

DIALOG

Elsinore 2023

D02 - The Road Ahead

Morten Kromberg, CTO

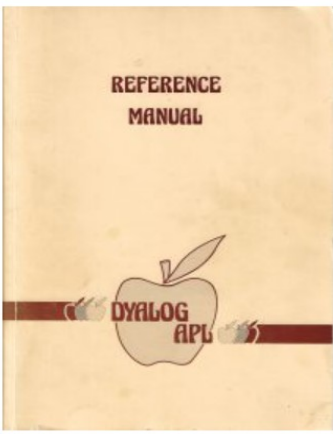
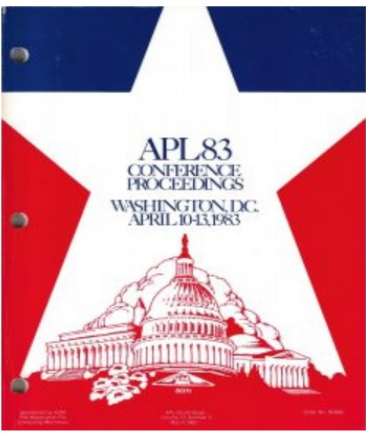


11 April 2023 – A Day to Celebrate!

Today we reach two very significant milestones.

40 Years of Dyalog APL

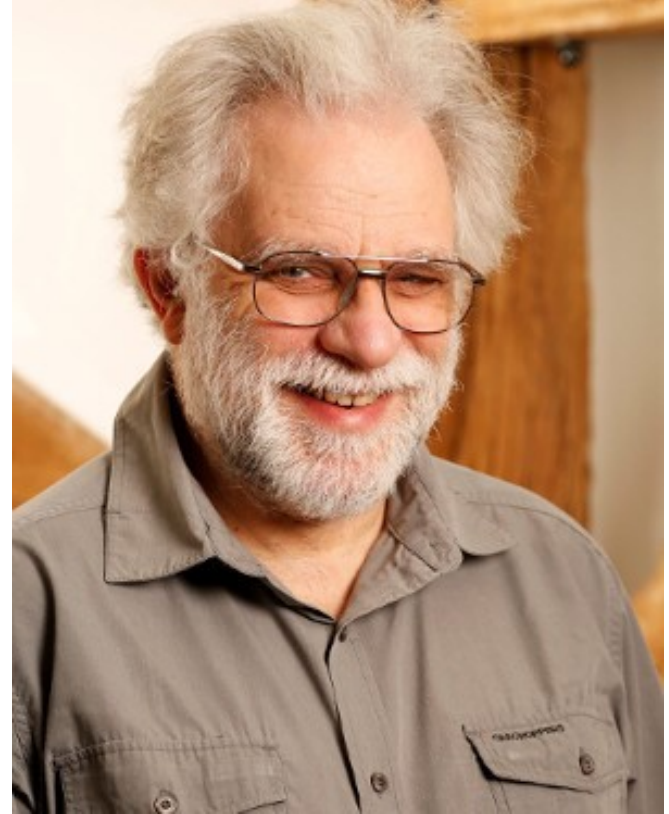
On this day, we have cause for celebration: it is 40 years since the release of Dyalog version 1.0! Geoff Streater would say that from his perspective we are already in the 42nd year, as he and John Scholes started work on the new interpreter in 1981. On the other hand, Pete Donnelly might argue that the interpreter wasn't really ready for serious use until a few years after that date. The fact remains that the APL '83 conference in Washington DC saw the first official release of the product, and is considered to be the "birth" of Dyalog APL.



Farewell Geoff Streater

Geoff has Retired (!)

- ◆ With John Scholes, Geoff Streeter implemented Dyalog APL v1.0 in 1981-1983
- ◆ We hope to welcome Geoff back for a retrospective talk at Dyalog'24 in (September 15-19, Glasgow)



40 years in pursuit of excellence

Decade	Leaders in...
1980's	The best APL for Workstations (X-Windows, □SM)
1990's	The best APL for Microsoft Windows (□WC)
2000's	The best APL for Microsoft .NET (□USING)
2010's	The most complete Array Language (+ ≠ ÷ ≠) ∘ ∘ ∘ @ ∘ ∘ ∘
2020's	The best APL for Cloud Computing (☁)



Major Milestones in 1st 30 years

- **1983:** APL2 Nested Arrays + SHARP APL Component Files, Error Trapping, etc
- **1990:** Namespaces, Windows GUI
- **1995:** Control structures (If/Then/Else, Repeat/Until, exception handling, and so on)
- **1996:** Functional programming: dfns provide lexical scope and lambda-style expressions
- **2006:** Object oriented programming, tighter integration with .NET & OO frameworks
- **2013:** Rank (¨) and Key (⊞) operators
- **2014:** Point-free or "tacit" syntax similar to that in the J programming language
- **2014:** Futures and isolates for parallel programming

More highlights of the last decade

- **2015:** macOS, RIDE, JSON, Secure Sockets
- **2016:** Cross Platform File Functions, load & edit Unicode Test Files using editor
- **2017:** CSV, HTTP Support in Conga, HTMLRenderer, Where (⊔), At (@), Stencil (⊞) and Nest/Partition (⊚)
- **2018:** aplssh & pynapl, APL as a DLL, Total Array Ordering, JSONServer
- **2019:** Link, Unregistered non-Commercial Version, Headless mode, Containers
- **2020:** ⌈C, ⌈DT, constant (¨) over (¨) and atop (¨), unique mask (≠), .NET Core Bridge, LOAD text files, text config files
- **2022:** Basic Licence, Shell Scripts, ⌈ATX

Performance increased consistently during most of this decade.



More highlights of the last decade

- **2015:** macOS, RIDE, JSON, Secure Sockets
- **2016:** Cross Platform File Functions, load & edit Unicode Test Files using editor
- **2017:** CSV, HTTP Support in Conga, HTMLRenderer, Where (l), At (@), Stencil (⊠) and Nest/Partition (⊟)
- **2018:** aplssh & pynapl, APL as a DLL, Total Array Ordering, JSONServer
- **2019:** Link, Unregistered non-Commercial Version, Headless mode, Containers
- **2020:** `⊠C`, `⊠DT`, constant (`⊠~`) over (`⊠ö`) and *atop* (`⊠ö`), unique mask (`⊠≠`), .NET Core Bridge, LOAD text files, text config files
- **2022:** Basic Licence, Shell Scripts, `⊠ATX`

Performance increased consistently during most of this decade.

Major Milestones in 1st Half of the Fifth Decade

- **2024 / Tool Projects**
 - Tatin & Cider become "mainstream"
 - Jarvis & WebSocket integration
 - One secret project, come to Glasgow 😊!
- **2025 / v20:**
 - Literal Array Notation
 - .NET Generics
 - HTMLRenderer (& Conga?) as Open-Source extensions
 - Android version?
- **2026 / v21:**
 - Co-dfns compiler integrated with Dyalog APL
 - All asynchronous operations return futures
 - Dual / Under, Differentiation
- **2027 / v22:**
 - Multiple Numeric Towers: 64-bit integers, Rationals, Unlimited-precision integers

(NB Mortens dreams, **NOT** promises!)



Major Milestones in 1st Half of the Fifth Decade

- **2024 / Tool Projects**
 - Tatin & Cider become "mainstream"
 - Jarvis & WebSocket integration
 - One secret project, come to Glasgow 😊!
- **2025 / v20:**
 - Literal Array Notation
 - .NET Generics
 - HTMLRenderer (& Conga?) as Open-Source extensions
 - Android version?
- **2026 / v21:**
 - Co-dfns compiler integrated with Dyalog APL
 - All asynchronous operations return futures
 - Dual / Under, Differentiation
- **2027 / v22:**
 - Multiple Numeric Towers: 64-bit integers, Rationals, Unlimited-precision integers

(NB Mortens dreams, **NOT** promises!)

And... without dates attached

- Universities start offering APL courses
 - Not as part of a "comparative" course
- APL recognised as a respectable tool for modeling machine learning and quantum computing
- First commercially significant APL application created by someone who learned APL in this millennium
- "Full Stack" Web development in APL



Ken Iverson's Blackboard arrives in Bramley (2010)



Dyalog ... The Next Generation



Aaron (ACE) 



Adam (AE) 



Rich (AE) 



Josh (A) 



Stine (D) 



Karta (C) 

Legend:

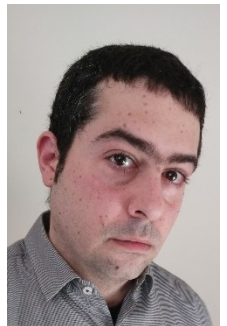
- A = APLer
- C = C developer
- D = Admin
- E = Doc/Evangelism



Silas (C) 



Peter (C) 



Jesus (E) 

 2022



Dyalog ... The Next Generation



Silas (C)



Peter (C)



Jesus (E)



Legend:

A = APLer

C = C developer

D = Admin

E = Doc/Evangelism

I = IT



Jada (D)



Stefan (AE)



Aarush (AC)



Abs (I)



Mike (DE)



+



Asher (ACE)



Kamila (ACE)



(Summer Interns)



Academic Collaboration

- ◆ Santiago Núñez-Corrales PhD, Markos Frenkel, Bruno de Abreu
National Center for Supercomputing Applications

- ◆ A Quantum Computing Library for APL

- ◆ Jesús Galán López (Ghent University)

- ◆ Metallurgy with APL

- ◆ Asher Harvey-Smith (University of Warwick)

- ◆ (and Summer Intern at Dyalog)

Tuesday 14:00 Teaching Algebra with APL
(Asher Harvey-Smith, U. Warwick)

Tuesday 13:00 quAPL
(Markos Frenkel, NCSA/U.Illinois)

Tuesday 13:30 APL and Metallurgy
(Jesús Galán López, Ghent U.)

Wednesday 11:00 Grain Growth
and Array Programming
(Jesús Galán López, Ghent U.)





LambdaConf speakers

[Apply to be a speaker!](#)



KEYNOTE

Adam Fraser

[Ziverge](#)

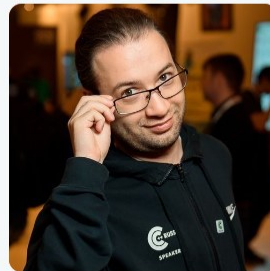


KEYNOTE

John A. De Goes



Aaron Hsu



Alexander Granin



Alexander Ioffe



Amanda Graham



Andreas Rumpf



André M van Meulebrouck



- ◆ Can you host a young(er) member of our team for 3-5 days?
- ◆ Or perhaps even run a small consulting project with 1 senior + 1 junior Dyalogger?



2022 Road Map



2023...



Why v19.0 is Late... (January 2024)

- ◆ Hired lots of new people
 - ◆ Adjusted ~~Created~~ processes to manage growth
 - ◆ Summer interns (Wonderful!)
 - ◆ Two .NET bridges
 - ◆ Two macOS variants to build & sign
 - ◆ ARM64 version for the Macs, Pi and AWS Graviton
 - ◆ Cumulative 3-4 month delay caused collision with Dyalog'23
-
- New Long Term Support version of .NET (8.0) on November 8th
 - Want to test with 8.0 before General Availability
 - Official Beta Testing to start mid-November

Excuses

Excuses

Excuses



Highlights Version 19.0

Platform Support / Distribution

- 64-bit ARM support
 - Mx Macs, Pi 4&5, AWS Graviton
- Enhanced .NET Bridge
 - Framework vs new .NET versions
- Bound executables on all platforms

Building Production Systems

- Token range reservation
- WS FULL handling
- NCOPY/NMOVE callbacks

Developer Productivity / IDE

- Source "as typed" by default
- Multi-line input on by default
- HTMLRenderer updates
- Link 4.0: Support for text data
- HttpCommand client, Jarvis web service

Installing & Managing APL

- Multiple session files
- Health Monitor



Service Orientation

A rapidly increasing proportion of new APL code is delivered as services

- ◆ **Jarvis** wraps APL code as HTTP/JSON or RESTful services on any platform
 - ◆ <https://github.com/dyalog/jarvis>
- ◆ Off-the-shelf docker containers containing Dyalog APL (optionally with Jarvis)
- ◆ **HttpCommand** is our HTTP client

RESTful API

GET PUT POST DELETE

HTTP(s) / JSON



Service Orientation

Monday 14:45 APL Worker Bees
(Stig Nielsen, SimCorp)

Monday 15:15 Transforming and Streamlining a Complex Process
(Mark Wolfson, BIG)

Tuesday 09:00 Dyalog Tools Update
(Brian Becker)

Tuesday 09:30 Converting from COM to a Jarvis Web Service
(Finn Flug, DPC Consulting)

Tuesday 11:00 Dyalog, AWS, Jarvis, Docker...What's Not to Like?
(Claus Madsen, FinE Analytics)

Tuesday 16:45 Dyalog + Kafka = True?
(Stefan Kruger)

RESTful API
GET PUT POST DELETE

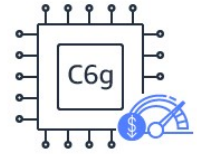
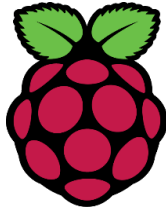
HTTP(s) / JSON



Arm64

64-bit ARM chips appearing in many places

- M1, M2 & M3 Macs
- Raspberry Pi – 64 Bit
- Amazon Web Services "Graviton"



Best price performance for compute-intensive workloads



Tuesday 10:00 Dyalog on ARM64
(Ron Murray)



Ta' den hyggeligste
vej til Sverige

Fyld
bilen fra
199,-



MICROSOFT

SOFTWARE

RASPBERRY PI

WINDOWS 10

Your Raspberry Pi 3 can now run Windows 10 ARM

The Raspberry Pi just got sweeter

By [Dean Pennington](#) February 13, 2019 at 12:08 PM | [13 comments](#)



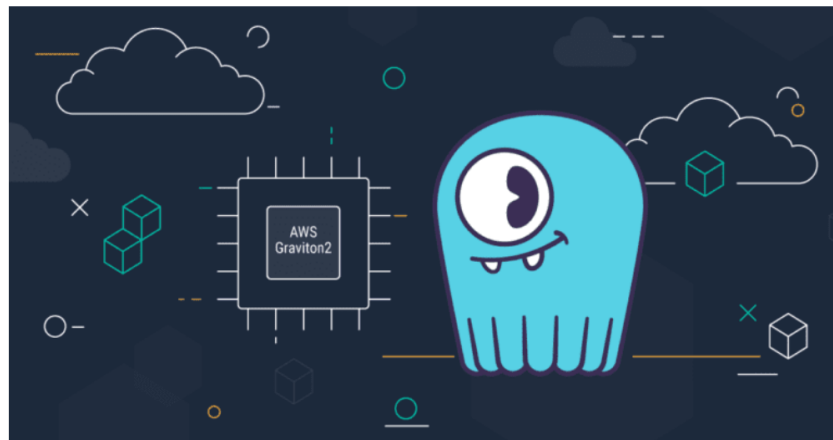
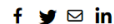
[< See all blog posts](#)

AWS Graviton2: Arm Brings Better Price-Performance than Intel



By Michal Chojnowski

September 16, 2021

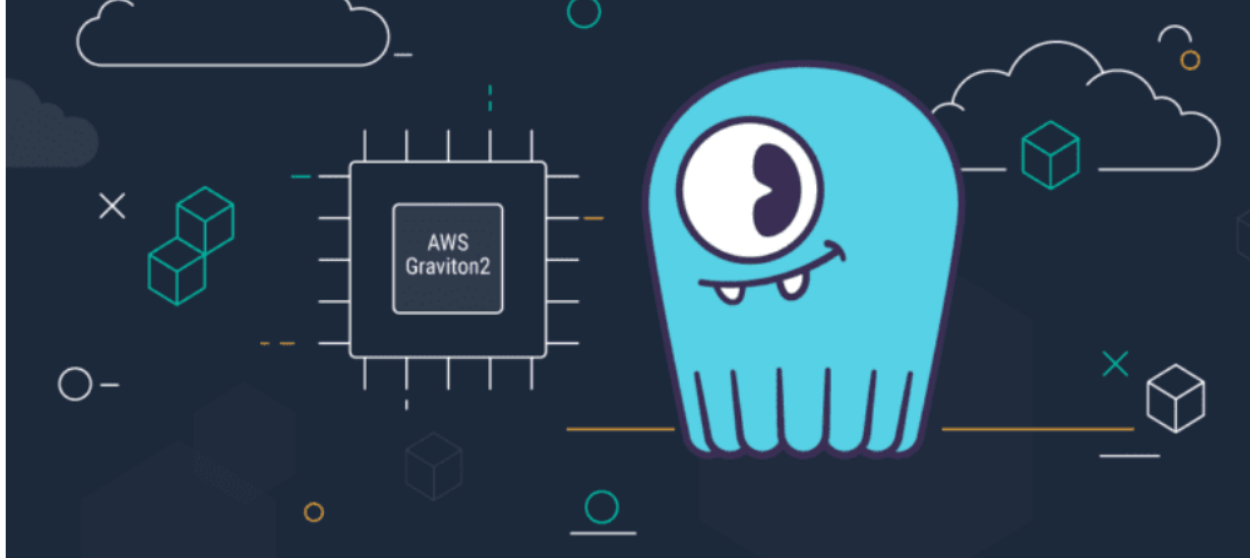


Since [the last time](#) we took a look at ScyllaDB's performance on Arm, its expansion into the desktop and server space has continued: Apple introduced its M1 CPUs, Oracle Cloud added Ampere Ultra-based instances to its offerings, and AWS expanded its selection of Graviton2-based machines. So now's a perfect time to test Arm again – this time with SSDs.

Summary

We compared ScyllaDB's performance on m5d (x86) and m6gd (Arm) instances of AWS. We found that **Arm instances provide 15%-25% better price-performance**, both for CPU-bound and disk-bound workloads, with similar latencies.





Since [the last time](#) we took a look at ScyllaDB's performance on Arm, its expansion into the desktop and server space has continued: Apple introduced its M1 CPUs, Oracle Cloud added Ampere Ultra-based instances to its offerings, and AWS expanded its selection of Graviton2-based machines. So now's a perfect time to test Arm again – this time with SSDs.

Summary

We compared ScyllaDB's performance on m5d (x86) and m6gd (Arm) instances of AWS. We found that **Arm instances provide 15%-25% better price-performance**, both for CPU-bound and disk-bound workloads, with similar latencies.



Dyalog APL for ARM-based Macs

- Version 19.0 will be available in two versions for macOS:
 - ARM64: M1, M2 and soon M3 based Macs
 - Intel x64: For older Macs
- 64 bit, Unicode only
- NB: Version 19.0 will be the last version to support Intel-based Macs.**

Apple Hardware

1979: 68000

1994: POWER

2005: Intel x64

2021: ARM64



Sun is Setting on Classic & 32 Bit

Classic

- Down to no more than half a dozen significant clients
- More than half of these actively planning - or working on - moving to Unicode
- May soon decide to offer Classic on IBM AIX only

32 Bit

- Rapidly dwindling user base
- Difficult to test due to lack of support from operating systems and other frameworks
- Discounted price will be removed next year; 32 will cost the same as 64.

Neither are for sale to new users or supported on new platforms

Tuesday 16:45 Return of Uncle Andy's Fireside Chat
(Uncle Andy)



[Microsoft].NET History

- .NET has been around for 20+ years. The old "Framework" is being replaced by an open source, cross-platform .NET, initially known as ".NET Core".

Name	Platforms	Version Numbers
Microsoft.NET Framework	Windows	1 2 3 4.0 4.8.1
".NET Core"	Windows Linux macOS	1 2 3
→ ".NET"	Windows Linux macOS	→ 5.0 6.0 7.0 8.0

- Dyalog v18.0 added a bridge to .NET Core 3, to complement the 20 year old bridge to the .NET Framework.
- v18.2 officially supported "Core" 3.1 but works with 5.0 and later
- v19.0 targets 8.0 (Long Term Support version due on Nov 8th 2023)



v19.0 .NET Bridge

- Adds support for .NET 6, 7, 8 ...
 - Tested with 6.0 & 8.0 - and 4.8 (aka ".NET Framework")
 - Will be shipped configured for 8.0
- Can export APL code as .NET assemblies
 - Will allow embedding APL code in .NET frameworks like ASP.NET Core, etc



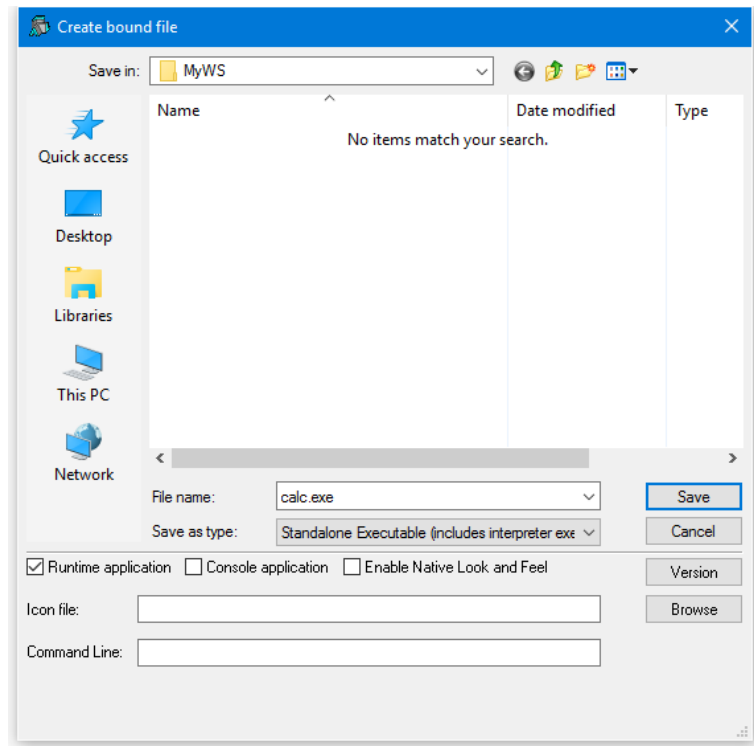
.NET 8.0 will be the Long Term Support version on Nov 8th



Bound Executables

A bound executable is a file which combines an interpreter and a workspace into a single .exe file

- "Always" been available under Windows
- In v19.0 also available for Linux
 - Maybe MacOS soon
 - (but you will need to sign the result)
- In the longer term, I expect we will look at encrypting and signing application code



Token Range Reservation

- ◆ System functions `□TGET` and `□TPUT` allow threads to synchronise execution, and pass values to each other using numbered "tokens".
- ◆ New system function `□TALLOC` allocates token ranges, allowing independent components to avoid using the same tokens.
- ◆ `□TALLOC` returns a single integer `n`, granting the right to use **floating-point** token ids in the range `< n , n+1 >`
- ◆ **NB not including the (integer) end points**
 - ◆ This allows old-style integer tokens to be used by existing applications without conflicting with new modules



WS FULL Handling

- ◆ IF a WS FULL leaves VERY little free space, THEN the interpreter and IDE can malfunction
 - ◆ For example, a runaway recursion can leave only a few bytes of free workspace
 - ◆ **Error trapping may not be possible (system might just stop)**
- ◆ Version 19.0 allocates 1% of MAXWS as a buffer which is released on WS FULL
 - ◆ Allows WS FULL traps to be safely processed
 - ◆ (the reservation size is configurable)
 - ◆ After successful trap handling, space is re-acquired



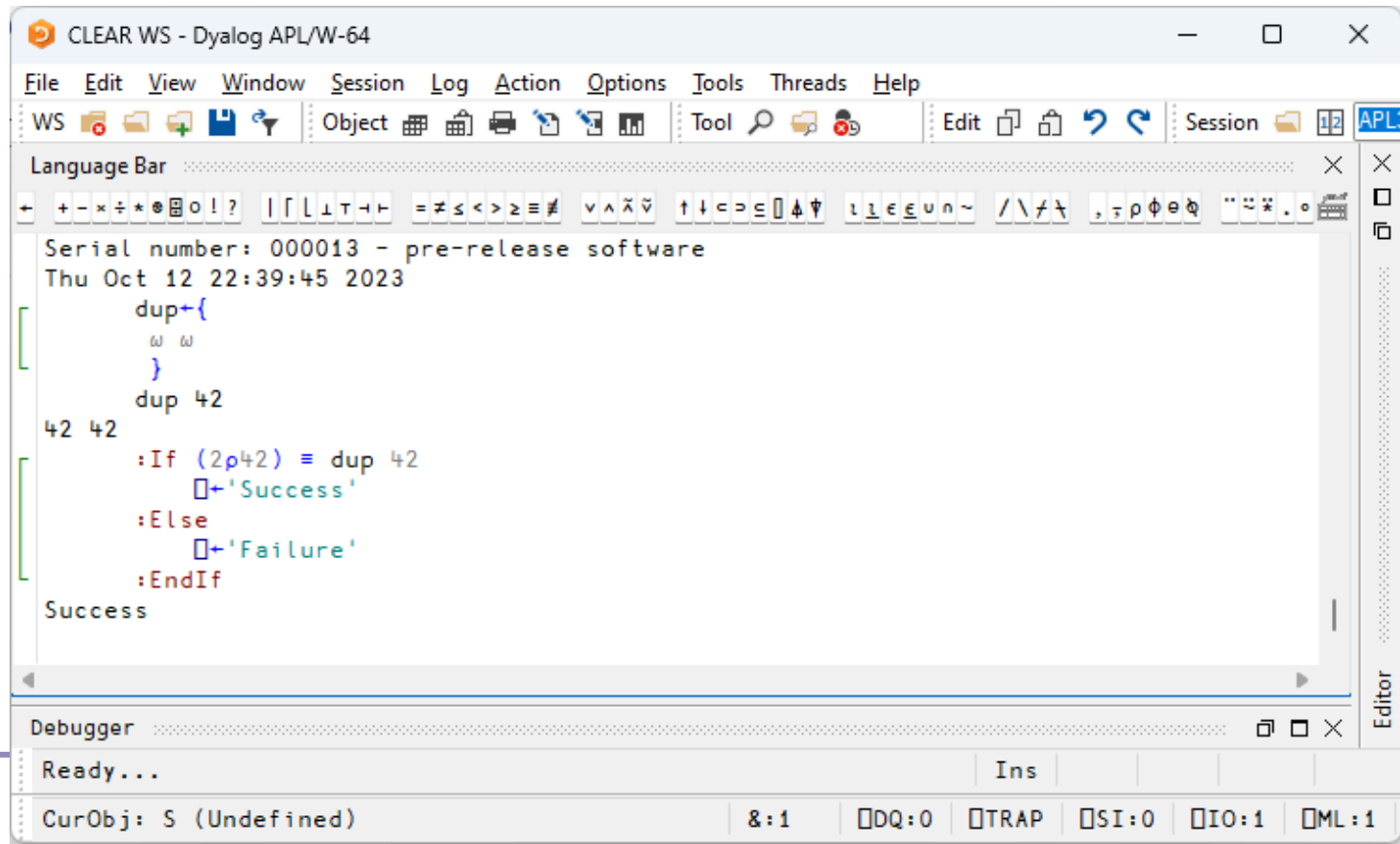
Source "as typed" by default

- Historically, Dyalog APL has re-generated source code from tokenised code when an editor is opened.
 - This does **not** preserve white space and constants exactly as typed by the user.
- For several releases, Dyalog APL has preserved source "as typed" when a function or operator was created using `⎕FIX`

```
2 ⎕FIX 'file://myapp/foo.aplf'
```
- From version 19.0, the default is to preserve source as typed within the workspace for **all** fns and ops
 - NB: Old behaviour can be selected if desired.



Multi-line input On by Default



The screenshot shows the CLEAR WS - Dyalog APL/W-64 application window. The menu bar includes File, Edit, View, Window, Session, Log, Action, Options, Tools, Threads, and Help. The toolbar contains icons for file operations, editing, and session management. The Language Bar shows various APL symbols. The main editor area displays the following APL code:

```
Serial number: 000013 - pre-release software
Thu Oct 12 22:39:45 2023
dup+{
  ω ω
}
dup 42
42 42
:If (2ρ42) ≡ dup 42
  □+'Success'
:Else
  □+'Failure'
:EndIf
Success
```

The Debugger window at the bottom shows the status 'Ready...' and the current object 'CurObj: S (Undefined)'. The status bar at the bottom right displays various system flags: &:1, □DQ:0, □TRAP, □SI:0, □IO:1, □ML:1.



HTMLRenderer updates

Use of the HTMLRenderer continues to grow.

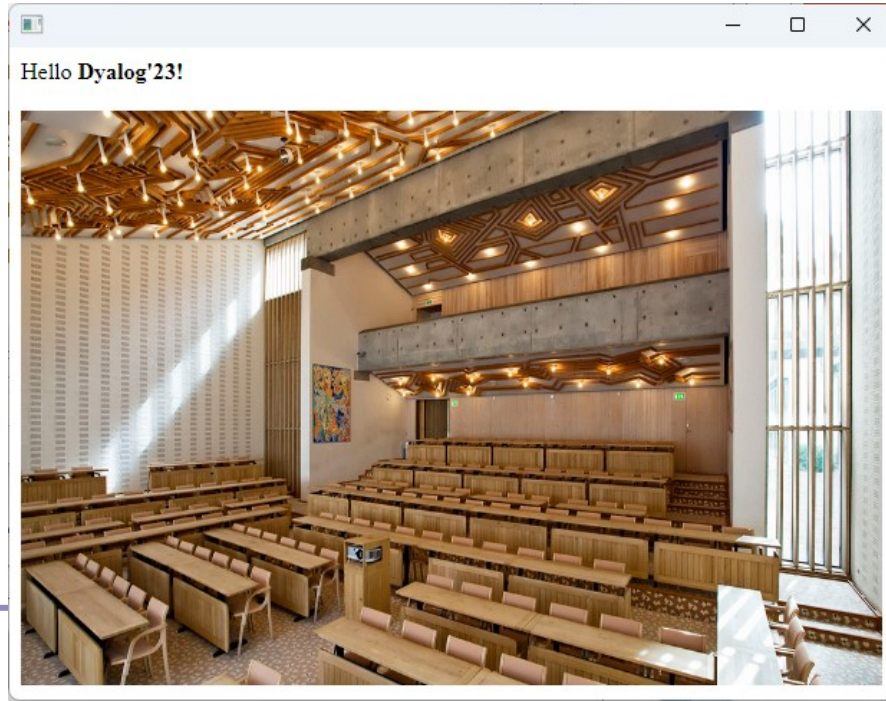
New features include:

- ExecuteJavaScript (asynchronous)
- AllowContextMenu
- Get/SetZoomLevel
- IsLoading + LoadEnd event



HTMLRenderer – what's that?

```
pic←'https://www.konventum.dk/media/jjcjop23/carousel_damgaard_1135x604.png'  
'MyForm' WC 'HTMLRenderer' ('Hello <b>Dyalog'23!</b><br/>  
    <img src="" ,pic,' height=400 width=600>')
```



Chromium
Embedded
Framework
(CEF)



Source Code Management

Productivity & IDE

Bundled with v19.0:

- Link v4.0, with support for simple text vectors, vectors of text vectors, and character matrices in simple text files, configuration files, many small features & fixes
- Prototypes of the Cider project manager and the Tatin package manager client will be bundled with v19.0



Monday 16:15 Evolution of APLTree and APL-cation
(Kai Jaeger)


Monday 16:45 Using Packages
(Morten Kromberg)



Tatin Registry

List of all packages (aggregated)

Package name	Description	Major versions	Link to project
aplteam-APLGit	Git interface from Dyalog APL via Git Bash	1	github.com
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com
aplteam-APLTreeUtils2	General utilities required by most members of the APLTree library	1	github.com
aplteam-CodeCoverage	Monitors which parts of an application got actually executed	1	github.com
aplteam-Compare	Allows comparing and merging objects in the workspace with a file or a file with another file	2	github.com
aplteam-CompareSimple	Allows comparing objects in the workspace with a file or a file with another file	2	github.com
aplteam-DateAndTime	Utilities related to Date and Time, including doing math	1	github.com
aplteam-DotNetZip	Zippping and unzipping with .NET Core on all major platforms	2	github.com
aplteam-EventCodes	Constants with meaningful names for Dyalog error codes	1	github.com
aplteam-Execute	Start a process from within APL	1	github.com
aplteam-FilesAndDirs	Utilities for doing gymnastics with files and directories	1	aithub.com
(... many more of Kai's packages skipped ...)			
aplteam-WindowsEventLog	tools to read from and write to the windows Event Log	1	github.com
aplteam-ZipArchive	Zippping and unzipping with .NET on Windows and zip/unzip on oth	2	github.com
davin-DateTime	Easy calculations with dates	1	github.com
davin-FilePlus	Extend component files to use named components	1	github.com
davin-SQLFns	Easily create text SQL commands for use with any SQL program in	1	github.com
davin-Tester	Simplified function-level testing of programs	1	github.com
dyalog-HttpCommand	Utility to execute HTTP requests	1	dyalog.github.io
dyalog-Jarvis	JSON and REST Web Service Framework	1	dyalog.github.io


UCS 129346

Packages Coming in 2024?

(Potentially, at least)

Tuesday 16:45 Dyalog + Kafka = True?
(Stefan Kruger)

Wednesday 09:30 A YAML Parser in APL
(Brandon Wilson, Effective Altruism)

Wednesday 10:00 Statistical Libraries for Dyalog
(Josh David)

Thursday 11:00 Vega Charts with Dyalog
(Rich Park)



Health Monitor

Experimental TCP-based monitor:

- Regular updates on (for example) :
 - CPU consumption
 - Memory statistics
 - Are any threads suspended?
 -)SI stack and Error information
- Notification on
 - untrapped error
 - ws compaction
- Exact execution location if "breadcrumbs" enabled
- Information about whether a RIDE connection is possible



Health Monitor Example

```
["PollFacts",{"Facts":["AccountInformation","Workspace","ThreadCount"],"Interval":5000,"UID":"1 1"}]
```

```
["Facts",  
{"Facts": [ {  
  "ID": 2, "Name": "AccountInformation",  
  "Value": {  
    "ComputeTime": 438,  
    "ConnectTime": 46973,  
    "KeyingTime": 0,  
    "UserIdentification": 0  
  }}, {  
  "ID": 3, "Name": "Workspace",  
  "Value": {  
    "Allocation": 33882064,  
    "AllocationHWM": 33882064,  
    "Available": 2144207480,  
    "Compactions": 2,  
    "FreePockets": 186682,  
    "GarbageCollections": 0,  
    "GarbagePockets": 0,  
    "Sediment": 2120,  
    "Used": 3276168,  
    "UsedPockets": 23209,  
    "WSID": "CLEAR WS"  
  }}, {  
  "ID": 6, "Name": "ThreadCount",  
  "Value": {  
    "Suspended": 1,  
    "Total": 2  
  }  
}],  
"Interval": 5000,  
"UID": "1 1"  
}]
```



Highlights of v19.0

Not to be forgotten:

- We closed a LOT of "issues"
 - Tricky dfn scoping issues were fixed \😊/
 - Significant contribution from new team members
- Unfortunately the list of open issues is growing
 - ...mostly because we're doing more, better testing
 - New users with new usage patterns
 - New employees finding bugs & logging "WIBNIs*"

*
Wouldn't
It
Be
Nice
If ...



Sketch of Version 20.0 (Q2/2024)

Transferred from v19.0

- Resume Optimisation Work
- .NET Bridge "enhancements"
 - Support "Generic" methods & classes
- More HTMLRenderer improvements
 - Work on open-sourcing it
- Health Monitor
- Script Engine Support
- □NATTRIBUTES

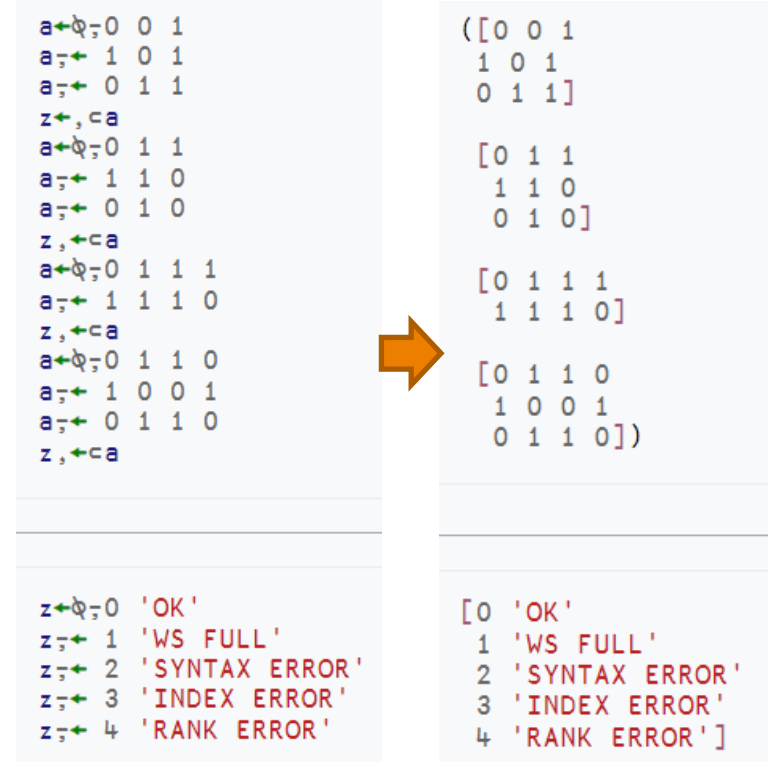
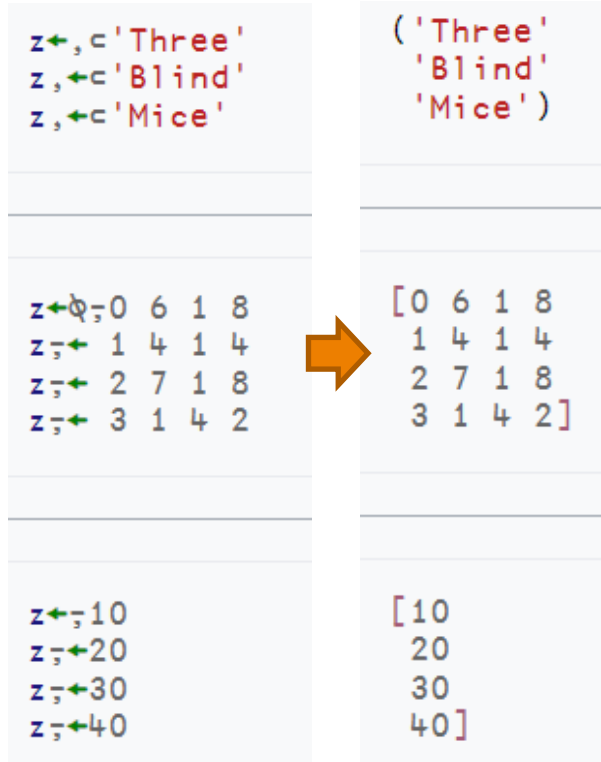
Next Set of Projects

- Array Notation
- Token-by-token Debugging
- Probably some New Operators
- Query Platform Features
- New "Shell" System Command
- Virtual "Execution Environments"
- Design / Prototyping of Async



Array Notation

https://aplwiki.com/wiki/Array_notation



Array Notation

```
z ← □NS⊖  
z.y ← □NS⊖  
z.y.x ← ⊖; 'hello'  
z.y.x; ← 'world'
```

```
(y:(x:['hello'  
      'world'])))
```

- ◆ Proposal published – awaiting community feedback
 - ◆ https://aplwiki.com/wiki/Array_notation
- ◆ APL model exists in Link and
□SE.Dyalog.Array.Serialise|Deserialise
- ◆ Hope to start C implementation in v20.0

Thursday 10:00 An Implementation of APL Array Notation
(Kamila Szewczyk, Saarland University - Dyalog Summer Intern)



Source Code Management

- Critical for adoption of APL by new generation of users
- Independent from v20.0 projects, but in same timeframe:
 - Project Management - Cider
 - Package Management - Tatin
 - Link 5.0 with a "Crawler" as backup / alternative to File System Watcher
- Voice opinions to Gilgamesh Athoraya, Kai Jaeger, Rich Park, Stefan Kruger, or myself.



Virtual Environments

- Closely related to Projects & Packages
- A virtual environment defines
 - A specific version of Dyalog APL
 - With a set of installed packages
 - And a set of environment variables
- Allows development or maintenance of applications in controlled / static settings



Optimisation Work

- Initial focus on performance of ϵ and ι
- Instrumented interpreter can collect data about calls made by running applications
- In version 20.0, we hope to implement the first improved algorithms



Monday 14:00 Performance of Set Functions
(Karta Kooner)



HTMLRenderer Enhancements

- Medium term: Separate HTMLRenderer from APL and make it a separate, open source component
 - Not sure this will make it to v20.0
- Until then, we may release regular updates to the HTMLRenderer to pick up new versions of CEF
 - When there are critical security patches to Chromium
- These releases may contain feature tweaks
 - File Upload, Multiple Modal instances



Configuration Files

- Continue the move towards portable configuration files
- Currently require a startup script on non-Windows platforms
 - Implement Good defaults for all settings under Linux, macOS, ...
 - All settings configurable via text config files
 - Eliminate the need for the script
- Remove need for the Windows Registry
 - Windows interpreter already has good defaults if no config found
 - Move all "interpreter settings" to text (JSON5) config files
 - Leave IDE settings in the Registry (as RIDE settings are in a JSON repository)

This might not make 19.0-20.0, but remains an important medium term goal.

```
{  
  Extend: "my_default_configuration.dcfg",  
  
  Settings: {  
    // maximum workspace  
    MAXWS: "2GB",  
    WSPATH: ["/dir1", "/dir2", ""],  
    UserOption: 123,  
    ROOTDIR: "/my/root/directory",  
    // references to other configuration parameters  
    FNAME: "[rootdir]/filename",  
  }  
}
```



.NET Bridge Enhancements

- The v19.0 bridge to "New .NET" is roughly on par with the Framework bridge
- Potential new features in v20.0
 - Generics
 - Delegates
 - (Tools to load/manage .NET dependencies)
- NB: New features will probably **NOT** be backported to the Framework bridge



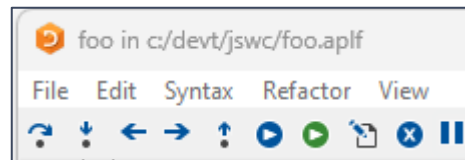
Tuesday 15:15 Dyalog Version 20.0 – Part 2
(John Daintree)



Token-by-token Debugging

- Not saying any more, John is up next

Monday 10:15 Dyalog Version 20.0 – Part 1
(John Daintree)



New Primitives / System Functions

- A small set of APL primitives is still missing
- See Adam's presentation at Dyalog'22
 - D15: Filling the Core Language Gaps
- And later today:

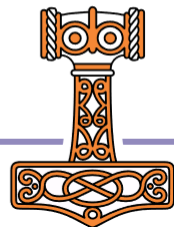
Monday 11:15 Setting and Getting Variable Values
(Adám Brudzewsky)

Tuesday 16:15 Giving Key a Vocabulary
(Adám Brudzewsky)



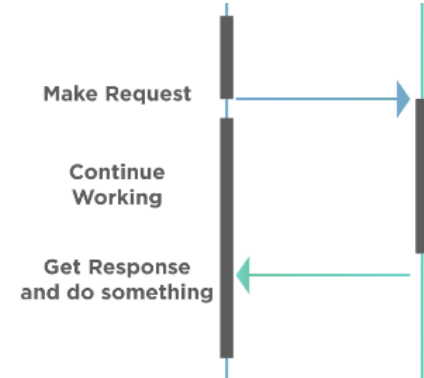
Core Language

Data Transformation	Select	$Y[X; ;]$	$X \supseteq Y$
Function Application	Depth	$X \text{ f } \dots \text{ c c } Y$	$X \text{ f } \text{ ö } k \ Y$
Function Composition	Behind	$(\text{ f } \ X) \text{ g } \ Y$	$X \text{ f } \underline{\text{ o }} \text{ g } \ Y$



Design / Prototyping of "Async"

- Work on the .NET bridge makes it clear to us that modern APIs are often asynchronous
- A future .NET Bridge needs to support this elegantly – see John Daintree's "2022 Conference Edition Part 3" talk from Dyalog'22
 - Async ... Await
 - And/or Futures

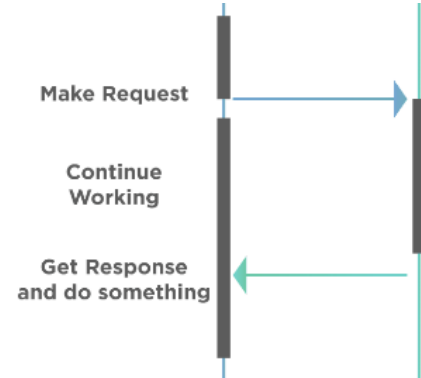


Grand Unified Theory of Async?

A dyadic parallel operator could invoke a function in a variety of asynchronous ways, all of which would immediately return a future:

Invoke foo in...

- `ns||foo` ↔ an "in process" fork of the ns
- `isolates||foo` ↔ a "separate process" (isolate)
- `⊖||foo` ↔ an empty isolate (current APL model)
- `0||foo` ↔ a green thread in current ws (like `current foo&`)
- `-1||foo` ↔ current ws, but lazily (when value is required)



Health Monitor

Version 19.0 contains a prototype. Ideas for v20.0 include:

- Complete implementation of "breadcrumbs" so it is possible to understand where an interpreter is hanging
- Sending signals to interrupt or terminate tasks
- Discoverability: allow APL process to broadcast services that it provides
- Switch PROFILE on and off; collect data



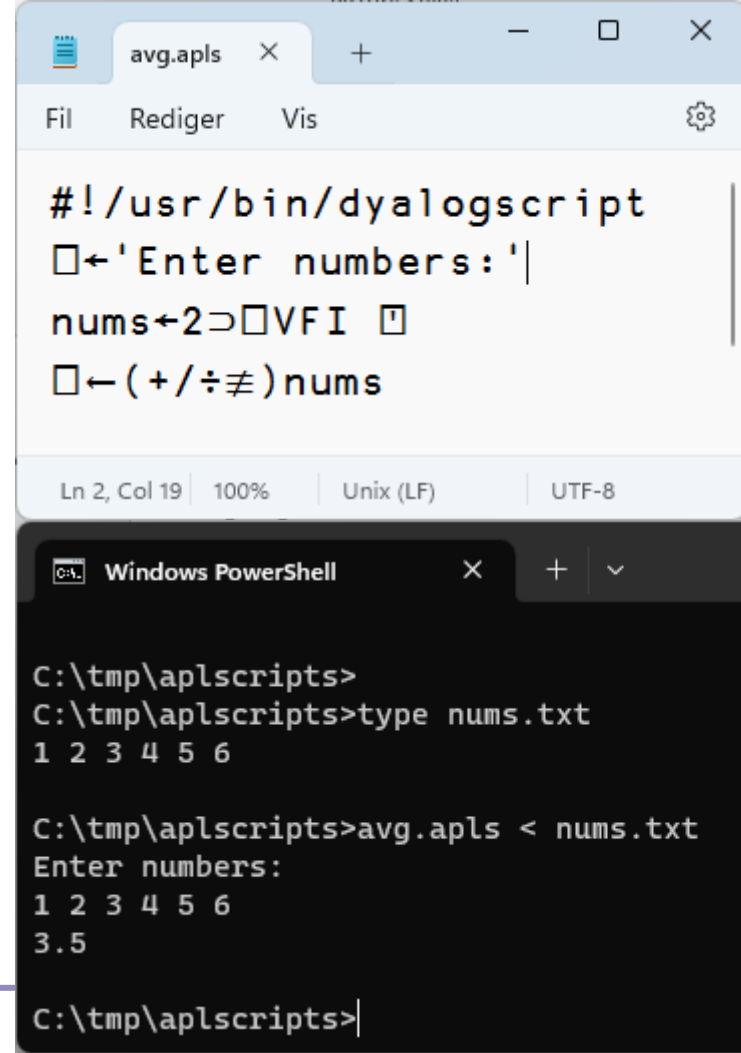
I/O Project

- I/O routines significantly refactored in v19.0
 - Goal is to properly manage redirection in the future
 - Changes should be invisible to users
- In v20.0, we aim to add
 - The ability to change redirection under program control
 - Ability to attach RIDE to a "script engine" and manage where programme and user input come from during debugging



Script-Engine Support

- #! (hash bang) scripting
- We think the script engine will be critical for attracting new users
- Still a bit of a prototype
 - Needs to be debug-able via RIDE
 - (awaiting next phase of I/O project)



The image shows a code editor window titled 'avg.apls' with a menu bar containing 'Fil', 'Rediger', 'Vis', and a settings icon. The code content is as follows:

```
#!/usr/bin/dyalogsript
☐←'Enter numbers: '|
nums←2☐☐VFI ☐
☐←(+/÷≠)nums
```

Below the code editor is a Windows PowerShell terminal window. It shows the following commands and output:

```
C:\tmp\aplscripts>
C:\tmp\aplscripts>type nums.txt
1 2 3 4 5 6

C:\tmp\aplscripts>avg.apls < nums.txt
Enter numbers:
1 2 3 4 5 6
3.5

C:\tmp\aplscripts>|
```

Other Small but Important Things

- `□SHELL` to replace existing `□SH`

- Interruptible
- Manage stdin, stdout & stderr

- `□PROFILE` enhancements

- Ideas: capture thread id, non-aggregating mode, memory allocations, binary export format

Monday 11:45 Revisiting `□SH` and `□CMD`
(Peter Mikkelsen)



Sketch of Version 20.0 (Q2/2024)

Transferred from v19.0

- Resume Optimisation Work
- .NET Bridge "enhancements"
 - Support "Generic" methods & classes
- More HTMLRenderer improvements
 - Work on open-sourcing it
- Health Monitor
- Script Engine Support
- □NATTRIBUTES

Next Set of Projects

- Array Notation
- Token-by-token Debugging
- Probably some New Operators
- Query Platform Features
- New "Shell" System Command
- Virtual "Execution Environments"
- Design / Prototyping of Async



Major Milestones in 1st Half of the Fifth Decade

- **2024 / Tool Projects**
 - Tatin & Cider become "mainstream"
 - Jarvis & WebSocket integration
 - One secret project, come to Glasgow 😊!
- **2025 / v20:**
 - Literal Array Notation
 - .NET Generics
 - HTMLRenderer (& Conga?) as Open-Source extensions
 - Android version?
- **2026 / v21:**
 - Co-dfns compiler integrated with Dyalog APL
 - All asynchronous operations return futures
 - Dual / Under, Differentiation
- **2027 / v22:**
 - Multiple Numeric Towers: 64-bit integers, Rationals, Unlimited-precision integers

(NB Mortens dreams, **NOT** promises!)

And... without dates attached

- Universities start offering APL courses
 - Not as part of a "comparative" course
- APL recognised as a respectable tool for modeling machine learning and quantum computing
- First commercially significant APL application created by someone who learned APL in this millennium
- "Full Stack" Web development in APL

And... not forgotten

- VS Code Integration (Debug Adapter Protocol)
- Inverted Tables (aka "Magic Arrays")
- WASM?



Hope that was all clear!

