

**NAME**

vmsish - Perl pragma to control VMS-specific language features

**SYNOPSIS**

```
use vmsish;

use vmsish 'status'; # or '$?'
use vmsish 'exit';
use vmsish 'time';

use vmsish 'hushed';
no vmsish 'hushed';
vmsish::hushed($hush);

use vmsish;
no vmsish 'time';
```

**DESCRIPTION**

If no import list is supplied, all possible VMS-specific features are assumed. Currently, there are four VMS-specific features available: 'status' (a.k.a '\$?'), 'exit', 'time' and 'hushed'.

If you're not running VMS, this module does nothing.

`vmsish status`

This makes  `$?`  and  `system`  return the native VMS exit status instead of emulating the POSIX exit status.

`vmsish exit`

This makes  `exit 1`  produce a successful exit (with status  `SS$_NORMAL` ), instead of emulating UNIX  `exit()`, which considers  `exit 1`  to indicate an error. As with the CRTL's  `exit()` function,  `exit 0`  is also mapped to an exit status of  `SS$_NORMAL` , and any other argument to  `exit()` is used directly as Perl's exit status.

`vmsish time`

This makes all times relative to the local time zone, instead of the default of Universal Time (a.k.a Greenwich Mean Time, or GMT).

`vmsish hushed`

This suppresses printing of VMS status messages to  `SYS$OUTPUT`  and  `SYS$error`  if Perl terminates with an error status. and allows programs that are expecting "unix-style" Perl to avoid having to parse VMS error messages. It does not suppress any messages from Perl itself, just the messages generated by DCL after Perl exits. The DCL symbol  `$STATUS`  will still have the termination status, but with a high-order bit set:

EXAMPLE:  `$ perl -e"exit 44;"`  Non-hushed error exit  `%SYSTEM-F-ABORT` , abort DCL message  `$ show sym $STATUS $STATUS == "%X0000002C"`

```
    $ perl -e"use vmsish qw(hushed); exit 44;"           Hushed error
exit
    $ show sym $STATUS
    $STATUS == "%X1000002C"
```

The 'hushed' flag has a global scope during compilation: the  `exit()` or  `die()` commands that are compiled after 'vmsish hushed' will be hushed when they are executed. Doing a "no vmsish 'hushed'" turns off the hushed flag.

The status of the `hushed` flag also affects output of VMS error messages from compilation errors. Again, you still get the Perl error message (and the code in `$STATUS`)

EXAMPLE: `use vmsish 'hushed'; # turn on hushed flag use Carp; # Carp compiled hushed exit 44; # will be hushed croak('I die'); # will be hushed no vmsish 'hushed'; # turn off hushed flag exit 44; # will not be hushed croak('I die2'); # WILL be hushed, croak was compiled hushed`

You can also control the `'hushed'` flag at run-time, using the built-in routine `vmsish::hushed()`. Without argument, it returns the hushed status. Since `vmsish::hushed` is built-in, you do not need to `"use vmsish"` to call it.

EXAMPLE: `if ($quiet_exit) { vmsish::hushed(1); } print "Sssshhh...I'm hushed...\n" if vmsish::hushed(); exit 44;`

Note that an `exit()` or `die()` that is compiled `'hushed'` because of `"use vmsish"` is not un-hushed by calling `vmsish::hushed(0)` at runtime.

The messages from error exits from inside the Perl core are generally more serious, and are not suppressed.

See *"Pragmatic Modules"* in *perlmod*.