Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'iconv.3p' command

$ man iconv.3p

ICONV(3P) POSIX Programmer's Manual ICONV(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

iconv ? codeset conversion function

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 shall convert the sequence of characters from one codeset, in the array specified by inbuf, into a sequence of corre? sponding characters in another codeset, in the array specified by out? buf. The codesets are those specified in the iconv_open() call that returned the conversion descriptor, cd. The inbuf argument points to a variable that points to the first character in the input buffer and in? bytesleft indicates the number of bytes to the end of the buffer to be converted. The outbuf argument points to a variable that points to the first available byte in the output buffer and outbytesleft indicates

the number of the available bytes to the end of the buffer.

For state-dependent encodings, the conversion descriptor cd is placed into its initial shift state by a call for which inbuf is a null pointer, or for which inbuf points to a null pointer. When iconv() is called in this way, and if outbuf is not a null pointer or a pointer to a null pointer, and outbytesleft points to a positive value, iconv() shall place, into the output buffer, the byte sequence to change the output buffer to its initial shift state. If the output buffer is not large enough to hold the entire reset sequence, iconv() shall fail and set errno to [E2BIG]. Subsequent calls with inbuf as other than a null pointer or a pointer to a null pointer cause the conversion to take place from the current state of the conversion descriptor.

If a sequence of input bytes does not form a valid character in the specified codeset, conversion shall stop after the previous success? fully converted character. If the input buffer ends with an incomplete character or shift sequence, conversion shall stop after the previous successfully converted bytes. If the output buffer is not large enough to hold the entire converted input, conversion shall stop just prior to the input bytes that would cause the output buffer to overflow. The variable pointed to by inbuf shall be updated to point to the byte fol? lowing the last byte successfully used in the conversion. The value pointed to by inbytesleft shall be decremented to reflect the number of bytes still not converted in the input buffer. The variable pointed to by outbuf shall be updated to point to the byte following the last byte of converted output data. The value pointed to by outbytesleft shall be decremented to reflect the number of bytes still available in the out? put buffer. For state-dependent encodings, the conversion descriptor shall be updated to reflect the shift state in effect at the end of the last successfully converted byte sequence.

If iconv() encounters a character in the input buffer that is valid, but for which an identical character does not exist in the target code? set, iconv() shall perform an implementation-defined conversion on this character.

RETURN VALUE

The iconv() function shall update the variables pointed to by the argu? ments to reflect the extent of the conversion and return the number of non-identical conversions performed. If the entire string in the input buffer is converted, the value pointed to by inbytesleft shall be 0. If the input conversion is stopped due to any conditions mentioned above, the value pointed to by inbytesleft shall be non-zero and errno shall be set to indicate the condition. If an error occurs, iconv() shall re? turn (size_t)-1 and set errno to indicate the error.

ERRORS

The iconv() function shall fail if:

EILSEQ Input conversion stopped due to an input byte that does not be? long to the input codeset.

E2BIG Input conversion stopped due to lack of space in the output buf? fer.

EINVAL Input conversion stopped due to an incomplete character or shift sequence at the end of the input buffer.

The iconv() function may fail if:

EBADF The cd argument is not a valid open conversion descriptor.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The inbuf argument indirectly points to the memory area which contains the conversion input data. The outbuf argument indirectly points to the memory area which is to contain the result of the conversion. The ob? jects indirectly pointed to by inbuf and outbuf are not restricted to containing data that is directly representable in the ISO C standard language char data type. The type of inbuf and outbuf, char **, does not imply that the objects pointed to are interpreted as null-termi? nated C strings or arrays of characters. Any interpretation of a byte sequence that represents a character in a given character set encoding scheme is done internally within the codeset converters. For example,

the area pointed to indirectly by inbuf and/or outbuf can contain all zero octets that are not interpreted as string terminators but as coded character data according to the respective codeset encoding scheme. The type of the data (char, short, long, and so on) read or stored in the objects is not specified, but may be inferred for both the input and output data by the converters determined by the fromcode and tocode ar? guments of iconv_open().

Regardless of the data type inferred by the converter, the size of the remaining space in both input and output objects (the intbytesleft and outbytesleft arguments) is always measured in bytes.

For implementations that support the conversion of state-dependent en? codings, the conversion descriptor must be able to accurately reflect the shift-state in effect at the end of the last successful conversion.

It is not required that the conversion descriptor itself be updated, which would require it to be a pointer type. Thus, implementations are free to implement the descriptor as a handle (other than a pointer type) by which the conversion information can be accessed and updated.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

iconv_open(), iconv_close(), mbsrtowcs()

The Base Definitions volume of POSIX.1?2017, <iconv.h>

at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .