Signal::StackTrace

When you're there and you know it.

Steven Lembark
<lembark@wrkhors.com>

Signals In General

- Signals are used on *NIX for asynchronous communication.
- You send them when you want to, the recipiant gets them [pretty much] right away.
- This gives a nice mechanism for notifying a process that it needs to do something – say print a stack trace.

Perly Signal Handling

- Pretty simple:

```
$SIG{ $signame } = sub { ... }
```

- You can turn them off just as easily:

```
$SIG{ $signame } = 'IGNORE'
or
delete $SIG{ $signame }
```

- You can localize them like any other value:

```
Local $SIG{ $signame } = sub { ... }
```

- In this case I used 'USR1' by default:

```
$SIG{ USR1 } = sub { ... }
```

Finding Where You Are From

- The 'caller' function tells where the currently executing sub was called from.
- In an array context this includes the subroutine and line number.
- Caller can look 'up' the stack by passing a value to caller.
- Incrementing the value until there are no more callers gives a stack trace.

Signal Handling With Caller

- Fortunately, signal handlers run in the context of the current call: caller reports the stack for the currently running subroutine.
- Stack tracing from a signal handler will tell where the code was running when the signal hit.

Stack Trace Code

```
my $stack_trace
= sub
{
    my %data = ();

    # walk up the stack until caller returns nada.

    for( my $i = 0 ; my @caller = caller $i ; ++$i )
    {
        # using a hash slice names the values.

        @data{ @headerz } = @caller;

        $print_list->( "Caller level $i:", \%data );
    }

    $print_list->( "End of trace" );

    return
};
```

Installing the Signal Handler

```
sub import
{
    shift;

    # remainder of the stack are signal names, default to SIGUSR1.
    # %SIG is global, no need to worry about the caller's package.

    if( @_ )
    {
        if( my @junk = grep { ! exists $known_sigz{ $_ } } @_ )
        {
            croak "Unknown signals: unknown signals @junk";
        }

        # all the signals are known, install them all
        # with the stack_trace handler.

        @SIG{ @_ } = ( $stack_trace ) x @_;
    }
    else
    {
        $SIG{ USR1 } = $stack_trace;
    }

    return
}
```

Oddz & Endz

- Legit signal names are installed with perl:

```
my %known_sigz  = ();

@known_sigz{ split ' ', $Config{ sig_name }  } = ();
```

- Pretty printing a list: Dumper refs.

```
my $print_list
= sub
{
    local $Data::Dumper::Purity        = 0;
    local $Data::Dumper::Terse         = 1;
    local $Data::Dumper::Indent        = 1;
    local $Data::Dumper::Deparse       = 1;
    local $Data::Dumper::Sortkeys      = 1;
    local $Data::Dumper::Deepcopy      = 0;
    local $Data::Dumper::Quotekeys     = 0;

    print STDERR join "\n", map { ref $_ ? Dumper $_ : $_ } @_
};
```