NAME

charnames - define character names for \N{named} string literal escapes

SYNOPSIS

use charnames ':full';
print "\N{GREEK SMALL LETTER SIGMA} is called sigma.\n";

use charnames ':short';
print "\N{greek:Sigma} is an upper-case sigma.\n";

use charnames qw(cyrillic greek);
print "\N{sigma} is Greek sigma, and \N{be} is Cyrillic b.\n";

use charnames ':full', ':alias' => { 
  e_ACUTE => "LATIN SMALL LETTER E WITH ACUTE",
};
print "\N{e_ACUTE} is a small letter e with an acute.\n";

use charnames ();
print charnames::viacode(0x1234); # prints "ETHIOPIC SYLLABLE SEE"
printf "%04X", charnames::vianame("GOTHIC LETTER AHSA"); # prints "10330"

DESCRIPTION

Pragma use charnames supports arguments :full, :short, script names and customized aliases. If :full is present, for expansion of \N(CHARNAME), the string CHARNAME is first looked up in the list of standard Unicode character names. If :short is present, and CHARNAME has the form SCRIPT:CNAME, then CNAME is looked up as a letter in script SCRIPT. If pragma use charnames is used with script name arguments, then for \N(CHARNAME) the name CHARNAME is looked up as a letter in the given scripts (in the specified order). Customized aliases are explained in CUSTOM ALIASES.

For lookup of CHARNAME inside a given script SCRIPTNAME this pragma looks for the names

     SCRIPTNAME CAPITAL LETTER CHARNAME
     SCRIPTNAME SMALL LETTER CHARNAME
     SCRIPTNAME LETTER CHARNAME

in the table of standard Unicode names. If CHARNAME is lowercase, then the CAPITAL variant is ignored, otherwise the SMALL variant is ignored.

Note that \N(...) is compile-time, it's a special form of string constant used inside double-quoted strings: in other words, you cannot use variables inside the \N(...). If you want similar run-time functionality, use charnames::vianame().

For the C0 and C1 control characters (U+0000..U+001F, U+0080..U+009F) as of Unicode 3.1, there are no official Unicode names but you can use instead the ISO 6429 names (LINE FEED, ESCAPE, and so forth). In Unicode 3.2 (as of Perl 5.8) some naming changes take place ISO 6429 has been updated, see ALIASES. Also note that the U+UU80, U+0081, U+0084, and U+0099 do not have names even in ISO 6429.

Since the Unicode standard uses "U+HHHH", so can you: "\N{U+263a}" is the Unicode smiley face, or "\N{WHITE SMILING FACE}".
A few aliases have been defined for convenience: instead of having to use the official names

LINE FEED (LF)
FORM FEED (FF)
CARRIAGE RETURN (CR)
NEXT LINE (NEL)

(yes, with parentheses) one can use

LINE FEED
FORM FEED
CARRIAGE RETURN
NEXT LINE
LF
FF
CR
NEL

One can also use

BYTE ORDER MARK
BOM

and

ZWNJ
ZWJ

for ZERO WIDTH NON-JOINER and ZERO WIDTH JOINER.

For backward compatibility one can use the old names for certain C0 and C1 controls

<table>
<thead>
<tr>
<th>old</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>HORIZONTAL TABULATION</td>
<td>CHARACTER TABULATION</td>
</tr>
<tr>
<td>VERTICAL TABULATION</td>
<td>LINE TABULATION</td>
</tr>
<tr>
<td>FILE SEPARATOR</td>
<td>INFORMATION SEPARATOR FOUR</td>
</tr>
<tr>
<td>GROUP SEPARATOR</td>
<td>INFORMATION SEPARATOR THREE</td>
</tr>
<tr>
<td>RECORD SEPARATOR</td>
<td>INFORMATION SEPARATOR TWO</td>
</tr>
<tr>
<td>UNIT SEPARATOR</td>
<td>INFORMATION SEPARATOR ONE</td>
</tr>
<tr>
<td>PARTIAL LINE DOWN</td>
<td>PARTIAL LINE FORWARD</td>
</tr>
<tr>
<td>PARTIAL LINE UP</td>
<td>PARTIAL LINE BACKWARD</td>
</tr>
</tbody>
</table>

but the old names in addition to giving the character will also give a warning about being deprecated.

CUSTOM ALIASES

This version of charnames supports three mechanisms of adding local or customized aliases to standard Unicode naming conventions (:full)

Anonymous hashes

use charnames ":full", ":alias" => { e_ACUTE => "LATIN SMALL LETTER E WITH ACUTE",
                                   };
my $str = "\N{e_ACUTE}";
Perl version 5.10.0 documentation - charnames

Alias file

use charnames ":full", ":alias" => "pro";

will try to read "unicore/pro_alias.pl" from the @INC path. This file should return a list in plain perl:

{
    A_GRAVE       => "LATIN CAPITAL LETTER A WITH GRAVE",
    A_CIRCUM      => "LATIN CAPITAL LETTER A WITH CIRCUMFLEX",
    A_DIAERES     => "LATIN CAPITAL LETTER A WITH DIAERESIS",
    A_TILDE       => "LATIN CAPITAL LETTER A WITH TILDE",
    A_BREVE       => "LATIN CAPITAL LETTER A WITH BREVE",
    A_RING         => "LATIN CAPITAL LETTER A WITH RING ABOVE",
    A_MACRON      => "LATIN CAPITAL LETTER A WITH MACRON",
};

Alias shortcut

use charnames ":alias" => ":pro";

works exactly the same as the alias pairs, only this time, ":full" is inserted automatically as first argument (if no other argument is given).

cahrnames::viacode(code)

Returns the full name of the character indicated by the numeric code. The example

    print charnames::viacode(0x2722);

prints "FOUR TEARDROP-SPOKED ASTERISK".

Returns undef if no name is known for the code.

This works only for the standard names, and does not yet apply to custom translators.

Notice that the name returned for of U+FEFF is "ZERO WIDTH NO-BREAK SPACE", not "BYTE ORDER MARK".

cahrnames::vianame(name)

Returns the code point indicated by the name. The example

    printf "%04X", charnames::vianame("FOUR TEARDROP-SPOKED ASTERISK");

prints "2722".

Returns undef if the name is unknown.

This works only for the standard names, and does not yet apply to custom translators.

CUSTOM TRANSLATORS

The mechanism of translation of \N{...} escapes is general and not hardwired into charnames.pm. A module can install custom translations (inside the scope which uses the module) with the following magic incantation:

    sub import {
        shift;
    }
Here translator() is a subroutine which takes CHARNAME as an argument, and returns text to insert into the string instead of the \N{CHARNAME} escape. Since the text to insert should be different in bytes mode and out of it, the function should check the current state of bytes-flag as in:

```perl
use bytes (); # for $bytes::hint_bits
sub translator {
  if ($^H & $bytes::hint_bits) {
    return bytes_translator(@_);
  } else {
    return utf8_translator(@_);
  }
}
```

ILLEGAL CHARACTERS

If you ask by name for a character that does not exist, a warning is given and the Unicode replacement character "\x{FFFD}" is returned.

If you ask by code for a character that does not exist, no warning is given and undef is returned. (Though if you ask for a code point past U+10FFFF you do get a warning.)

BUGS

Since evaluation of the translation function happens in a middle of compilation (of a string literal), the translation function should not do any evals or requires. This restriction should be lifted in a future version of Perl.