Component Files

- Require tying and untying
- Structurally similar to a nested vector
- Components have a "component number"
- We can only append, replace or drop components
- Multiple users can access at the same time using shared ties
- We can control multi-user access

See the documentation for a list of system functions used with component files (http://help.dyalog.com/17.1/#Language/APL%20Component%20Files/Component%20Files.htm).

In [12]:
```
cfile ← '\tmp\my_component_file'
```

Old Dyalog versions use **.DCF** extension.

Newer versions use **.dcf** extension.

We do not need to provide a file extension to ⎕FTIE and ⎕FCREATE.

Create, tie and untie

Note: Tied files are *not* untied when we use ⎕LOAD,)LOAD or)CLEAR.

In [13]:
```
tn ← cfile ⎕FCREATE 0
⎕←⎕FNUMS   ⍝ List tied files
```

Out[13]:
```
1
```

In [14]:
```
⎕←⎕FUNTIE tn
```

Out[14]:
```
1
```

Reading and writing

Component files have a similar structure to **nested vectors**

In [15]:
```
⎕←nested_vector ← 'first' (2 2⍴⍳4) '3rd'
```

Out[15]:
```
┌─────┬───┬───┐
│first│1 2│3rd│
│     │3 4│   │
└─────┴───┴───┘
```

We can only append components and drop components. This is like if we could only catenate or drop from our nested vector.

In [21]:
```
nested_vector,←'new component'
⎕←nested_vector
```

Out[21]:
```
new component
```

In [ ]:
```
nested_vector ↓← 2
⎕←nested_vector
```

In [22]:
```
nested_vector,←('we can' 'include' (2 3⍴-⍳6) 'any array')
```

In [16]:
```
tn ← cfile ⎕FTIE 0
'first' ⎕FAPPEND tn
(2 2⍴⍳4)'3rd' ⎕FAPPEND¨tn
```

To read a component from the file, you must specify a component number.

In [18]:
```
⎕FREAD tn 1   ⍝ Read the first component
```

Out[18]:
```
first
```

In [19]:
```
⎕FREAD¨tn,¨⍳3
```

Out[19]:
```
┌─────┬───┬───┐
│first│1 2│3rd│
│     │3 4│   │
└─────┴───┴───┘
```

Of course, the components you try to read must exist:

In [20]:
```
⎕FREAD tn 4
```

```
FILE INDEX ERROR: \tmp\my_component_file.dcf: No such component
      ⎕FREAD tn 4
      ^
```

Replacing and dropping components

We can replace any existing component in a file with a new array.

In [ ]:
```
```

```
(3 2⍴'new' 'component' (2 2⍴3↑1))⎕FREPLACE tn 2
⎕FREAD tn 2
```

`⎕FDROP` behaves similarly to `α↓ω`

In [23]:
```
⎕FDROP tn 1   ⍝ Drop 1 from the beginning
⎕FREAD tn 1
```

```
FILE INDEX ERROR: \tmp\my_component_file.dcf: No such component
      ⎕FREAD tn 1
      ^
```

How do I know which component numbers are used?

`⎕FSIZE` shows:

- The number of the first component in the file
- The number of the next new component (1 + number of the last component)
- The current file size in bytes
- The maximum file size in bytes (usually a very big number on modern systems)

In [25]:
```
⎕FSIZE tn
```

Out[25]:
```
2 4 1456 1.84467E19
```

In [24]:
```
⎕FREAD¨tn,¨2 3
```

Out[24]:
```
┌─────┬───┐
│1 2  │3rd│
│3 4  │   │
└─────┴───┘
```

Dropping too many components leaves an empty file:

In [26]:
```
⎕FDROP tn ¯2   ⍝ Drop two from the end
⎕FSIZE tn
```

Out[26]:
```
2 2 1456 1.84467E19
```

In [27]:
```
⎕FREAD tn 2
```

```
FILE INDEX ERROR: \tmp\my_component_file.dcf: No such component
      ⎕FREAD tn 2
      ^
```

Multi-user access

Use a shared file tie so multiple users can access at the same time.

A user is identified with an account number. This is both the *configuration parameter* (http://course.dyalog.com/autumn2021/Interpreter-internals/#configuration-parameters) called **APLNID** and also it is the first element of `⎕AI` (account information.

The default user is user `0`.

In [28]:
```
⎕←2⎕NQ'.' 'GetEnvironment' 'APLNID'
```

Out[28]:
```
0
```

In [29]:
```
⎕AI
```

Out[29]:
```
0 2281 5115337 5112951
```

In order to simulate a separate user, you can set an environment variable.

Let us pretend to be a user with ID 42.

On macOS and Linux command lines:

```
$ dyalog APLNID=42
```

On Microsoft Windows command prompt:

```
> set APLNID=42
> "C:\Program Files\Dyalog\Dyalog APL-64 17.1 Classic\dya
log.exe"
```

On Powershell:

```
PS $Env:APLNID=42
PS & 'C:\Program Files\Dyalog\Dyalog APL-64 17.1 Classic\
dyalog.exe'
```

Use `⎕FSTIE` for shared file ties. If you use `⎕FTIE` (exclusive tie), then no other user will be able to access the file at the same time.

Holding a file

"Holding" a file means that access to the file by other processes is not allowed until the **hold** is released. This is done using `⎕FHOLD`.

An example to demonstrate component file holds is provided below:

In [2]:

```
∇ FHold file;tn;cn;e;s
[1]     tn←file ⎕FSTIE 0
[2]     ⎕←'File ',file,' tied.'
[3]     ⎕←⎕FHOLD tn
[4]     ⎕←'File ',file,' held...'
[5]     (s e)←2↑⎕FSIZE tn
[6]     cn←¯1+s+⍳e-s
[7]     ⎕←⎕FREAD tn, cn
[8]     ⎕←⎕DL 3
[9]     ⎕←⎕FHOLD tn
[10]    ⎕←'File ',file,' released.
[11]    ⎕FUNTIE tn
[12]    ⎕←'File ',file,' untied.'
      ∇
```

Access matrices

Specific control of files (ability to read, write, append, delete etc.) is configured using **access matrices**.

See also the help for ⎕FRDAC (https://help.dyalog.com/17.1/index.htm#Language/System%20Functions/frdac.htm) and ⎕FSTAC (https://help.dyalog.com/17.1/index.htm#Language/System%20Functions/fstac.htm).

See Chapter N section 1.3.3 (https://www.dyalog.com/uploads/documents/MasteringDyalogAPL.pdf#%5B%7B%22num%22%3A1071%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C40%2C367%2C0%5D) of Mastering Dyalog APL.