conversions performed during this call.

3. 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.

4. 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 non-reversible way during this call; reversible conversions are not counted. In case of error, it sets **errno** and returns (size_t)(-1).

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.

CONFORMING TO

UNIX98

SEE ALSO

iconv_open(3), **iconv_close**(3)

NAME

iconv_close − deallocate descriptor for character set conversion

SYNOPSIS

#include <iconv.h>

int iconv_close (iconv_t *cd***);**

DESCRIPTION

The **iconv_close** function deallocates a conversion descriptor *cd* previously allocated using **iconv_open**.

RETURN VALUE

When successful, the **iconv_close** function returns 0. In case of error, it sets **errno** and returns -1.

CONFORMING TO

UNIX98

SEE ALSO

iconv_open(3), **iconv**(3)

NAME

iconv_open − allocate descriptor for character set conversion

SYNOPSIS

#include <iconv.h>

iconv_t iconv_open (const char* *tocode***, const char*** *fromcode***);**

DESCRIPTION

The **iconv_open** function allocates a conversion descriptor suitable for converting byte sequences from character encoding *fromcode* to character encoding *tocode*.

The values permitted for *fromcode* and *tocode* and the supported combinations are system dependent. For the libiconv library, the following encodings are supported, in all combinations.

European languages
: ASCII, ISO-8859-{1,2,3,4,5,7,9,10,13,14,15,16}, KOI8-R, KOI8-U, KOI8-RU, CP{1250,1251,1252,1253,1254,1257}, CP{850,866}, Mac{Roman,CentralEurope,Iceland,Croatian,Romania}, Mac{Cyrillic,Ukraine,Greek,Turkish}, Macintosh

Semitic languages
: ISO-8859-{6,8}, CP{1255,1256}, CP862, Mac{Hebrew,Arabic}

Japanese
: EUC-JP, SHIFT_JIS, CP932, ISO-2022-JP, ISO-2022-JP-2, ISO-2022-JP-1

Chinese
: EUC-CN, HZ, GBK, GB18030, EUC-TW, BIG5, CP950, BIG5-HKSCS, ISO-2022-CN, ISO-2022-CN-EXT

Korean
: EUC-KR, CP949, ISO-2022-KR, JOHAB

Armenian
: ARMSCII-8

Georgian
: Georgian-Academy, Georgian-PS

Tajik
: KOI8-T

Thai
: TIS-620, CP874, MacThai

Laotian
: MuleLao-1, CP1133

Vietnamese
: VISCII, TCVN, CP1258

Platform specifics
: HP-ROMAN8, NEXTSTEP

Full Unicode
: UTF-8
UCS-2, UCS-2BE, UCS-2LE
UCS-4, UCS-4BE, UCS-4LE
UTF-16, UTF-16BE, UTF-16LE
UTF-32, UTF-32BE, UTF-32LE
UTF-7
C99, JAVA

Full Unicode, in terms of **uint16_t** or **uint32_t**
: (with machine dependent endianness and alignment)
UCS-2-INTERNAL, UCS-4-INTERNAL

Locale dependent, in terms of **char** or **wchar_t**
(with machine dependent endianness and alignment, and with semantics depending on the OS and the current LC_CTYPE locale facet)
char, wchar_t

When configured with the option **−−enable-extra-encodings**, it also provides support for a few extra encodings:

European languages
CP{437,737,775,852,853,855,857,858,860,861,863,865,869,1125}

Semitic languages
CP864

Japanese
EUC-JISX0213, Shift_JISX0213, ISO-2022-JP-3

Turkmen
TDS565

Platform specifics
RISCOS-LATIN1

The empty encoding name "" is equivalent to "char": it denotes the locale dependent character encoding.

When the string "//TRANSLIT" is appended to *tocode*, transliteration is activated. This means that when a character cannot be represented in the target character set, it can be approximated through one or several similarly looking characters.

When the string "//IGNORE" is appended to *tocode*, characters that cannot be represented in the target character set will be silently discarded.

The resulting conversion descriptor can be used with **iconv** any number of times. It remains valid until deallocated using **iconv_close**.

A conversion descriptor contains a conversion state. After creation using **iconv_open**, the state is in the initial state. Using **iconv** modifies the descriptor's conversion state. (This implies that a conversion descriptor can not be used in multiple threads simultaneously.) To bring the state back to the initial state, use **iconv** with NULL as *inbuf* argument.

RETURN VALUE

The **iconv_open** function returns a freshly allocated conversion descriptor. In case of error, it sets **errno** and returns (iconv_t)(-1).

ERRORS

The following error can occur, among others:

EINVAL
The conversion from *fromcode* to *tocode* is not supported by the implementation.

CONFORMING TO

UNIX98

SEE ALSO

iconv(3), **iconv_close**(3)