

## NAME

ExtUtils::Constant::Base - base class for ExtUtils::Constant objects

## SYNOPSIS

```
require ExtUtils::Constant::Base;
@ISA = 'ExtUtils::Constant::Base';
```

## DESCRIPTION

ExtUtils::Constant::Base provides a base implementation of methods to generate C code to give fast constant value lookup by named string. Currently it's mostly used ExtUtils::Constant::XS, which generates the lookup code for the constant() subroutine found in many XS modules.

## USAGE

ExtUtils::Constant::Base exports no subroutines. The following methods are available

### header

A method returning a scalar containing definitions needed, typically for a C header file.

### memEQ\_clause args\_hashref

A method to return a suitable C `if` statement to check whether *name* is equal to the C variable *name*. If *checked\_at* is defined, then it is used to avoid `memEQ` for short names, or to generate a comment to highlight the position of the character in the `switch` statement.

If `<checked_at>` is a reference to a scalar, then instead it gives the characters pre-checked at the beginning, (and the number of chars by which the C variable name has been advanced. These need to be chopped from the front of *name*).

### dump\_names arg\_hashref, ITEM...

An internal function to generate the embedded perl code that will regenerate the constant subroutines. *default\_type*, *types* and *ITEMs* are the same as for `C_constant`. *indent* is treated as number of spaces to indent by. If *declare\_types* is true a `$types` is always declared in the perl code generated, if defined and false never declared, and if undefined `$types` is only declared if the values in *types* as passed in cannot be inferred from *default\_types* and the *ITEMs*.

### assign arg\_hashref, VALUE...

A method to return a suitable assignment clause. If *type* is aggregate (eg *PVN* expects both pointer and length) then there should be multiple *VALUES* for the components. *pre* and *post* if defined give snippets of C code to proceed and follow the assignment. *pre* will be at the start of a block, so variables may be defined in it.

### return\_clause arg\_hashref, ITEM

A method to return a suitable `#ifdef` clause. *ITEM* is a hashref (as passed to `C_constant` and `match_clause`). *indent* is the number of spaces to indent, defaulting to 6.

### switch\_clause arg\_hashref, NAMELEN, ITEMHASH, ITEM...

An internal method to generate a suitable `switch` clause, called by `C_constant`. *ITEMs* are in the hash ref format as given in the description of `C_constant`, and must all have the names of the same length, given by *NAMELEN*. *ITEMHASH* is a reference to a hash, keyed by name, values being the hashrefs in the *ITEM* list. (No parameters are modified, and there can be keys in the *ITEMHASH* that are not in the list of *ITEMs* without causing problems - the hash is passed in to save generating it afresh for each call).

### params WHAT

An "internal" method, subject to change, currently called to allow an overriding class to cache information that will then be passed into all the `*param*` calls. (Yes, having to read the source

to make sense of this is considered a known bug). *WHAT* is be a hashref of types the constant function will return. In ExtUtils::Constant::XS this method is used to returns a hashref keyed IV NV PV SV to show which combination of pointers will be needed in the C argument list generated by C\_constant\_other\_params\_definition and C\_constant\_other\_params

dogfood arg\_hashref, ITEM...

An internal function to generate the embedded perl code that will regenerate the constant subroutines. Parameters are the same as for C\_constant.

Currently the base class does nothing and returns an empty string.

normalise\_items args, default\_type, seen\_types, seen\_items, ITEM...

Convert the items to a normalised form. For 8 bit and Unicode values converts the item to an array of 1 or 2 items, both 8 bit and UTF-8 encoded.

C\_constant arg\_hashref, ITEM...

A function that returns a **list** of C subroutine definitions that return the value and type of constants when passed the name by the XS wrapper. *ITEM...* gives a list of constant names. Each can either be a string, which is taken as a C macro name, or a reference to a hash with the following keys

name

The name of the constant, as seen by the perl code.

type

The type of the constant (*IV*, *NV* etc)

value

A C expression for the value of the constant, or a list of C expressions if the type is aggregate. This defaults to the *name* if not given.

macro

The C pre-processor macro to use in the *#ifdef*. This defaults to the *name*, and is mainly used if *value* is an *enum*. If a reference an array is passed then the first element is used in place of the *#ifdef* line, and the second element in place of the *#endif*. This allows pre-processor constructions such as

```
#if defined (foo)
#if !defined (bar)
...
#endif
#endif
```

to be used to determine if a constant is to be defined.

A "macro" 1 signals that the constant is always defined, so the *#if/#endif* test is omitted.

default

Default value to use (instead of *croaking* with "your vendor has not defined...") to return if the macro isn't defined. Specify a reference to an array with type followed by value(s).

pre

C code to use before the assignment of the value of the constant. This allows you to use temporary variables to extract a value from part of a *struct* and return this as *value*. This C code is places at the start of a block, so you can declare variables in it.

post

C code to place between the assignment of value (to a temporary) and the return from the function. This allows you to clear up anything in *pre*. Rarely needed.

def\_pre

def\_post

Equivalents of *pre* and *post* for the default value.

utf8

Generated internally. Is zero or undefined if name is 7 bit ASCII, "no" if the name is 8 bit (and so should only match if SvUTF8() is false), "yes" if the name is utf8 encoded.

The internals automatically clone any name with characters 128-255 but none 256+ (ie one that could be either in bytes or utf8) into a second entry which is utf8 encoded.

weight

Optional sorting weight for names, to determine the order of linear testing when multiple names fall in the same case of a switch clause. Higher comes earlier, undefined defaults to zero.

In the argument hashref, *package* is the name of the package, and is only used in comments inside the generated C code. *subname* defaults to `constant` if undefined.

*default\_type* is the type returned by `ITEMs` that don't specify their type. It defaults to the value of `default_type()`. *types* should be given either as a comma separated list of types that the C subroutine *subname* will generate or as a reference to a hash. *default\_type* will be added to the list if not present, as will any types given in the list of `ITEMs`. The resultant list should be the same list of types that `XS_constant` is given. [Otherwise `XS_constant` and `C_constant` may differ in the number of parameters to the constant function. *indent* is currently unused and ignored. In future it may be used to pass in information used to change the C indentation style used.] The best way to maintain consistency is to pass in a hash reference and let this function update it.

*breakout* governs when child functions of *subname* are generated. If there are *breakout* or more `ITEMs` with the same length of name, then the code to switch between them is placed into a function named *subname\_len*, for example `constant_5` for names 5 characters long. The default *breakout* is 3. A single `ITEM` is always inlined.

## BUGS

Not everything is documented yet.

Probably others.

## AUTHOR

Nicholas Clark <nick@ccl4.org> based on the code in `h2xs` by Larry Wall and others