NAME

bindtextdomain − set directory containing message catalogs

SYNOPSIS

#include <libintl.h>

char * bindtextdomain (const char * *domainname***, const char *** *dirname***);**

DESCRIPTION

The **bindtextdomain** function sets the base directory of the hierarchy containing message catalogs for a given message domain.

A message domain is a set of translatable *msgid* messages. Usually, every software package has its own message domain. The need for calling **bindtextdomain** arises because packages are not always installed with the same prefix as the <libintl.h> header and the libc/libintl libraries.

Message catalogs will be expected at the pathnames *dirname*/*locale*/*category*/*domainname*.mo, where *locale* is a locale name and *category* is a locale facet such as **LC_MESSAGES**.

domainname must be a non-empty string.

If *dirname* is not NULL, the base directory for message catalogs belonging to domain *domainname* is set to *dirname*. The function makes copies of the argument strings as needed. If the program wishes to call the **chdir** function, it is important that *dirname* be an absolute pathname; otherwise it cannot be guaranteed that the message catalogs will be found.

If *dirname* is NULL, the function returns the previously set base directory for domain *domainname*.

RETURN VALUE

If successful, the **bindtextdomain** function returns the current base directory for domain *domainname*, after possibly changing it. The resulting string is valid until the next **bindtextdomain** call for the same *domainname* and must not be modified or freed. If a memory allocation failure occurs, it sets **errno** to **ENOMEM** and returns NULL.

ERRORS

The following error can occur, among others:

ENOMEM

Not enough memory available.

BUGS

The return type ought to be **const char ***, but is **char *** to avoid warnings in C code predating ANSI C.

SEE ALSO

gettext(3), **dgettext**(3), **dcgettext**(3), **ngettext**(3), **dngettext**(3), **dcngettext**(3), **textdomain**(3), **realpath**(3)

NAME

bind_textdomain_codeset − set encoding of message translations

SYNOPSIS

#include <libintl.h>

char * bind_textdomain_codeset (const char * *domainname***,**
const char * *codeset***);**

DESCRIPTION

The **bind_textdomain_codeset** function sets the output codeset for message catalogs for domain *domainname*.

A message domain is a set of translatable *msgid* messages. Usually, every software package has its own message domain.

By default, the **gettext** family of functions returns translated messages in the locale's character encoding, which can be retrieved as **nl_langinfo(CODESET)**. The need for calling **bind_textdomain_codeset** arises for programs which store strings in a locale independent way (e.g. UTF-8) and want to avoid an extra character set conversion on the returned translated messages.

domainname must be a non-empty string.

If *codeset* is not NULL, it must be a valid encoding name which can be used for the **iconv_open** function. The **bind_textdomain_codeset** function sets the output codeset for message catalogs belonging to domain *domainname* to *codeset*. The function makes copies of the argument strings as needed.

If *codeset* is NULL, the function returns the previously set codeset for domain *domainname*. The default is NULL, denoting the locale's character encoding.

RETURN VALUE

If successful, the **bind_textdomain_codeset** function returns the current codeset for domain *domainname*, after possibly changing it. The resulting string is valid until the next **bind_textdomain_codeset** call for the same *domainname* and must not be modified or freed. If a memory allocation failure occurs, it sets **errno** to **ENOMEM** and returns NULL. If no codeset has been set for domain *domainname*, it returns NULL.

ERRORS

The following error can occur, among others:

ENOMEM
Not enough memory available.

BUGS

The return type ought to be **const char ***, but is **char *** to avoid warnings in C code predating ANSI C.

SEE ALSO

gettext(3), **dgettext**(3), **dcgettext**(3), **ngettext**(3), **dngettext**(3), **dcngettext**(3), **textdomain**(3), **nl_langinfo**(3), **iconv_open**(3)

NAME

gettext, dgettext, dcgettext − translate message

SYNOPSIS

#include <libintl.h>

char * gettext (const char * *msgid***);**
char * dgettext (const char * *domainname***, const char *** *msgid***);**
char * dcgettext (const char * *domainname***, const char *** *msgid***,**
int *category***);**

DESCRIPTION

The **gettext**, **dgettext** and **dcgettext** functions attempt to translate a text string into the user's native language, by looking up the translation in a message catalog.

The *msgid* argument identifies the message to be translated. By convention, it is the English version of the message, with non-ASCII characters replaced by ASCII approximations. This choice allows the translators to work with message catalogs, called PO files, that contain both the English and the translated versions of each message, and can be installed using the **msgfmt** utility.

A message domain is a set of translatable *msgid* messages. Usually, every software package has its own message domain. The domain name is used to determine the message catalog where the translation is looked up; it must be a non-empty string. For the **gettext** function, it is specified through a preceding **textdomain** call. For the **dgettext** and **dcgettext** functions, it is passed as the *domainname* argument; if this argument is NULL, the domain name specified through a preceding **textdomain** call is used instead.

Translation lookup operates in the context of the current locale. For the **gettext** and **dgettext** functions, the **LC_MESSAGES** locale facet is used. It is determined by a preceding call to the **setlocale** function. **setlocale(LC_ALL,"")** initializes the **LC_MESSAGES** locale based on the first nonempty value of the three environment variables **LC_ALL**, **LC_MESSAGES**, **LANG**; see **setlocale**(3). For the **dcgettext** function, the locale facet is determined by the *category* argument, which should be one of the **LC_xxx** constants defined in the <locale.h> header, excluding **LC_ALL**. In both cases, the functions also use the **LC_CTYPE** locale facet in order to convert the translated message from the translator's codeset to the current locale's codeset, unless overridden by a prior call to the **bind_textdomain_codeset** function.

The message catalog used by the functions is at the pathname *dirname*/*locale*/*category*/*domainname*.mo. Here *dirname* is the directory specified through **bindtextdomain**. Its default is system and configuration dependent; typically it is *prefix*/share/locale, where *prefix* is the installation prefix of the package. *locale* is the name of the current locale facet; the GNU implementation also tries generalizations, such as the language name without the territory name. *category* is **LC_MESSAGES** for the **gettext** and **dgettext** functions, or the argument passed to the **dcgettext** function.

If the **LANGUAGE** environment variable is set to a nonempty value, and the locale is not the "C" locale, the value of **LANGUAGE** is assumed to contain a colon separated list of locale names. The functions will attempt to look up a translation of *msgid* in each of the locales in turn. This is a GNU extension.

In the "C" locale, or if none of the used catalogs contain a translation for *msgid*, the **gettext**, **dgettext** and **dcgettext** functions return *msgid*.

RETURN VALUE

If a translation was found in one of the specified catalogs, it is converted to the locale's codeset and returned. The resulting string is statically allocated and must not be modified or freed. Otherwise *msgid* is returned.

ERRORS

errno is not modified.

BUGS

The return type ought to be **const char ***, but is **char *** to avoid warnings in C code predating ANSI C.

When an empty string is used for *msgid*, the functions may return a nonempty string.

SEE ALSO

ngettext(3), **dngettext**(3), **dcngettext**(3), **setlocale**(3), **textdomain**(3), **bindtextdomain**(3), **bind_textdomain_codeset**(3), **msgfmt**(1)

NAME

ngettext, dngettext, dcngettext − translate message and choose plural form

SYNOPSIS

#include <libintl.h>

char * ngettext (const char * *msgid***, const char *** *msgid_plural***,**
unsigned long int *n***);**
char * dngettext (const char * *domainname***,**
const char * *msgid***, const char *** *msgid_plural***,**
unsigned long int *n***);**
char * dcngettext (const char * *domainname***,**
const char * *msgid***, const char *** *msgid_plural***,**
unsigned long int *n***, int** *category***);**

DESCRIPTION

The **ngettext**, **dngettext** and **dcngettext** functions attempt to translate a text string into the user's native language, by looking up the appropriate plural form of the translation in a message catalog.

Plural forms are grammatical variants depending on the a number. Some languages have two forms, called singular and plural. Other languages have three forms, called singular, dual and plural. There are also languages with four forms.

The **ngettext**, **dngettext** and **dcngettext** functions work like the **gettext**, **dgettext** and **dcgettext** functions, respectively. Additionally, they choose the appropriate plural form, which depends on the number *n* and the language of the message catalog where the translation was found.

In the "C" locale, or if none of the used catalogs contain a translation for *msgid*, the **ngettext**, **dngettext** and **dcngettext** functions return *msgid* if *n* == 1, or *msgid_plural* if *n* != 1.

RETURN VALUE

If a translation was found in one of the specified catalogs, the appropriate plural form is converted to the locale's codeset and returned. The resulting string is statically allocated and must not be modified or freed. Otherwise *msgid* or *msgid_plural* is returned, as described above.

ERRORS

errno is not modified.

BUGS

The return type ought to be **const char ***, but is **char *** to avoid warnings in C code predating ANSI C.

SEE ALSO

gettext(3), **dgettext**(3), **dcgettext**(3)

NAME

textdomain − set domain for future gettext() calls

SYNOPSIS

#include <libintl.h>

char * textdomain (const char * *domainname***);**

DESCRIPTION

The **textdomain** function sets or retrieves the current message domain.

A message domain is a set of translatable *msgid* messages. Usually, every software package has its own message domain. The domain name is used to determine the message catalog where a translation is looked up; it must be a non-empty string.

The current message domain is used by the **gettext**, **ngettext** functions, and by the **dgettext**, **dcgettext**, **dngettext** and **dcngettext** functions when called with a NULL domainname argument.

If *domainname* is not NULL, the current message domain is set to *domainname*. The string the function stores internally is a copy of the *domainname* argument.

If *domainname* is NULL, the function returns the current message domain.

RETURN VALUE

If successful, the **textdomain** function returns the current message domain, after possibly changing it. The resulting string is valid until the next **textdomain** call and must not be modified or freed. If a memory allocation failure occurs, it sets **errno** to **ENOMEM** and returns NULL.

ERRORS

The following error can occur, among others:

ENOMEM

Not enough memory available.

BUGS

The return type ought to be **const char ***, but is **char *** to avoid warnings in C code predating ANSI C.

SEE ALSO

gettext(3), **ngettext**(3), **bindtextdomain**(3), **bind_textdomain_codeset**(3)