NAME

XS::APItest - Test the perl C API

SYNOPSIS

```
use XS::APItest;
print_double(4);
```

ABSTRACT

This module tests the perl C API. Currently tests that `printf` works correctly.

DESCRIPTION

This module can be used to check that the perl C API is behaving correctly. This module provides test functions and an associated test script that verifies the output.

This module is not meant to be installed.

EXPORT

Exports all the test functions:

print_double

Test that a double-precision floating point number is formatted correctly by `printf`.

```
print_double( $val );
```

Output is sent to STDOUT.

print_long_double

Test that a `long double` is formatted correctly by `printf`. Takes no arguments - the test value is hard-wired into the function (as "7").

```
print_long_double();
```

Output is sent to STDOUT.

have_long_double

Determine whether a `long double` is supported by Perl. This should be used to determine whether to test `print_long_double`.

```
print_long_double() if have_long_double;
```

print_nv

Test that an `NV` is formatted correctly by `printf`.

```
print_nv( $val );
```

Output is sent to STDOUT.

print_iv

Test that an `IV` is formatted correctly by `printf`.

```
print_iv( $val );
```

Output is sent to STDOUT.

print_uv

Test that an `UV` is formatted correctly by `printf`.

```
print_uv( $val );
```

Output is sent to STDOUT.

print_int

Test that an `int` is formatted correctly by `printf`.

```
print_int( $val );
```

Output is sent to STDOUT.

print_long

Test that an `long` is formatted correctly by `printf`.

```
print_long( $val );
```

Output is sent to STDOUT.

print_float

Test that a single-precision floating point number is formatted correctly by `printf`.

```
print_float( $val );
```

Output is sent to STDOUT.

call_sv, **call_pv**, **call_method**

These exercise the C calls of the same names. Everything after the flags arg is passed as the the args to the called function. They return whatever the C function itself pushed onto the stack, plus the return value from the function; for example

```
    call_sv( sub { @_, 'c' }, G_ARRAY,  'a', 'b'); # returns 'a',
'b', 'c', 3
    call_sv( sub { @_ },      G_SCALAR, 'a', 'b'); # returns 'b', 1
```

eval_sv

Evaluates the passed SV. Result handling is done the same as for `call_sv()` etc.

eval_pv

Exercises the C function of the same name in scalar context. Returns the same SV that the C function returns.

require_pv

Exercises the C function of the same name. Returns nothing.

SEE ALSO

XS::Typemap, *perlapi*.

AUTHORS

Tim Jenness, <t.jenness@jach.hawaii.edu>, Christian Soeller, <csoelle@mph.auckland.ac.nz>, Hugo van der Sanden <hv@crypt.compulink.co.uk>

COPYRIGHT AND LICENSE