

**NAME**

open - perl pragma to set default PerlIO layers for input and output

**SYNOPSIS**

```
use open IN => ":crlf", OUT => ":bytes";
use open OUT => ':utf8';
use open IO => ":encoding(iso-8859-7)";

use open IO => ':locale';

use open ':encoding(utf8)';
use open ':locale';
use open ':encoding(iso-8859-7)';

use open ':std';
```

**DESCRIPTION**

Full-fledged support for I/O layers is now implemented provided Perl is configured to use PerlIO as its IO system (which is now the default).

The `open` pragma serves as one of the interfaces to declare default “layers” (also known as “disciplines”) for all I/O. Any two-argument `open()`, `readpipe()` (aka `qx//`) and similar operators found within the lexical scope of this pragma will use the declared defaults. Even three-argument opens may be affected by this pragma when they don’t specify IO layers in `MODE`.

With the `IN` subpragma you can declare the default layers of input streams, and with the `OUT` subpragma you can declare the default layers of output streams. With the `IO` subpragma you can control both input and output streams simultaneously.

If you have a legacy encoding, you can use the `:encoding(...)` tag.

If you want to set your encoding layers based on your locale environment variables, you can use the `:locale` tag. For example:

```
$ENV{LANG} = 'ru_RU.KOI8-R';
# the :locale will probe the locale environment variables like LANG
use open OUT => ':locale';
open(O, ">koi8");
print O chr(0x430); # Unicode CYRILLIC SMALL LETTER A = KOI8-R 0xc1
close O;
open(I, "<koi8");
printf "%#x\n", ord(<I>), "\n"; # this should print 0xc1
close I;
```

These are equivalent

```
use open ':encoding(utf8)';
use open IO => ':encoding(utf8)';
```

as are these

```
use open ':locale';
use open IO => ':locale';
```

and these

```
use open ':encoding(iso-8859-7)';
use open IO => ':encoding(iso-8859-7)';
```

The matching of encoding names is loose: case does not matter, and many encodings have several aliases. See [Encode::Supported](#) for details and the list of supported locales.

When `open()` is given an explicit list of layers (with the three-arg syntax), they override the list declared

using this pragma. *open()* can also be given a single colon (:) for a layer name, to override this pragma and use the default (:raw on Unix, :crlf on Windows).

The `:std` subpragma on its own has no effect, but if combined with the `:utf8` or `:encoding` subpragmas, it converts the standard filehandles (STDIN, STDOUT, STDERR) to comply with encoding selected for input/output handles. For example, if both input and out are chosen to be `:encoding(utf8)`, a `:std` will mean that STDIN, STDOUT, and STDERR are also in `:encoding(utf8)`. On the other hand, if only output is chosen to be in `:encoding(koi8r)`, a `:std` will cause only the STDOUT and STDERR to be in `koi8r`. The `:locale` subpragma implicitly turns on `:std`.

The logic of `:locale` is described in full in `encoding`, but in short it is first trying `nl_langinfo(CODESET)` and then guessing from the `LC_ALL` and `LANG` locale environment variables.

Directory handles may also support PerlIO layers in the future.

## NONPERLIO FUNCTIONALITY

If Perl is not built to use PerlIO as its IO system then only the two pseudo-layers `:bytes` and `:crlf` are available.

The `:bytes` layer corresponds to “binary mode” and the `:crlf` layer corresponds to “text mode” on platforms that distinguish between the two modes when opening files (which is many DOS-like platforms, including Windows). These two layers are no-ops on platforms where *binmode()* is a no-op, but perform their functions everywhere if PerlIO is enabled.

## IMPLEMENTATION DETAILS

There is a class method in `PerlIO::Layer` `find` which is implemented as XS code. It is called by `import` to validate the layers:

```
PerlIO::Layer::->find("perlio")
```

The return value (if defined) is a Perl object, of class `PerlIO::Layer` which is created by the C code in *perlio.c*. As yet there is nothing useful you can do with the object at the perl level.

## SEE ALSO

“binmode” in [perlfunc\(1\)](#), “open” in [perlfunc\(1\)](#), [perlunicode\(1\)](#), PerlIO, `encoding`