NAME

Thread::Signal - Start a thread which runs signal handlers reliably (for old code)

CAVEAT

For new code the use of the `Thread::Signal` module is discouraged and the direct use of the `threads` and associated modules is encouraged instead.

However, there is no direct equivalent of the Thread::Signal module in the new implementation of threads. On the bright side: signals are now delivered reliably to Perl programs that do not use threads. The handling of signals with the new threading features is up to the underlying thread implementation that is being used and may therefor be less reliable.

If you want to specify a thread-specific signal, you can alter the %SIG hash in the thread where you want to handle a signal differently from other threads. This at least seems to work under Linux. But there are no guarantees and your mileage may vary.

For the whole story about the development of threads in Perl, and why you should **not** be using this module unless you know what you're doing, see the CAVEAT of the `Thread` module.

SYNOPSIS

```
use Thread::Signal;

$SIG{HUP} = \&some_handler;
```

DESCRIPTION

The `Thread::Signal` module starts up a special signal handler thread. All signals to the process are delivered to it and it runs the associated `$SIG{FOO}` handlers for them. Without this module, signals arriving at inopportune moments (such as when perl's internals are in the middle of updating critical structures) cause the perl code of the handler to be run unsafely which can cause memory corruption or worse.

BUGS

This module changes the semantics of signal handling slightly in that the signal handler is run separately from the main thread (and in parallel with it). This means that tricks such as calling `die` from a signal handler behave differently (and, in particular, can't be used to exit directly from a system call).