

NAME

Dpkg::Version – handling and comparing dpkg-style version numbers

DESCRIPTION

The Dpkg::Version module provides pure-Perl routines to compare dpkg-style version numbers (as used in Debian packages) and also an object oriented interface overriding perl operators to do the right thing when you compare Dpkg::Version object between them.

METHODS

`$v = Dpkg::Version->new($version, %opts)`

Create a new Dpkg::Version object corresponding to the version indicated in the string (scalar) `$version`. By default it will accept any string and consider it as a valid version. If you pass the option `“check => 1”`, it will return undef if the version is invalid (see `version_check` for details).

You can always call `$v->is_valid()` later on to verify that the version is valid.

boolean evaluation

When the Dpkg::Version object is used in a boolean evaluation (for example in `“if ($v)”` or `“$v ? \”$v\“ : ‘default’”`) it returns true if the version stored is valid (`$v->is_valid()`) and false otherwise.

Notice: Between dpkg 1.15.7.2 and 1.19.1 this overload used to return `$v->as_string()` if `$v->is_valid()`, a breaking change in behavior that caused “0” versions to be evaluated as false. To catch any possibly intended code that relied on those semantics, this overload will emit a warning with category `“Dpkg::Version::semantic_change::overload::bool”` until dpkg 1.20.x. Once fixed, or for already valid code the warning can be quiesced with

```
no if $Dpkg::Version::VERSION ge '1.02',
    warnings => qw(Dpkg::Version::semantic_change::overload::bool);
```

added after the use `Dpkg::Version`.

`$v->is_valid()`

Returns true if the version is valid, false otherwise.

`$v->epoch()`, `$v->version()`, `$v->revision()`

Returns the corresponding part of the full version string.

`$v->is_native()`

Returns true if the version is native, false if it has a revision.

`$v1 <=> $v2`, `$v1 < $v2`, `$v1 <= $v2`, `$v1 > $v2`, `$v1 >= $v2`

Numerical comparison of various versions numbers. One of the two operands needs to be a Dpkg::Version, the other one can be anything provided that its string representation is a version number.

`“$v”`, `$v->as_string()`, `$v->as_string(%options)`

Accepts an optional option hash reference, affecting the string conversion.

Options:

`omit_epoch` (defaults to 0)

Omit the epoch, if present, in the output string.

`omit_revision` (defaults to 0)

Omit the revision, if present, in the output string.

Returns the string representation of the version number.

FUNCTIONS

All the functions are exported by default.

`version_compare($a, $b)`

Returns -1 if `$a` is earlier than `$b`, 0 if they are equal and 1 if `$a` is later than `$b`.

If `$a` or `$b` are not valid version numbers, it dies with an error.

`version_compare_relation($a, $rel, $b)`

Returns the result (0 or 1) of the given comparison operation. This function is implemented on top of **version_compare()**.

Allowed values for `$rel` are the exported constants `REL_GT`, `REL_GE`, `REL_EQ`, `REL_LE`, `REL_LT`. Use **version_normalize_relation()** if you have an input string containing the operator.

`$rel = version_normalize_relation($rel_string)`

Returns the normalized constant of the relation `$rel` (a value among `REL_GT`, `REL_GE`, `REL_EQ`, `REL_LE` and `REL_LT`). Supported relations names in input are: “gt”, “ge”, “eq”, “le”, “lt”, “>>”, “>=”, “=”, “<=”, “<<”. “>” and “<” are also supported but should not be used as they are obsolete aliases of “>=” and “<=”.

`version_compare_string($a, $b)`

String comparison function used for comparing non-numerical parts of version numbers. Returns `-1` if `$a` is earlier than `$b`, `0` if they are equal and `1` if `$a` is later than `$b`.

The “~” character always sort lower than anything else. Digits sort lower than non-digits. Among remaining characters alphabetic characters (`A-Z`, `a-z`) sort lower than the other ones. Within each range, the ASCII decimal value of the character is used to sort between characters.

`version_compare_part($a, $b)`

Compare two corresponding sub-parts of a version number (either upstream version or debian revision).

Each parameter is split by **version_split_digits()** and resulting items are compared together. As soon as a difference happens, it returns `-1` if `$a` is earlier than `$b`, `0` if they are equal and `1` if `$a` is later than `$b`.

`@items = version_split_digits($version)`

Splits a string in items that are each entirely composed either of digits or of non-digits. For instance for “1.024~beta1+svn234” it would return (“1”, “.”, “024”, “~beta”, “1”, “+svn”, “234”).

`($ok, $msg) = version_check($version)`

`$ok = version_check($version)`

Checks the validity of `$version` as a version number. Returns `1` in `$ok` if the version is valid, `0` otherwise. In the latter case, `$msg` contains a description of the problem with the `$version` scalar.

CHANGES

Version 1.02 (dpkg 1.19.1)

Semantic change: bool evaluation semantics restored to their original behavior.

Version 1.01 (dpkg 1.17.0)

New argument: Accept an options argument in `$v->as_string()`.

New method: `$v->is_native()`.

Version 1.00 (dpkg 1.15.6)

Mark the module as public.