Error Handling

Adám Brudzewsky

Error Handling

Adám Brudzewsky

Error Handling – Part 1

Number for given message
aplcart.info?q=:: message

Tradfn structure `:Trap` errnos (0 means all)

Dfn error guard errnos`::`code

Error Number of last error `⎕EN`

Error Message for number `⎕EM`

Error Handling – Part 2

Construct error message
aplcart.info?q=dmx 1

Execute there on error	`⎕TRAP←errnos 'E' code`
Cut back and execute	`⎕TRAP←errnos 'C' code`
Ext'd Diagnostic Message	`⎕DMX`

Error Handling – Part 2

Construct error message
aplcart.info?q=dmx 1

Execute there on error `⎕TRAP←errnos 'E' code`

Cut back and execute `⎕TRAP←errnos 'C' code`

Ext'd Diagnostic Message `⎕DMX`

dyalog.tv

Overview

- Signalling an Error

Overview

- Signalling an Error
- Assertions

Overview

- Signalling an Error
- Assertions
- Resignalling an Error

Signalling an Event

```
      M←{
          (+≠⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
```

Signalling an Event

```
      M←{
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ⍬
1
```

Signalling an Event

```
      M←{
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M 0
DOMAIN ERROR
      M 0
      ^
```

Signalling an Event

```
      M←{
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M 0
DOMAIN ERROR: Divide by zero
      M 0
      ^
```

Signalling an Event

```
      M←{
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ⍬
MEAN ERROR: Empty 1st axis
      M ⍬
      ^
```

Signal Error

system function

```
⎕SIGNAL
```

Signal Error

`⎕SIGNAL` scalar

Signal Error

```
⎕SIGNAL scalar
```

Signal Error

```
⎕SIGNAL ┌─── scalar ───┐
        └─── errno ────┘
```

Signal Error

⎕SIGNAL scalar

errno

⊂name-value pairs

Signal Error

⎕SIGNAL scalar

errno

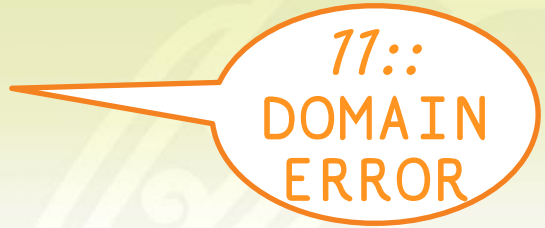
⊂name-value pairs

Signal Error

```
      M←{
          0=≢⍵: ⎕SIGNAL 11
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ⍬
DOMAIN ERROR
      M ⍬
      ^
```

Signal Error

```
      M←{
          0=≢⍵: ⎕SIGNAL 11
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M 0
DOMAIN ERROR
      M 0
      ^
```



Signal Error

⎕SIGNAL ┌─── scalar ───┐

errno

⊂name-value pairs

Signal Error

```
⎕SIGNAL ┌─── scalar ───┐
        │     errno    │
        ⊂name-value pairs
```

Signal Error

name-value pairs

Signal Error

name-value pairs

Extended Diagnostic Message

```
⎕JSON⍠'Compact'0⊢⎕DMX
```

name-value pairs

Extended Diagnostic Message

```
      ⎕JSON⍠'Compact'0⊢⎕DMX
{
  "Category": "General",
  "DM": [
    "DOMAIN ERROR",
    "      1÷0",
    "        ^"
  ],
  "EM": "DOMAIN ERROR",
  "EN": 11,
  "ENX": 1,
  "HelpURL": "https://help.dyalog.com/dm
```

name-value pairs

Extended Diagnostic Message

```
      ⎕JSON⍠'Compact'0⊢⎕DMX
{
  "Category": "General",
  "DM": [
    "DOMAIN ERROR",
    "      1÷0",
    "       ^"
  ],
  "EM": "DOMAIN ERROR",
  "EN": 11,
  "ENX": 1,
  "HelpURL": "https://help.dyalog.com/dm
```

name-value pairs

Extended Diagnostic Message

```
      ⎕JSON⍠'Compact'0⊢⎕DMX
{  names      values
   "Category": "General",
   "DM": [
                "DOMAIN ERROR",
                "      1÷0",
                "       ^"
   ],
   "EM":       "DOMAIN ERROR",
   "EN":       11,
   "ENX":      1,
   "HelpURL":  "https://help.dyalog.com/d
```

name-value pairs

Signal Error

name-value pairs

Signal Error

name-value pairs

Signal Error

name-value pairs

Signal Error

```
⎕SIGNAL ┌──── scalar ────┐
        │      errno     │
        ⊂name-value pairs⊃
```

```
'EN'       11
'EM'       'MEAN ERROR'
'Message'  'Empty 1st axis'
```

Signal Error

```
⎕SIGNAL ┌─── scalar ───┐
        │    errno     │
        ⊂name-value pairs
'EN'        11
'EM'        'MEAN ERROR'
'Message'   'Empty 1st axis'
```

Signal Error

```
⎕SIGNAL⊂('EN' 11)('EM' 'MEAN ERROR')('Mes
```

Signal Error

```
M←{
    0=≢⍵:⎕SIGNAL⊂('EN' 11)('EM' 'MEAN ERROR
    (+⌿⍵)÷≢⍵
}
```

Signal Error

```
      M←{
          0=≢⍵:⎕SIGNAL⊂('EN' 11)('EM' 'MEAN ERROR
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ⍬
MEAN ERROR: Empty 1st axis
      M ⍬
      ^
```

use-cases

for

⍞SIGNAL

Assert

utility function

```
{ ... ⎕SIGNAL ... }
```

Assert

utility function

```
{ ... ⎕SIGNAL ... }
```

aplcart.info

Assert

utility function

```
{ ... ⎕SIGNAL ... }
```

aplcart.info

18·1:
]APLcart

⍋⍒ Tell me about:

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

⍬	Empty Numeric Vector
⊢Y	Same: Y
X dop Y⊢Z	Separate dyadic operator's right operand from its right argument (same as (X dop Y)Z)
X⊢Y	Right: Y
X⊢Y	Church Boolean false (X if false, else Y)
⊣Y	Same: Y
X⊣Y	Left: X
X⊣Y	Church Boolean true (X if true, else Y)
+Y	Conjugate ('Identity' if Y not complex)
+N	Mirror complex N across x-axis
M+N	Adding N to M

Showing 2571 of 2571

Quiz

Tell me about: assert

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

```
{α←'assertion failure' ⋄ 0∊ω:α ⎕SIGNAL 8 ⋄ shy←0}B
```

Signal an error if any condition is not met (optional left argument: message on failure)

Tell me about: assert

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

```
{α←'assertion failure' ⋄ 0∊ω:α ⎕SIGNAL 8 ⋄ shy←0}B
```

Signal an error if any condition is not met (optional left argument: message on failure)

Assert

utility function

```
{⍺←'assertion failure' ⋄
 0∊⍵:⍺ ⎕SIGNAL 8 ⋄
 shy←0}
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

```
      2×{shy←⍵}3
```

```
shy←0}
```

```
      2×{shy←⍵}3
6
```

```
shy←0}
```

```
      2×{shy←⍵}3
6
      2×{shy←⍵ ⋄ }3
```

```
shy←0}
```

```
      2×{shy←⍵}3
6
      2×{shy←⍵ ⋄ }3
VALUE ERROR: No result was provided
when the context expected one
      2×{shy←⍵ ⋄ }3
        ^
```

```
shy←0}
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

```
α ⎕SIGNAL 8
```

Signal Error

```
⎕SIGNAL⊂('EN' 11)('EM' 'MEAN ERROR')
```

```
⍺ ⎕SIGNAL 8
```

Signal Error

```
'MEAN ERROR' ⎕SIGNAL 11
      ⍺ ⎕SIGNAL 8
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

```
2{⍺←10 ⋄ ⍺×⍵}3
```

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

```
      2{⍺←10 ⋄ ⍺×⍵}3
6
```

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

```
      2{α←10 ⋄ α×ω}3
6
      {α←10 ⋄ α×ω}3
```

```
{α←'assertion failure'
 0∈ω:α ⎕SIGNAL 8
 shy←0}
```

```
      2{α←10 ⋄ α×ω}3
6
      {α←10 ⋄ α×ω}3
30
```

```
{α←'assertion failure'
 0∊ω:α ⎕SIGNAL 8
 shy←0}
```

Assert

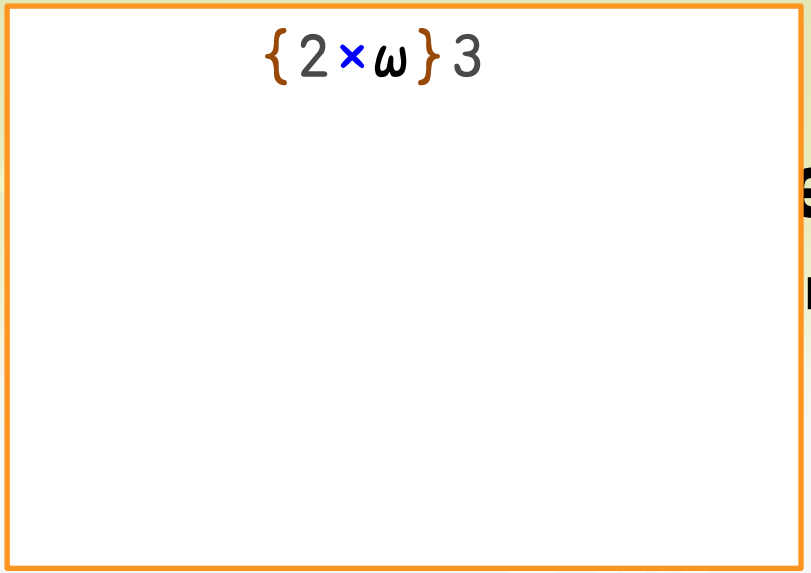
utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

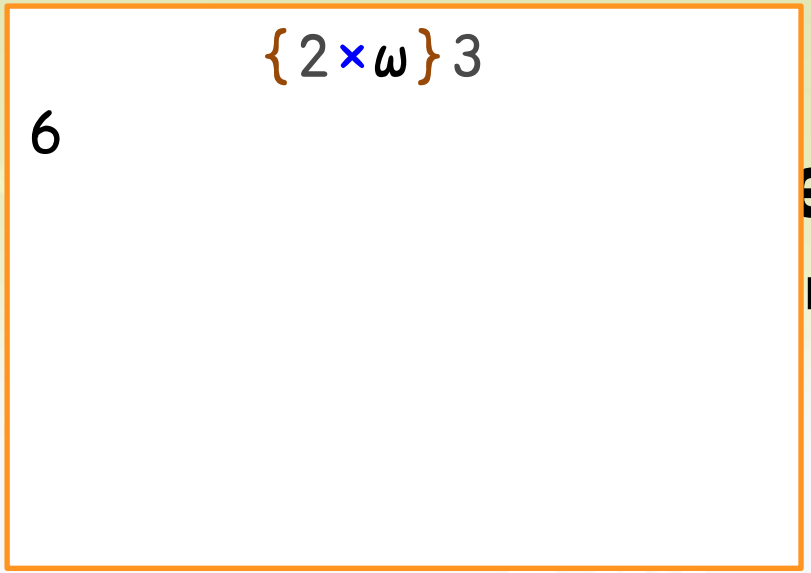


ert

nction

failure'
8

shy←0}



ert

nction

failure'
8

shy←0}

```
      {2×⍵}3
6

      {r←2×⍵}3
```

ert

nction

failure'
8

shy←0}

```
      {2×⍵}3
6

      {r←2×⍵}3
      {1+r←2×⍵}3
```

ert

nction

failure'
8

shy←0}

```
      {2×⍵}3
6
      {r←2×⍵}3
      {1+r←2×⍵}3
7
```

ert

nction

failure'
8

shy←0}

```
      {2×⍵}3
6
      {r←2×⍵}3
      {1+r←2×⍵}3
7
      1+{r←2×⍵}3
```

ert

nction

failure'
8

shy←0}

```
      {2×⍵}3
6
      {r←2×⍵}3
      {1+r←2×⍵}3
7
      1+{r←2×⍵}3
7
```

ert

nction

failure'
8

shy←0}

Assert

utility function

```
{⍺←'assertion failure'
 0∊⍵:⍺ ⎕SIGNAL 8
 shy←0}
```

Assert

utility function

```
{⍺←'assertion failure' ⋄
 0∊⍵:⍺ ⎕SIGNAL 8 ⋄
 shy←0}
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≢⍵:
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ¯⍵
MEAN ERROR
      M ¯⍵
        ^
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
              {0 ◇ ⍵}3

2.8

MEAN
      ^
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
                  {0 ◇ ⍵}3
      0

2.8

MEAN

      ^
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
                  {0 ◇ ⍵}3
0
                  {0: ◇ ⍵}3
2.8

MEAN
          ^
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
                  {0 ◇ ⍵}3
0
                  {0: ◇ ⍵}3
2.8
3
MEAN
          ^
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
          {0 ◇ ⍵}3
      0
          {0: ◇ ⍵}3
      3
          {_←2 ◇ ⍵}3
      ^
2.8
MEAN
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≠⍵:
          {0 ◇ ⍵}3
      0
          {0: ◇ ⍵}3
      3
          {_←2 ◇ ⍵}3
      3
      }
2.8
      ^
MEAN
```

Assert: Use-case

```
      M←{
          'MEAN ERROR' Assert 0<≢⍵:
          (+⌿⍵)÷≢⍵
      }
      M 3 1 4 1 5
2.8
      M ¯⍵
MEAN ERROR
      M ¯⍵
        ^
```

Assert: Use-case

```
      ∇   z←M y
          'MEAN ERROR' Assert 0<≢y
          (+⌿y)÷≢y
      ∇
      M 3 1 4 1 5
2.8
      M ¯3
MEAN ERROR
      M ¯3
        ^
```

Assert: Use-case

```
SsetTest ← {
    Assert 8 = sset 3:
    Assert 551872 = sset 857:
    Assert 935424 = sset 870:
    'Test passed'
}
```

Assert: Use-case

is.gd/apl2020

```
SsetTest ← {
    Assert 8 = sset 3:
    Assert 551872 = sset 857:
    Assert 935424 = sset 870:
    'Test passed'
}
```

Resignal Error

utility function

```
⎕SIGNAL{ ... }
```

Resignal

```
Lines←{
    text←⊃⎕NGET ⍵ 1
    ≠text
}
```

Resignal

```
Lines '/tmp/my.txt'
```

Resignal

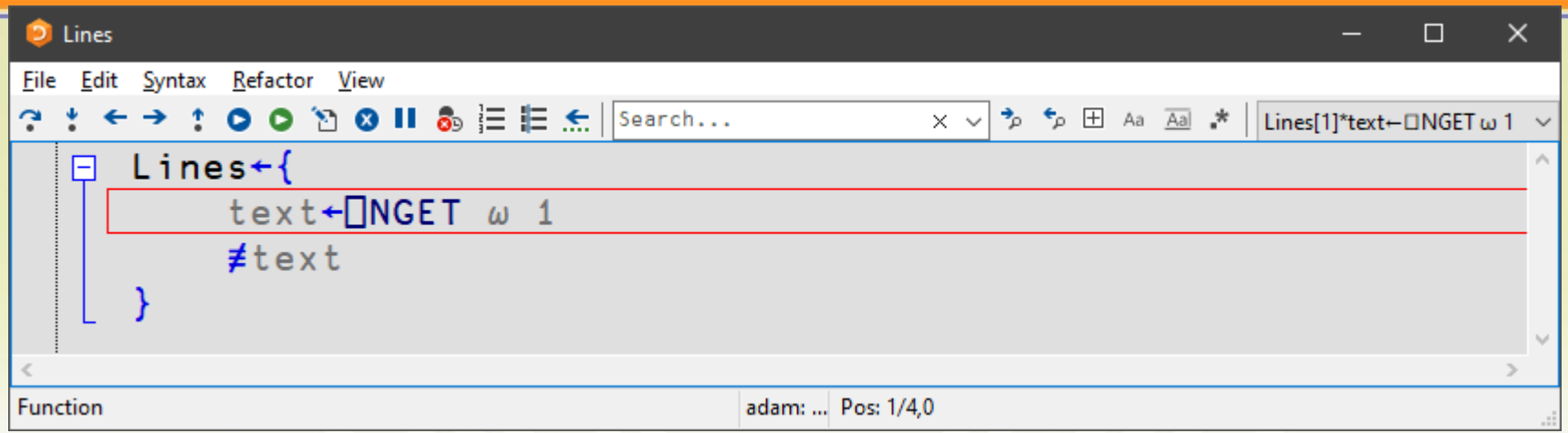
```
      Lines '/tmp/my.txt'
42
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
FILE NAME ERROR: /tmp/no.txt: Unable to open ("T
Lines[1] text←⎕NGET ⍵ 1
                  ^
```




```
FILE NAME ERROR: /tmp/no.txt: Unable to open ("T
Lines[1] text←⎕NGET ⍵ 1
                   ^
```

Resignal

```
Lines←{
      0::⎕SIGNAL⍵
      text←⊃⎕NGET ⍵ 1
      ≢text
}
```

Resignal

```
Lines←{
      0::⎕SIGNAL ⍵
      text←⊃⎕NGET ⍵ 1
      ≢text
```

Resignal

```
Lines←{
      0::⎕SIGNAL ⍵
      text←⊃⎕NGET ⍵ 1
      ≢text
```

0:: ALL ERRORS

aplcart.info

⍋⍒ Tell me about:

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

⍬	Empty Numeric Vector
⊢Y	Same: Y
X dop Y⊢Z	Separate dyadic operator's right operand from its right argument (same as (X dop Y)Z)
X⊢Y	Right: Y
X⊢Y	Church Boolean false (X if false, else Y)
⊣Y	Same: Y
X⊣Y	Left: X
X⊣Y	Church Boolean true (X if true, else Y)
+Y	Conjugate ('Identity' if Y not complex)
+N	Mirror complex N across x-axis
M+N	Adding N to M

Showing 2571 of 2571

Quiz

Tell me about: resignal

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

```
⎕SIGNAL⊂⎕DMX.(('EN'EN)('EM'EM)
('Message'(OSError{⍵,2⌽(×≠⊃⍬⍴2⌽⍺,⊂'')/''}
("',⍵←⊃⍬⍴2⌽⍺}Message)))
```

Re-signal last caught error to caller (works with any ⎕IO and ⎕ML)

Showing 1 of 2571

Quiz

Tell me about: resignal

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

```
⎕SIGNAL⊂⎕DMX.(('EN'EN)('EM'EM)
('Message'(OSError{⍵,2⍕(×≢⊃⍬⍴2⍕⍺,⊂'')/''}
('',o←⊃⍬⍴2⍕⍺}Message)))
```

Re-signal last caught error to caller (works with any ⎕IO and ⎕ML)

Showing 1 of 2571

Quiz

Resignal

```
Lines←{
    0::⎕SIGNAL⊂⎕DMX.(('EN'EN)('EN'EM)('Me
    text←⊃⎕NGET ⍵ 1
    ≢text
}
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines'/tmp/no.txt'
      ^
```

Resignal

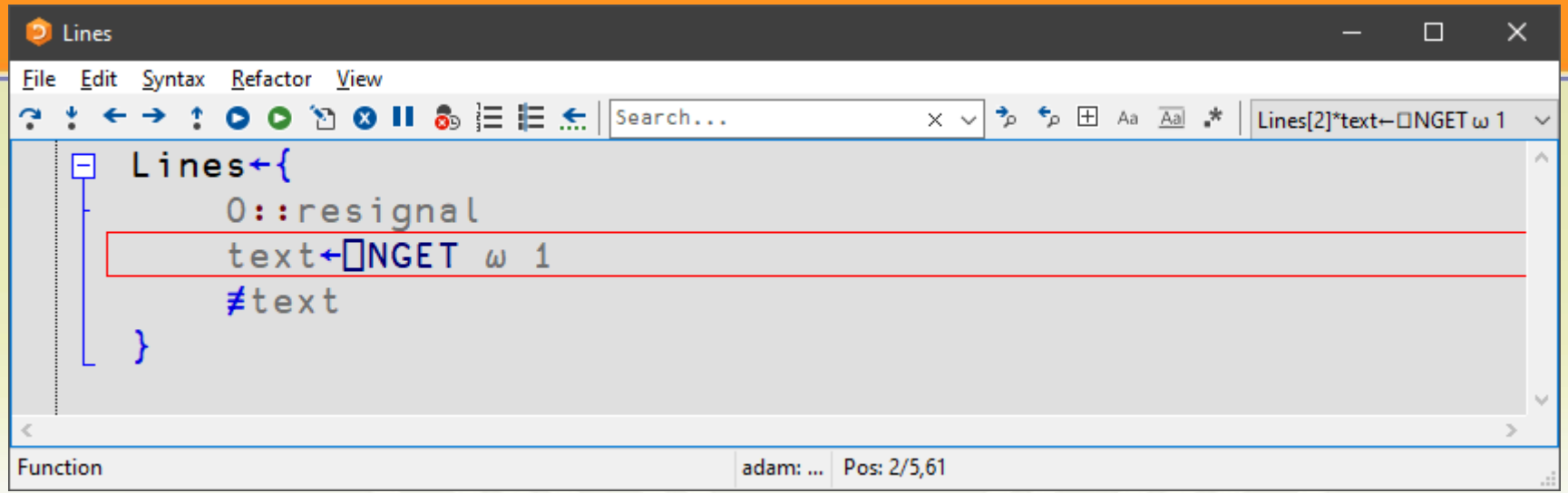
```
Lines←{
      0::resignal
      text←⊃⎕NGET ⍵ 1
      ≢text
}
∇ resignal
  ⎕SIGNAL⊂⎕DMX.(('EN'EN)
∇
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines[1] 0::resignal
                         ^
```




```
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines[1] 0::resignal
                ^
```

Resignal

```
Lines←{
      0::resignal
      text←⊃⎕NGET ⍵ 1
      ≢text
}
∇ resignal
  ⎕SIGNAL⊂⎕DMX.(('EN'EN)
∇
```

Resignal

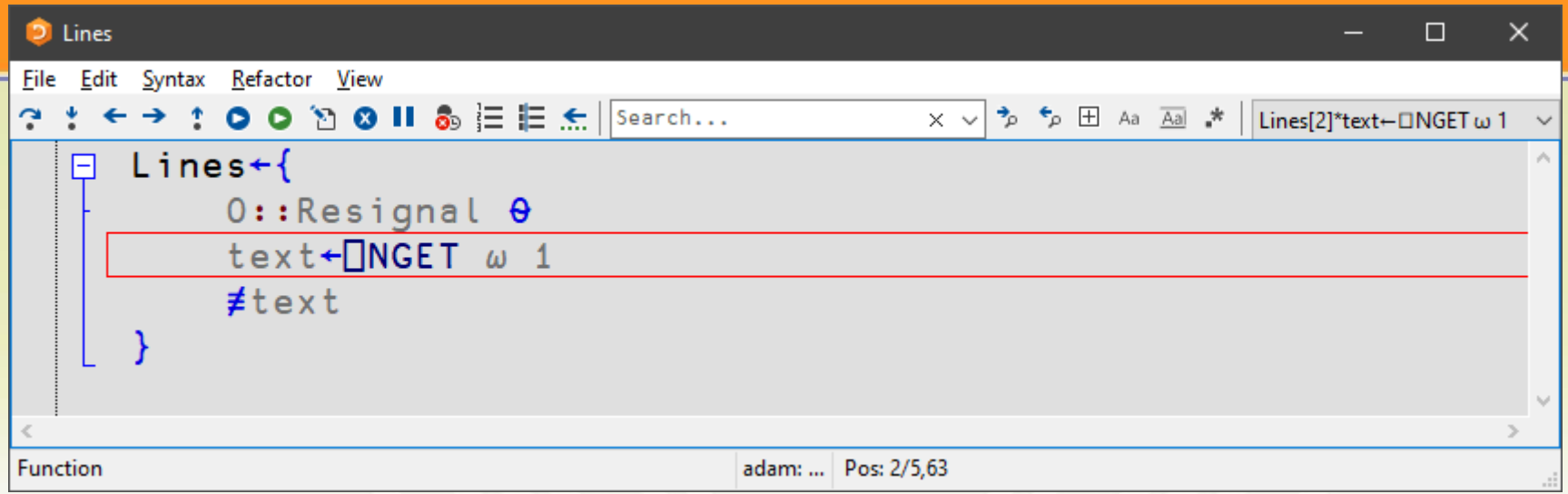
```
Lines←{
      0::Resignal ⍬
      text←⊃⎕NGET ⍵ 1
      ≢text
}
Resignal←{⎕SIGNAL⊂⎕DMX.((('EN'EN)
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines[1] 0::Resignal 0
                           ^
```




```
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines[1] 0::Resignal ⍬
               ^
```

Resignal

```
Lines←{
      0::Resignal ⍬
      text←⊃⎕NGET ⍵ 1
      ≢text
}
Resignal←{⎕SIGNAL⊂⎕DMX.(
```

Resignal

```
Lines←{
      0::Resignal ⍬
      text←⊃⎕NGET ⍵ 1
      ≢text
}
Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
      Foo←{Goo ⍬}
```

```
      ) 1
```

```
      Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
Foo←{Goo ⍬}
Goo←{⎕SI}
```

```
      1

Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
Foo←{Goo 0}
Goo←{⎕SI}
```

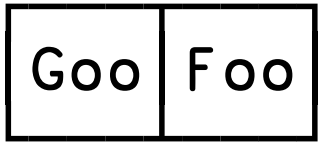
Stack Indicator

```
      1

Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
Foo←{Goo ⍬}
Goo←{⎕SI}
Foo ⍬
```

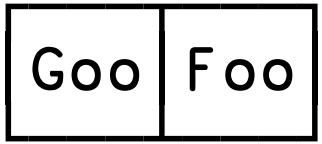


Stack Indicator

```
Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
Foo←{Goo ⍬}
Goo←{⎕SI}
Foo ⍬
```



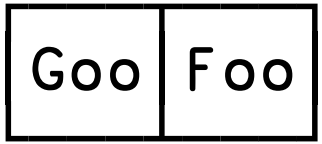
Stack Indicator

```
Foo←{Goo ⍬}
Goo←⍎⍕'⎕SI'⊢⍕#
```

```
Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
Foo←{Goo ⍬}
Goo←{⎕SI}
Foo ⍬
```



Stack Indicator

```
Foo←{Goo ⍬}
Goo←⍎⍕'⎕SI'⊢⍕#
Foo ⍬
```


```
Resignal←⎕SIGNAL{⊂⎕DMX.(
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
```

Resignal

```
      Lines '/tmp/my.txt'
42
      Lines '/tmp/no.txt'
FILE NAME ERROR: /tmp/no.txt: Unable to open fil
      Lines'/tmp/no.txt'
      ^
```

Error Handling – Part 3

Signal an error: `⎕SIGNAL errno`

Signal a custom error: `⎕SIGNAL⊂name-value pairs`

Assert: aplcart.info?q=assert

Resignal: aplcart.info?q=resignal

Error Handling – Part 3

Press F1 for full docs!

Signal an error: ⎕SIGNAL errno

Signal a custom error: ⎕SIGNAL⊂name-value pairs

Assert: aplcart.info?q=assert

Resignal: aplcart.info?q=resignal

Upcoming webinars

Jun	17	BAA	open session
Jun	24	- - -	
July	1	BAA	open session
July	8	Dyalog	TBD
July	15	BAA	open session
July	22	- - -	
July	29	BAA	open session
Aug	5	Dyalog	Error handling – pt. 4

Upcoming webinars

Jun	17	BAA	open session
Jun	24	- - -	
July	1	BAA	open session
July	8	Dyalog	TBD
July	15	BAA	open session
July	22	- - -	
July	29	BAA	open session
Aug	5	Dyalog	Error handling – pt. 4

One open session will be AGM (Annual General Meeting)
For details, check BAA's site:
british**a**pl**a**ssociation.org