

NAME

B::Lint - Perl lint

SYNOPSIS

```
perl -MO=Lint[,OPTIONS] foo.pl
```

DESCRIPTION

The B::Lint module is equivalent to an extended version of the **-w** option of **perl**. It is named after the program *lint* which carries out a similar process for C programs.

OPTIONS AND LINT CHECKS

Option words are separated by commas (not whitespace) and follow the usual conventions of compiler backend options. Following any options (indicated by a leading -) come lint check arguments. Each such argument (apart from the special **all** and **none** options) is a word representing one possible lint check (turning on that check) or is **no-foo** (turning off that check). Before processing the check arguments, a standard list of checks is turned on. Later options override earlier ones. Available options are:

context

Produces a warning whenever an array is used in an implicit scalar context. For example, both of the lines

```
$foo = length(@bar);
$foo = @bar;
```

will elicit a warning. Using an explicit **scalar()** silences the warning. For example,

```
$foo = scalar(@bar);
```

implicit-read and implicit-write

These options produce a warning whenever an operation implicitly reads or (respectively) writes to one of Perl's special variables. For example, **implicit-read** will warn about these:

```
/foo/;
```

and **implicit-write** will warn about these:

```
s/foo/bar/;
```

Both **implicit-read** and **implicit-write** warn about this:

```
for (@a) { ... }
```

bare subs

This option warns whenever a bareword is implicitly quoted, but is also the name of a subroutine in the current package. Typical mistakes that it will trap are:

```
use constant foo => 'bar';
@a = ( foo => 1 );
$b{foo} = 2;
```

Neither of these will do what a naive user would expect.

dollar-underscore

This option warns whenever `$_` is used either explicitly anywhere or as the implicit argument of a **print** statement.

private-names

This option warns on each use of any variable, subroutine or method name that lives in a non-current package but begins with an underscore ("_"). Warnings aren't issued for the special case of the single character name "_" by itself (e.g. \$_ and @_).

undefined subs

This option warns whenever an undefined subroutine is invoked. This option will only catch explicitly invoked subroutines such as `foo()` and not indirect invocations such as `&$subref()` or `$obj->meth()`. Note that some programs or modules delay definition of subs until runtime by means of the AUTOLOAD mechanism.

regex variables

This option warns whenever one of the regexp variables `$'`, `$&` or `$'` is used. Any occurrence of any of these variables in your program can slow your whole program down. See *perlre* for details.

all

Turn all warnings on.

none

Turn all warnings off.

NON LINT-CHECK OPTIONS**-u Package**

Normally, Lint only checks the main code of the program together with all subs defined in package main. The `-u` option lets you include other package names whose subs are then checked by Lint.

BUGS

This is only a very preliminary version.

This module doesn't work correctly on thread-enabled perls.

AUTHOR

Malcolm Beattie, mbeattie@sable.ox.ac.uk.