Debug
Debugger
while
Debugging

Speaker: Eugen Konkov
e-mail: eugen@konkov.top


4GIFs.com


4GIFs.com

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5
6      $x->{ a }++;
7  }
8
9
10
11
```

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5      print “$_\n”;
6      $x->{ a }++;
7  }
8
9  print Dump $x;
10
11
```

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5      # print “$_\n”;
6      $x->{ a }++;
7  }
8
9  # print Dump $x;
10
11
```

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5
6      $x->{ a }++;
7  }
8
9
10
11
```

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5      #DBG:      $_ #
6      $x->{ a }++;
7  }
8
9  #DBG: e $x #
10
11
```

Devel::DebugHooks::KillPrint

```
1
2   my $x =  { a =>  7 };
3
4   for( 1 .. 3 ) {
5       #DBG:iter $_ #
6       $x->{ a }++;
7   }
8
9   #DBG: e $x #
10
11
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint                    t.pl
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint                    t.pl

1
2
3
{ a => 10 }
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint                    t.pl

1
2
3
{ a => 10 }
```

Devel::DebugHooks::KillPrint

```
1
2   my $x =  { a =>  7 };
3
4   for( 1 .. 3 ) {
5       #DBG:iter $_ #
6       $x->{ a }++;
7   }
8
9   #DBG: e $x #
10
11
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint=iter        t.pl
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint=iter        t.pl

1
2
3
```

Devel::DebugHooks::KillPrint

```
$ perl -d:DebugHooks::KillPrint=default    t.pl

{ a => 10 }
```

Devel::DebugHooks::KillPrint

```
1
2  my $x =  { a =>  7 };
3
4  for( 1 .. 3 ) {
5      #DBG:iter $_ #
6      $x->{ a }++;
7  }
8
9  #DBG: e $x #
10
11
```

DEBUGGER'S INTERFACE

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
    0: use Devel::DebugHooks::Terminal;
    1: sub t2 {
   x2:     2;
    3: }
  >>4: 1;
   x5: t2();
   x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

Interface of Devel::DebugHooks

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

DEBUGGER'S COMMANDS

s — do step

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

s — do step

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
    x4: 1;
   >>5: t2();
    x6: 3
```

s — do step

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
   >>2:     2;
     3: }
    x4: 1;
   1>5: t2();
    x6: 3
```

s — do step

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
  >>2:     2;
     3: }
    x4: 1;
  1>5: t2();
    x6: 3
```

s — do step

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
    x4: 1;
    x5: t2();
   >>6: 3
```

n — do step over

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
   >>4: 1;
    x5: t2();
    x6: 3
```

n — do step over

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
    x4: 1;
   >>5: t2();
    x6: 3
```

n — do step over

```
$ perl -d:DebugHooks::Terminal t2.pl

/home/kes/tmp/t2.pl
     0: use Devel::DebugHooks::Terminal;
     1: sub t2 {
    x2:     2;
     3: }
    x4: 1;
    x5: t2();
   >>6: 3;
```

r — step out

```
  1: sub t2 {
>>2:     1;
 x3:     2;
  4: }
 x5: 1;
1>6: t2();
 x7: 3;
```

r — step out

```
r

   1: sub t2 {
  x2:     1;
  x3:     2;
   4: }
  x5: 1;
  x6: t2();
 >>7: 3;
```

r — step out

```
 x1: my $x =  bless \$x;
  2: sub t1 {
>>3:     return $x;
  4: }
  5: sub t2 {
 x6:     return $x;
  7: }
1>8: $x->t1()->t2();
 x9: 3;
```

r — step out

```
 x1: my $x =  bless \$x;
  2: sub t1 {
>>3:     return $x;
  4: }
  5: sub t2 {
 x6:     return $x;
  7: }
1>8: $x->t1()->t2();
 x9: 3;
```

r — step out

```
 x1: my $x =  bless \$x;
  2: sub t1 {
>>3:     return $x;
  4: }
  5: sub t2 {
 x6:     return $x;
  7: }
1>8: $x->t1()->t2();
 x9: 3;
```

r — step out

```
r

  x1: my $x =  bless \$x;
   2: sub t1 {
  x3:     return $x;
   4: }
   5: sub t2 {
>>6:     return $x;
   7: }
1>8: $x->t1()->t2();
  x9: 3;
```

r — step out

```
r

  x1: my $x =  bless \$x;
   2: sub t1 {
  x3:     return $x;
   4: }
   5: sub t2 {
  x6:     return $x;
   7: }
  x8: $x->t1()->t2();
 >>9: 3;
```

r N — step out few times

N – number of frames

```
  x1: my $x =  bless \$x;
   2: sub t1 {
>>3:     return $x;
   4: }
   5: sub t2 {
  x6:     return $x;
   7: }
1>8: $x->t1()->t2();
  x9: 3;
```

r N — step out few times

r 1

```
  x1: my $x =  bless \$x;
   2: sub t1 {
 >>3:     return $x;
   4: }
   5: sub t2 {
  x6:     return $x;
   7: }
 1>8: $x->t1()->t2();
  x9: 3;
```

r N — step out few times

```
 x1: my $x =  bless \$x;
  2: sub t1 {
 x3:     return $x;
  4: }
  5: sub t2 {
 x6:     return $x;
  7: }
 x8: $x->t1()->t2();
>>9: 3;
```

r N — step out few times #2

```
  1: sub t2 {
>>2:     3;
  3: }
  4: sub t1 {
1>5:     t2();
  6:     4;
  7: }
2>8: t1();
 x9: 5;
```

r N — step out few times #2

```
   1: sub t2 {
>>2:       3;
   3: }
   4: sub t1 {
1>5:       t2();
   6:       4;
   7: }
2>8: t1();
 x9: 5;
```

r N — step out few times #2

```
  1: sub t2 {
>>2:     3;
  3: }
  4: sub t1 {
1>5:     t2();
  6:     4;
  7: }
2>8: t1();
 x9: 5;
```

r N — step out few times #2

```
r 2

   1: sub t2 {
  >>2:     3;
   3: }
   4: sub t1 {
 1>5:     t2();
   6:     4;
   7: }
 2>8: t1();
  x9: 5;
```

r N — step out few times #2

```
  1: sub t2 {
>>2:     3;
  3: }
  4: sub t1 {
1>5:     t2();
  6:     4;
  7: }
2>8: t1();
 x9: 5;
```

r N — step out few times #2

```
 1: sub t2 {
x2:     3;
 3: }
 4: sub t1 {
x5:     t2();
 6:     4;
 7: }
x8: t1();
>>9: 5;
```

T — List frames

```
   1: sub t2 {
>>2:     3;
   3: }
   4: sub t1 {
1>5:     t2();
   6:     4;
   7: }
2>8: print t1( 33 );
 x9: 5;
```

T — List frames

T

```
   1: sub t2 {
>>2:       3;
   3: }
   4: sub t1 {
1>5:       t2();
   6:       4;
   7: }
2>8: print t1( 33 );
 x9: 5;
```

T — List frames

```
T

-1   main::t2
     ; ()
     <--  /home/kes/tmp/t.pl:5

-2   main::t1
     @ (33)
     <--  /home/kes/tmp/t.pl:8
```

T — List frames

```
T

-1  main::t2
    ; ()
    <--  /home/kes/tmp/t.pl:5

-2  main::t1
    @ (33)
    <--  /home/kes/tmp/t.pl:8
```

T — List frames

stack trace

```
   1: sub t2 {
>>2:       3;
   3: }
   4: sub t1 {
1>5:       t2();
   6:       4;
   7: }
2>8: print t1( 33 );
 x9: 5;
```

T — List frames

```
T

-1  main::t2
    ; ()
    <--  /home/kes/tmp/t.pl:5

-2  main::t1
    @ (33)
    <--  /home/kes/tmp/t.pl:8
```

T — List frames

```
T

-1  main::t2
    ; ()
    <--  /home/kes/tmp/t.pl:5

-2  main::t1
    @ (33)
    <--  /home/kes/tmp/t.pl:8
```

vars — List variables

```
vars a          # other params: m o g u c s

   x1: my $x =  { a => 7 };
   x2: my @z; our %h;
    3: sub t2 {
   x4:      my $x =  4;
   x5:      our $y =  7;
  >>6:      $x + scalar @z;
    7: }
  1>8: @z =  t2( @z, 33 );
```

vars — List variables

```
MY:
$x, @z

OUR:
$y, %h

GLOBAL:
$y, %h

USED:
$x, @z

CLOSED OVER:
@z
```

vars — List variables

```
MY:
$x, @z

OUR:
$y, %h

GLOBAL:
$y, %h

USED:
$x, @z

CLOSED OVER:
@z
```

vars — List variables

```
MY:
$x, @z

OUR:
$y, %h

GLOBAL:
$y, %h

USED:
$x, @z

CLOSED OVER:
@z
```

vars — List variables

```
MY:
$x, @z

OUR:
$y, %h

GLOBAL:
$y, %h

USED:
$x, @z

CLOSED OVER:
@z
```

vars — List variables

```
MY:
$x, @z

OUR:
$y, %h

GLOBAL:
$y, %h

USED:
$x, @z

CLOSED OVER:
@z
```

vars — List variables

```
$x              # or you can input: e $x

   x1: my $x =  { a => 7 };
   x2: my @z; our %h;
    3: sub t2 {
   x4:      my $x =  4;
   x5:      our $y =  7;
  >>6:      $x + scalar @z;
    7: }
  1>8: @z =  t2( @z, 33 );
```

vars — List variables

```
$x   #$x= 'new value' to chage value

   x1: my $x =  { a => 7 };
   x2: my @z; our %h;
    3: sub t2 {
   x4:      my $x =  4;
   x5:      our $y =  7;
  >>6:      $x + scalar @z;
    7: }
  1>8: @z =  t2( @z, 33 );
```

vars — List variables

vars -1 $x

```
  x1: my $x =  { a => 7 };
  x2: my @z; our %h;
   3: sub t2 {
  x4:      my $x =  4;
  x5:      our $y =  7;
>>6:      $x + scalar @z;
   7: }
1>8: @z =  t2( @z, 33 );
```

vars — List variables

```
vars -1 $x   N frames upper

   x1: my $x =  { a => 7 };
   x2: my @z; our %h;
    3: sub t2 {
   x4:      my $x =  4;
   x5:      our $y =  7;
  >>6:      $x + scalar @z;
    7: }
  1>8: @z =  t2( @z, 33 );
```

vars — List variables

```
vars -1 $x

    x1: my $x =  { a => 7 };
    x2: my @z; our %h;
     3: sub t2 {
    x4:      my $x =  4;
    x5:      our $y =  7;
  >>6:      $x + scalar @z;
     7: }
  1>8: @z =  t2( @z, 33 );
```

vars — List variables

```
vars -1 $x->{ a }

   x1: my $x =  { a => 7 };
   x2: my @z; our %h;
    3: sub t2 {
   x4:      my $x =  4;
   x5:      our $y =  7;
  >>6:      $x + scalar @z;
    7: }
  1>8: @z =  t2( @z, 33 );
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l [file:]<line> – at given file:line

l -<N> – list function atN frames up

l <function name> – list function

l $<variable> – deparse CODEREF

l &<N> – deparse function at given
frame
```

l — List source

```
l 0:8
     1: sub t2 {
    x2:         my $x =  1, 2; #ERROR
     3: }
     4: sub t1 {
    x5:         $z =  \&t2;
   >>6:         t2();
    x7:         $x + scalar @z + scalar
keys %h;
     8: }
   1>9: t1();
```

f — List loaded files

```
f <regex> – list files,
      Which names match <regex>

f t2
0 /home/kes/tmp/t2.pl
```

f — List loaded files

```
f <regex> – list files,
      Which names match <regex>

f t2
0 /home/kes/tmp/t2.pl
```

l — List source

```
l $z                    # Deparse CODEREF
{
    ((my $x = 1), 2);
}
l t2
/home/kes/tmp/t3.pl
     1: sub t2 {
    x2:     my $x =  1, 2;
     3: }
l &
sub main::t1 {
     ($z = (\&t2));
     t2();
     ($x + scalar((@z + scalar(keys(%h)))));
}
```

l — List source

```
l $z
{
    ((my $x = 1), 2);
}
l t2                    # List function by name
/home/kes/tmp/t3.pl
    1: sub t2 {
   x2:     my $x =  1, 2;
    3: }
l &
sub main::t1 {
    ($z = (\&t2));
    t2();
    ($x + scalar((@z + scalar(keys(%h)))));
}
```

l — List source

```
l $z
{
    ((my $x = 1), 2);
}
l t2
/home/kes/tmp/t3.pl
    1: sub t2 {
   x2:     my $x =  1, 2;
    3: }
l &       # Deparse current frame
sub main::t1 {
    ($z = (\&t2));
    t2();
    ($x + scalar((@z + scalar(keys(%h)))));
}
```

l — List source

```
l $z
{
    ((my $x = 1), 2);
}
l t2
/home/kes/tmp/t3.pl
    1: sub t2 {
   x2:     my $x =  1, 2;
    3: }
l &       # You can provide frame number: l &3
sub main::t1 {
    ($z = (\&t2));
    t2();
    ($x + scalar((@z + scalar(keys(%h)))));
}
```

go F:L — run until

go 6

```
   x1: my $x =  bless \$x;
    2: sub t1 {
   x3:     return $x;
    4: }
    5: sub t2 {
   x6:     return $x;
    7: }
  >>8: $x->t1()->t2();
    9: 3;
```

go F:L — run until

```
go t2

   x1: my $x =  bless \$x;
    2: sub t1 {
   x3:     return $x;
    4: }
    5: sub t2 {
  >>6:     return $x;
    7: }
   x8: $x->t1()->t2();
   x9: 3;
```

DB::state('dd',1) – debug debugger

```
/home/kes/tmp/t3.pl
    0: use Devel::DebugHooks::Terminal;
  >>1: my $x =  { a => 7 };
    2: sub t2 {
   x3:     my $x =  1, 2;
   x4:     1;
    5: }
    6: sub t1 {
   x7:     $z =  \&t2;
   x8:     t2();
   x9:     $x + scalar @z + scalar keys %h;
    10: }
   x11: t1();
```

DB::state('dd',1) – debug debugger

```
/home/kes/tmp/t3.pl
    0: use Devel::DebugHooks::Terminal;
  >>1: my $x =  { a => 7 };
    2: sub t2 {
   x3:     my $x =  1, 2;
   x4:     1;
    5: }
    6: sub t1 {
   x7:     $z =  \&t2;
   x8:     t2();
   x9:     $x + scalar @z + scalar keys %h;
    10: }
   x11: t1();

DB::state( 'dd', 1 )
```

DB::state('dd',1) – debug debugger

```
/home/kes/tmp/t3.pl
    0: use Devel::DebugHooks::Terminal;
  >>1: my $x =  { a => 7 };
    2: sub t2 {
   x3:     my $x =  1, 2;
   x4:     1;
    5: }
    6: sub t1 {
   x7:     $z =  \&t2;
   x8:     t2();
   x9:     $x + scalar @z + scalar keys %h;
    10: }
   x11: t1();

DB::state( 'dd', 1 )
1
```

DB::state('dd',1) – debug debugger

```
DB::state( 'dd', 1 )
1
s
```

DB::state('dd',1) – debug debugger

```
DB::state( 'dd', 1 )
1
s

/home/kes/lib/Devel/DebugHooks.pm
    106: sub process {
   x107:     BEGIN{ 'strict'->unimport( 'refs' )   ...
    108:
    109:
  >>110:     &{ $DB::options{ cmd_processor } .'::process' }( @_ );
    111: }
```

DB::state('dd',1) – debug debugger

```
DB::state( 'dd', 1 )
1
s

/home/kes/lib/Devel/DebugHooks.pm
    106: sub process {
   x107:     BEGIN{ 'strict'->unimport( 'refs' )   ...
    108:
    109:
  >>110:     &{ $DB::options{ cmd_processor } .'::process' }( @_ );
    111: }

s 9
```

DB::state('dd',1) – debug debugger

```
  567: ,s => sub {
>>568:     if( shift =~ m/^(\d+)$/ ) {
 x569:         my $handler =  DB::reg( 'stop', 'step' );
 x570:         $$handler->{ code } =  \&step_done;
 x571:         $$handler->{ steps_left } =  $1;
  572:     }
  573:
 x574:     $_->{ single }= 1 for @{ DB::state('stack') };
  575:
 x576:     return;
  577: }
```

DB::state('dd',1) – debug debugger

```
  567: ,s => sub {
 x568:   if( shift =~ m/^(\d+)$/ ) {
 x569:       my $handler =  DB::reg( 'stop', 'step' );
 x570:       $$handler->{ code } =  \&step_done;
 x571:       $$handler->{ steps_left } =  $1;
  572:   }
  573:
>>574:   $_->{ single }= 1 for @{ DB::state('stack') };
  575:
 x576:   return;
  577: }
```

e $DB::state – show debugger state

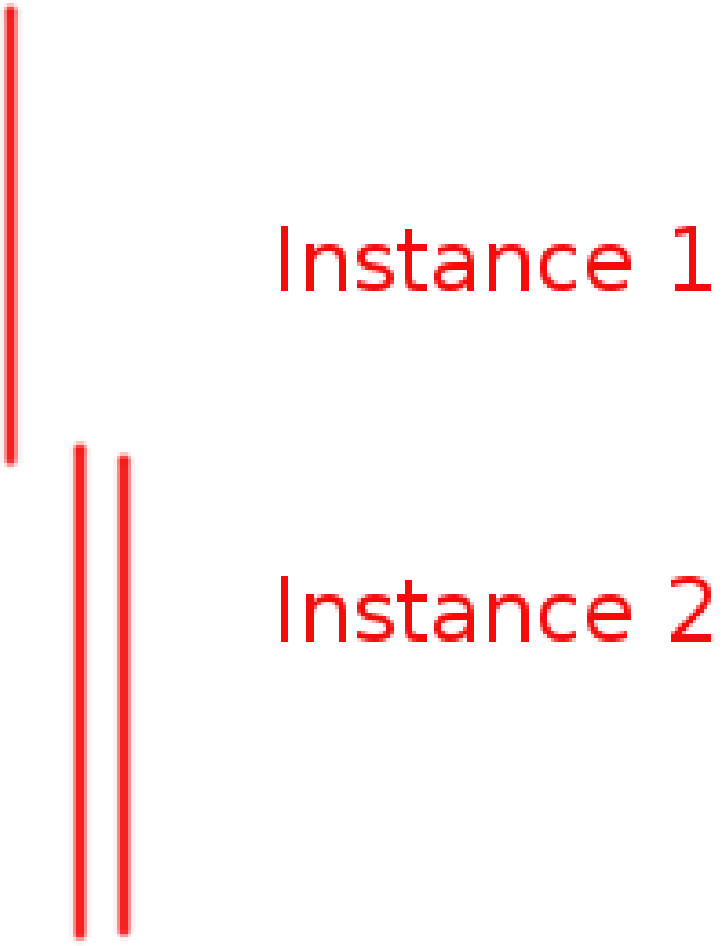
```
e $DB::state
```

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { file => t3.pl, line => 5, single => 1 }
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      ...
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```



e $DB::state – show debugger state

```
1. Global variables
    *DB::
```

e $DB::state – show debugger state

1. Global variables
   *DB::

2. Global variables for a copy
   DB::state('name1')

e $DB::state – show debugger state

1. Global variables
   *DB::

2. Global variables for a copy
   DB::state('name1')

3. Variables for current frame
   DB::state('name2')

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { file => t3.pl, line => 5, single => 1 }
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { file => t3.pl, line => 5, single => 1 }
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { file => t3.pl, line => 5, single => 1 }
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```

e $DB::state – show debugger state

```
e $DB::state
[
  {
    "dd" => 1,          # DB::state( 'dd', 1 )
    "inDB" => 1,
    ...
    "stack" => [
      { file => t3.pl, line => 5, single => 1 }
    ],
  },{
    "cmd" => 1,
    "inDB" => 1,
    ...
    "stack" => [
      { ... },
      { ... },
    ],
  },
]
```

e $DB::state – show debugger state

Main logic is inside next function:

DB::state

e $DB::state – show debugger state

Main logic is inside next function:
DB::state

```
  $DB::variables =  {()
    ,'*'              =>  \&dbg_vrbl
    ,single           =>  \&int_vrbl
    ,line             =>  \&frm_vrbl
    ...
  };
```

e $DB::state – show debugger state

Main logic is inside next function:
DB::state

```
  $DB::variables =  {()
     ,'*'                 =>  \&dbg_vrbl
     ,single              =>  \&int_vrbl
     ,line                =>  \&frm_vrbl
     ...
  };
```

e $DB::state – show debugger state

Main logic is inside next function:
DB::state

```
  $DB::variables =  {()
     ,'*'                 =>  \&dbg_vrbl
     ,single              =>  \&int_vrbl
     ,line                =>  \&frm_vrbl
     ...
  };
```

Summary

- OpenSource gives easy start

Summary

- OpenSource gives easy start
- OpenSource not asking for money

Summary

- OpenSource gives easy start
- OpenSource not asking for money
- OpenSource saves your time

Summary

- OpenSource gives easy start
- OpenSource not asking for money
- OpenSource saves your time
- OpenSource allows to earn

Summary

- OpenSource gives easy start
- OpenSource not asking for money
- OpenSource saves your time
- OpenSource allows to earn

OpenSource supports you

Summary

- OpenSource gives easy start
- OpenSource not asking for money
- OpenSource saves your time
- OpenSource allows to earn

OpenSource supports you

Support OpenSource

Statistics for Devel::DebugHooks

Done 1203 commits

Written ~200Kbytes of code

Spent ~1 Year

Questions?