

## Dyalog'16 Workshop SA2: Threading and Synchronisation

### Exercises

Our application is a “Keyed Array” function which allow you to store data in an array and retrieve it using an arbitrary key. The code can also be found in KA.dyalog in the workshop root folder.

```

    ▽ r←{data}KA key;i
[1]  A Keyed Array
[2]  :If 0=⊖NC'KEYS' ⋄ KEYS←DATA←⊖ ⋄ :EndIf
[3]  :If 0=⊖NC'data' A Query
[4]      i←KEYS⊖key
[5]      r←i>DATA
[6]  :Else A Upsert
[7]      :If (≠KEYS)≥r←KEYS⊖key
[8]          (r>KEYS)←data
[9]      :Else
[10]         KEYS,←key
[11]         ⊖DL 0.001 A increase likelihood of a problem
[12]         DATA,←data
[13]      :EndIf
[14]      :If data≠(KEYS⊖key)>DATA
[15]         ○○○ A KA damaged
[16]      :EndIf
[17]  :EndIf
    ▽
```

In order to make it easy to verify whether the keyed array is in good shape, you might decide to always use the same left and right argument:

```

    1 2 3 KA`1 2 3
1 2 3
    KEYS≡DATA
1
```

#### Exercise 0: Break it

Demonstrate that the function is not thread safe, by invoking it in parallel using &`.

#### Exercise 1: Fix KA Using :Hold

Insert an :Hold :EndHold pair of statements to make KA thread safe, and test it.

#### Exercise 2: Fix KA Using ⊖TPUT/⊖TGET

Discussion: :Hold vs TGET, and where they go.

### Exercise 3: Use a Latch

Create a Latch. Launch a thread which waits on it and then displays some output before terminating. Open the Latch.

### Exercise 4: Use a Gate

Create a Gate. Launch a thread which waits on it and updates a global variable COUNT every second while the Gate is open. Open an editor window on COUNT, then open and close the gate.

### Exercise 5: Fix KA using a Mutex

Take out the :Hold and insert a Mutex instead, and verify it is still using.

### Exercise 6: Use a Queue

Launch a thread which reads from a Queue and displays what it gets. Insert some data into a Queue and verify that it comes out in the same order.

### Exercise 7: Use a SynchObject

Create a SyncObject, launch a thread which waits 10 seconds and then Sets it. Start waiting for the result and verify that it arrives after roughly the right amount of time.

### Exercise 8: Use a ReadWrite

If we want to be able to remove items from our Keyed Array, reading threads also need protection. Add a Delete mechanism to remove keys from the keyed array, and then use a ReadWrite so that readers only take out a Read Lock, while updates acquire a Write Lock.

### Exercise 9: Update KF to use `□FHOLD`

Update the KF.IO function to use `□FHOLD` and verify that it is now able to run in parallel.

Hints: Look at Demos\futiso.txt for expressions which can be used to launch multiple KFTest calls in parallel.