NAME

sigtrap - Perl pragma to enable simple signal handling

SYNOPSIS

```
use sigtrap;
use sigtrap qw(stack-trace old-interface-signals); # equivalent
use sigtrap qw(BUS SEGV PIPE ABRT);
use sigtrap qw(die INT QUIT);
use sigtrap qw(die normal-signals);
use sigtrap qw(die untrapped normal-signals);
use sigtrap qw(die untrapped normal-signals
  stack-trace any error-signals);
use sigtrap 'handler' => \&my_handler, 'normal-signals';
use sigtrap qw(handler my_handler normal-signals
          stack-trace error-signals);
```

DESCRIPTION

The **sigtrap** pragma is a simple interface to installing signal handlers. You can have it install one of two handlers supplied by **sigtrap** itself (one which provides a Perl stack trace and one which simply `die()`s), or alternately you can supply your own handler for it to install. It can be told only to install a handler for signals which are either untrapped or ignored. It has a couple of lists of signals to trap, plus you can supply your own list of signals.

The arguments passed to the `use` statement which invokes **sigtrap** are processed in order. When a signal name or the name of one of **sigtrap**'s signal lists is encountered a handler is immediately installed, when an option is encountered it affects subsequently installed handlers.

OPTIONS

SIGNAL HANDLERS

These options affect which handler will be used for subsequently installed signals.

stack-trace

The handler used for subsequently installed signals outputs a Perl stack trace to STDERR and then tries to dump core. This is the default signal handler.

die

The handler used for subsequently installed signals calls `die` (actually `croak`) with a message indicating which signal was caught.

handler *your-handler*

your-handler will be used as the handler for subsequently installed signals. *your-handler* can be any value which is valid as an assignment to an element of `%SIG`. See *perlvar* for examples of handler functions.

SIGNAL LISTS

sigtrap has a few built-in lists of signals to trap. They are:

normal-signals

These are the signals which a program might normally expect to encounter and which by default cause it to terminate. They are HUP, INT, PIPE and TERM.

error-signals

These signals usually indicate a serious problem with the Perl interpreter or with your script. They are ABRT, BUS, EMT, FPE, ILL, QUIT, SEGV, SYS and TRAP.

old-interface-signals

These are the signals which were trapped by default by the old **sigtrap** interface, they are ABRT, BUS, EMT, FPE, ILL, PIPE, QUIT, SEGV, SYS, TERM, and TRAP. If no signals or signals lists are passed to **sigtrap**, this list is used.

For each of these three lists, the collection of signals set to be trapped is checked before trapping; if your architecture does not implement a particular signal, it will not be trapped but rather silently ignored.

OTHER

untrapped

This token tells **sigtrap** to install handlers only for subsequently listed signals which aren't already trapped or ignored.

any

This token tells **sigtrap** to install handlers for all subsequently listed signals. This is the default behavior.

signal

Any argument which looks like a signal name (that is, `/^[A-Z][A-Z0-9]*$/`) indicates that **sigtrap** should install a handler for that name.

number

Require that at least version *number* of **sigtrap** is being used.

EXAMPLES

Provide a stack trace for the old-interface-signals:

```
use sigtrap;
```

Ditto:

```
use sigtrap qw(stack-trace old-interface-signals);
```

Provide a stack trace on the 4 listed signals only:

```
use sigtrap qw(BUS SEGV PIPE ABRT);
```

Die on INT or QUIT:

```
use sigtrap qw(die INT QUIT);
```

Die on HUP, INT, PIPE or TERM:

```
use sigtrap qw(die normal-signals);
```

Die on HUP, INT, PIPE or TERM, except don't change the behavior for signals which are already trapped or ignored:

```
use sigtrap qw(die untrapped normal-signals);
```

Die on receipt one of an of the **normal-signals** which is currently **untrapped**, provide a stack trace on receipt of **any** of the **error-signals**:

```
use sigtrap qw(die untrapped normal-signals
  stack-trace any error-signals);
```

Install my_handler() as the handler for the **normal-signals**:

```
use sigtrap 'handler', \&my_handler, 'normal-signals';
```

Install my_handler() as the handler for the normal-signals, provide a Perl stack trace on receipt of one of the error-signals:

```
use sigtrap qw(handler my_handler normal-signals
          stack-trace error-signals);
```