NAME

encoding::warnings - Warn on implicit encoding conversions

VERSION

This document describes version 0.11 of encoding::warnings, released June 5, 2007.

SYNOPSIS

```
use encoding::warnings; # or 'FATAL' to raise fatal exceptions

utf8::encode($a = chr(20000)); # a byte-string (raw bytes)
$b = chr(20000); # a unicode-string (wide characters)

# "Bytes implicitly upgraded into wide characters as iso-8859-1"
$c = $a . $b;
```

DESCRIPTION

Overview of the problem

By default, there is a fundamental asymmetry in Perl's unicode model: implicit upgrading from byte-strings to unicode-strings assumes that they were encoded in *ISO 8859-1 (Latin-1)*, but unicode-strings are downgraded with UTF-8 encoding. This happens because the first 256 codepoints in Unicode happens to agree with Latin-1.

However, this silent upgrading can easily cause problems, if you happen to mix unicode strings with non-Latin1 data — i.e. byte-strings encoded in UTF-8 or other encodings. The error will not manifest until the combined string is written to output, at which time it would be impossible to see where did the silent upgrading occur.

Detecting the problem

This module simplifies the process of diagnosing such problems. Just put this line on top of your main program:

```
use encoding::warnings;
```

Afterwards, implicit upgrading of high-bit bytes will raise a warning. Ex.:`Bytes implicitly upgraded into wide characters as iso-8859-1 at - line 7`.

However, strings composed purely of ASCII code points (`0x00..0x7F`) will *not* trigger this warning.

You can also make the warnings fatal by importing this module as:

```
use encoding::warnings 'FATAL';
```

Solving the problem

Most of the time, this warning occurs when a byte-string is concatenated with a unicode-string. There are a number of ways to solve it:

- Upgrade both sides to unicode-strings

  If your program does not need compatibility for Perl 5.6 and earlier, the recommended approach is to apply appropriate IO disciplines, so all data in your program become unicode-strings. See encoding, open and "binmode" in perlfunc(1) for how.

- Downgrade both sides to byte-strings

  The other way works too, especially if you are sure that all your data are under the same encoding, or if compatibility with older versions of Perl is desired.

  You may downgrade strings with `Encode::encode` and `utf8::encode` See Encode and utf8 for details.

- Specify the encoding for implicit byte-string upgrading

  If you are confident that all byte-strings will be in a specific encoding like UTF-8, *and* need not support older versions of Perl, use the `encoding` pragma:

```
use encoding 'utf8';
```

Similarly, this will silence warnings from this module, and preserve the default behaviour:

```
use encoding 'iso-8859-1';
```

However, note that `use encoding` actually had three distinct effects:

- PerlIO layers for **STDIN** and **STDOUT**

  This is similar to what open pragma does.

- Literal conversions

  This turns *all* literal string in your program into unicode-strings (equivalent to a `use utf8`), by decoding them using the specified encoding.

- Implicit upgrading for byte-strings

  This will silence warnings from this module, as shown above.

Because literal conversions also work on empty strings, it may surprise some people:

```
use encoding 'big5';

my $byte_string = pack("C*", 0xA4, 0x40);
print length $a; # 2 here.
$a .= ""; # concatenating with a unicode string...
print length $a; # 1 here!
```

In other words, do not `use encoding` unless you are certain that the program will not deal with any raw, 8-bit binary data at all.

However, the `Filter => 1` flavor of `use encoding` will *not* affect implicit upgrading for byte-strings, and is thus incapable of silencing warnings from this module. See encoding for more details.

CAVEATS

For Perl 5.9.4 or later, this module's effect is lexical.

For Perl versions prior to 5.9.4, this module affects the whole script, instead of inside its lexical block.

SEE ALSO

perlunicode(1), perluniintro(1)

open, utf8, encoding, Encode

AUTHORS

Audrey Tang

COPYRIGHT