
Subject: Re: LOCALE_GET()

Posted by [James Kuyper](#) on Thu, 10 Apr 2003 19:56:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

Wayne Landsman wrote:

>

> Anyone know anything about the documented (since V5.3) LOCALE_GET()
> function? It has no parameters or keywords, and the documentation just
> says that it "returns the current locale (string) of the operating
> platform". Well, on my Solaris and Linux machines, it just returns
> the single letter "C" -- not very useful, I should think. On Windows

"C" is the standard initial locale. In our copy of IDL 5.4, there's no hint in the online help that there's any IDL equivalent of the setlocale() function in C. Therefore, I imagine that LOCALE_GET() is going to continue giving you "C" unless you link to some C code that calls setlocale(). I'm going to talk about locales from a C perspective, because that's the one I know best.

There's a separate locale for each of the following categories:

LC_COLLATE
LC_CTYPE
LC_MONETARY
LC_NUMERIC
LC_TIME
LC_MESSAGES
LC_ALL

For instance, Americans will write floating point numbers as "7,396.45", while Europeans would write that same number as "7.396,45". That's controlled by the LC_NUMERIC category. I'd assume that since LOCALE_GET() doesn't seem to take a category argument, it does the equivalent of setlocale(LC_ALL, NULL) (a null pointer returns a pointer to the current value of the category, without actually changing it).

If you execute setlocale(category, "") from within a C program, it selects the default locale for that category. How the default locale is determined varies from one type of system to another. On the unix-like systems that I use most, the default locale is controlled by environment variables with names corresponding to the category names. The environment variable LANG is used if the category-specific environment variable is not defined.

The list of valid locales is determined in a system specific way. On my IRIX system, the set of supported locales is stored in /usr/lib/locale:

.C/ en_AU.ISO8859-15/ fr_BE.ISO8859-15/

```

nl_NL.ISO8859-15/
POSIX/      en_CA/      fr_CA/      no/
TZ/         en_CA.ISO8859-15/ fr_CA.ISO8859-15/
no_NO.ISO8859-15/
charmap/     en_GB.ISO8859-15/ fr_CH/      pl/
cs/          en_US/      fr_CH.ISO8859-15/ pt/
da/          en_US.ISO8859-15/ fr_FR.ISO8859-15/ pt_BR/
da_DK.ISO8859-15/ es/      hu/
pt_BR.ISO8859-15/
de/          es_AR/      is/
pt_PT.ISO8859-15/
de_AT/       es_AR.ISO8859-15/ is_IS.ISO8859-15/ ru/
de_AT.ISO8859-15/ es_ES.ISO8859-15/ it/      sk/
de_CH/       es_MX/      it_CH/      sv/
de_CH.ISO8859-15/ es_MX.ISO8859-15/ it_CH.ISO8859-15/
sv_SE.ISO8859-15/
de_DE.ISO8859-15/ fi/      it_IT.ISO8859-15/ tr/
el/          fi_FI.ISO8859-15/ nl/
en/          fr/          nl_BE/
en_AU/       fr_BE/      nl_BE.ISO8859-15/

```

> it returns "English_United States:442" and so I suppose it could be used
> to determine the language and nationality of the current user.

It's not the current user, but the current operating environment. The computer has no way to know whether or not the current user is a frenchman, but it does know whether it's currently set up to print things in french-style formats.

> ... Has
> anyone actually used `LOCALE_GET()` for anything useful?

Without having the equivalent of C's `setlocale()`, I can't see much point to it. In C code, `setlocale()` incidentally returns the current locale. The typical use is to save the current locale at the same time you change it:

```
old_locale = setlocale(category, "fr");
```

Then you do a bunch of output which requires that you print things out in a French style. When you're done, restore the old locale:

```
setlocale(category, old_locale);
```

It's generally not very useful to actually examine `old_locale`.