NAME

iconv_open, **iconv_open_into**, **iconv_close**, **iconv** - codeset conversion functions

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <iconv.h>

iconv_t
iconv_open(*const char *dstname, const char *srcname*);

int
iconv_open_into(*const char *dstname, const char *srcname, iconv_allocation_t *ptr*);

int
iconv_close(*iconv_t cd*);

size_t
iconv(*iconv_t cd, char ** restrict src, size_t * restrict srcleft, char ** restrict dst, size_t * restrict dstleft*);

size_t
__iconv(*iconv_t cd, char ** restrict src, size_t * restrict srcleft, char ** restrict dst,*
*size_t * restrict dstleft, uint32_t flags, size_t * invalids*);

DESCRIPTION

The **iconv_open**() function opens a converter from the codeset *srcname* to the codeset *dstname* and returns its descriptor. The arguments *srcname* and *dstname* accept "" and "char", which refer to the current locale encoding.

The **iconv_open_into**() creates a conversion descriptor on a preallocated space. The *iconv_allocation_t* is used as a spaceholder type when allocating such space. The *dstname* and *srcname* arguments are the same as in the case of **iconv_open**(). The *ptr* argument is a pointer of *iconv_allocation_t* to the preallocated space.

The **iconv_close**() function closes the specified converter *cd*.

The **iconv**() function converts the string in the buffer **src* of length **srcleft* bytes and stores the converted string in the buffer **dst* of size **dstleft* bytes. After calling **iconv**(), the values pointed to by *src*, *srcleft*, *dst*, and *dstleft* are updated as follows:

*src Pointer to the byte just after the last character fetched.

*srcleft Number of remaining bytes in the source buffer.

*dst Pointer to the byte just after the last character stored.

*dstleft Number of remainder bytes in the destination buffer.

If the string pointed to by **src* contains a byte sequence which is not a valid character in the source codeset, the conversion stops just after the last successful conversion. If the output buffer is too small to store the converted character, the conversion also stops in the same way. In these cases, the values pointed to by *src*, *srcleft*, *dst*, and *dstleft* are updated to the state just after the last successful conversion.

If the string pointed to by **src* contains a character which is valid under the source codeset but can not be converted to the destination codeset, the character is replaced by an "invalid character" which depends on the destination codeset, e.g., '?', and the conversion is continued. **iconv**() returns the number of such "invalid conversions".

There are two special cases of **iconv**():

src == NULL || *src == NULL
: If the source and/or destination codesets are stateful, **iconv**() places these into their initial state.

  If both *dst* and **dst* are non-NULL, **iconv**() stores the shift sequence for the destination switching to the initial state in the buffer pointed to by **dst*. The buffer size is specified by the value pointed to by *dstleft* as above. **iconv**() will fail if the buffer is too small to store the shift sequence.

  On the other hand, *dst* or **dst* may be NULL. In this case, the shift sequence for the destination switching to the initial state is discarded.

The **__iconv**() function works just like **iconv**() but if **iconv**() fails, the invalid character count is lost there. This is a not bug rather a limitation of IEEE Std 1003.1-2008 ("POSIX.1"), so **__iconv**() is provided as an alternative but non-standard interface. It also has a flags argument, where currently the following flags can be passed:

__ICONV_F_HIDE_INVALID
: Skip invalid characters, instead of returning with an error.

RETURN VALUES

Upon successful completion of **iconv_open**(), it returns a conversion descriptor. Otherwise,

iconv_open() returns (iconv_t)-1 and sets errno to indicate the error.

Upon successful completion of **iconv_open_into**(), it returns 0. Otherwise, **iconv_open_into**() returns -1, and sets errno to indicate the error.

Upon successful completion of **iconv_close**(), it returns 0. Otherwise, **iconv_close**() returns -1 and sets errno to indicate the error.

Upon successful completion of **iconv**(), it returns the number of "invalid" conversions. Otherwise, **iconv**() returns (size_t)-1 and sets errno to indicate the error.

ERRORS

The **iconv_open**() function may cause an error in the following cases:

[ENOMEM] Memory is exhausted.

[EINVAL] There is no converter specified by *srcname* and *dstname*.

The **iconv_open_into**() function may cause an error in the following cases:

[EINVAL] There is no converter specified by *srcname* and *dstname*.

The **iconv_close**() function may cause an error in the following case:

[EBADF] The conversion descriptor specified by *cd* is invalid.

The **iconv**() function may cause an error in the following cases:

[EBADF] The conversion descriptor specified by *cd* is invalid.

[EILSEQ] The string pointed to by **src* contains a byte sequence which does not describe a valid character of the source codeset.

[E2BIG] The output buffer pointed to by **dst* is too small to store the result string.

[EINVAL] The string pointed to by **src* terminates with an incomplete character or shift sequence.

SEE ALSO

iconv(1), mkcsmapper(1), mkesdb(1), __iconv_get_list(3), iconv_canonicalize(3), iconvctl(3), iconvlist(3)

STANDARDS

The **iconv_open**(), **iconv_close**(), and **iconv**() functions conform to IEEE Std 1003.1-2008 ("POSIX.1").

The **iconv_open_into**() function is a GNU-specific extension and it is not part of any standard, thus its use may break portability. The **__iconv**() function is an own extension and it is not part of any standard, thus its use may break portability.