NAME

bytes - Perl pragma to force byte semantics rather than character semantics

NOTICE

This pragma reflects early attempts to incorporate Unicode into perl and has since been superseded. It breaks encapsulation (i.e. it exposes the innards of how the perl executable currently happens to store a string), and use of this module for anything other than debugging purposes is strongly discouraged. If you feel that the functions here within might be useful for your application, this possibly indicates a mismatch between your mental model of Perl Unicode and the current reality. In that case, you may wish to read some of the perl Unicode documentation: perluniintro, perlunitut, perlunifaq and perlunicode.

SYNOPSIS

```
use bytes;
... chr(...); # or bytes::chr
... index(...); # or bytes::index
... length(...); # or bytes::length
... ord(...); # or bytes::ord
... rindex(...); # or bytes::rindex
... substr(...); # or bytes::substr
no bytes;
```

DESCRIPTION

The `use bytes` pragma disables character semantics for the rest of the lexical scope in which it appears. `no bytes`can be used to re verse the effect of `use bytes` within the current lexical scope.

Perl normally assumes character semantics in the presence of character data (i.e. data that has come from a source that has been marked as being of a particular character encoding). When `use bytes` is in effect, the encoding is temporarily ignored, and each string is treated as a series of bytes.

As an example, when Perl sees `$x = chr(400)`, it encodes the character in UTF-8 and stores it in `$x`. Then it is marked as character data, so, for instance, `length $x` returns `1`. However, in the scope of the `bytes` pragma, `$x` is treated as a series of bytes - the bytes that make up the UTF8 encoding - and `length $x`returns `2`:

```
$x = chr(400);
print "Length is ", length $x, "\n"; # "Length is 1"
printf "Contents are %vd\n", $x; # "Contents are 400"
{
use bytes; # or "require bytes; bytes::length()"
print "Length is ", length $x, "\n"; # "Length is 2"
printf "Contents are %vd\n", $x; # "Contents are 198.144"
}
```

chr(), *ord()*, *substr()*, *index()* and *rindex()* behave similarly.

For more on the implications and differences between character semantics and byte semantics, see perluniintro(1) and perlunicode.

LIMITATIONS

bytes::substr() does not work as an *lvalue()*.

SEE ALSO

perluniintro(1), perlunicode(1), utf8