

NAME

I18N::Langinfo - query locale information

SYNOPSIS

```
use I18N::Langinfo;
```

DESCRIPTION

The `langinfo()` function queries various locale information that can be used to localize output and user interfaces. The `langinfo()` requires one numeric argument that identifies the locale constant to query: if no argument is supplied, `$_` is used. The numeric constants appropriate to be used as arguments are exportable from `I18N::Langinfo`.

The following example will import the `langinfo()` function itself and three constants to be used as arguments to `langinfo()`: a constant for the abbreviated first day of the week (the numbering starts from Sunday = 1) and two more constants for the affirmative and negative answers for a yes/no question in the current locale.

```
use I18N::Langinfo qw(langinfo ABDAY_1 YESSTR NOSTR);

my ($abday_1, $yesstr, $nostr) = map { langinfo } qw(ABDAY_1 YESSTR
NOSTR);

print "$abday_1? [$yesstr/$nostr] ";
```

In other words, in the "C" (or English) locale the above will probably print something like:

```
Sun? [yes/no]
```

but under a French locale

```
dim? [oui/non]
```

The usually available constants are

```
ABDAY_1 ABDAY_2 ABDAY_3 ABDAY_4 ABDAY_5 ABDAY_6 ABDAY_7
ABMON_1 ABMON_2 ABMON_3 ABMON_4 ABMON_5 ABMON_6
ABMON_7 ABMON_8 ABMON_9 ABMON_10 ABMON_11 ABMON_12
DAY_1 DAY_2 DAY_3 DAY_4 DAY_5 DAY_6 DAY_7
MON_1 MON_2 MON_3 MON_4 MON_5 MON_6
MON_7 MON_8 MON_9 MON_10 MON_11 MON_12
```

for abbreviated and full length days of the week and months of the year,

```
D_T_FMT D_FMT T_FMT
```

for the date-time, date, and time formats used by the `strftime()` function (see *POSIX*)

```
AM_STR PM_STR T_FMT_AMPM
```

for the locales for which it makes sense to have ante meridiem and post meridiem time formats,

```
CODESET CRNCYSTR RADIXCHAR
```

for the character code set being used (such as "ISO8859-1", "cp850", "koi8-r", "sjis", "utf8", etc.), for the currency string, for the radix character used between the integer and the fractional part of decimal

numbers (yes, this is redundant with `POSIX::localeconv()`)

```
YESSTR YESEXPR NOSTR NOEXPR
```

for the affirmative and negative responses and expressions, and

```
ERA ERA_D_FMT ERA_D_T_FMT ERA_T_FMT
```

for the Japanese Emperor eras (naturally only defined under Japanese locales).

See your *langinfo(3)* for more information about the available constants. (Often this means having to look directly at the *langinfo.h* C header file.)

Note that unfortunately none of the above constants are guaranteed to be available on a particular platform. To be on the safe side you can wrap the import in an eval like this:

```
eval {
    require I18N::Langinfo;
    I18N::Langinfo->import(qw(langinfo CODESET));
    $codeset = langinfo(CODESET()); # note the ()
};
if (!$?) { ... failed ... }
```

EXPORT

By default only the `langinfo()` function is exported.

SEE ALSO

perllocale, *"localeconv" in POSIX*, *"setlocale" in POSIX*, *nl_langinfo(3)*.

The `langinfo()` is just a wrapper for the C `nl_langinfo()` interface.

AUTHOR

Jarkko Hietaniemi, <jhi@hut.fi>

COPYRIGHT AND LICENSE

Copyright 2001 by Jarkko Hietaniemi

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.