

# DYALOG



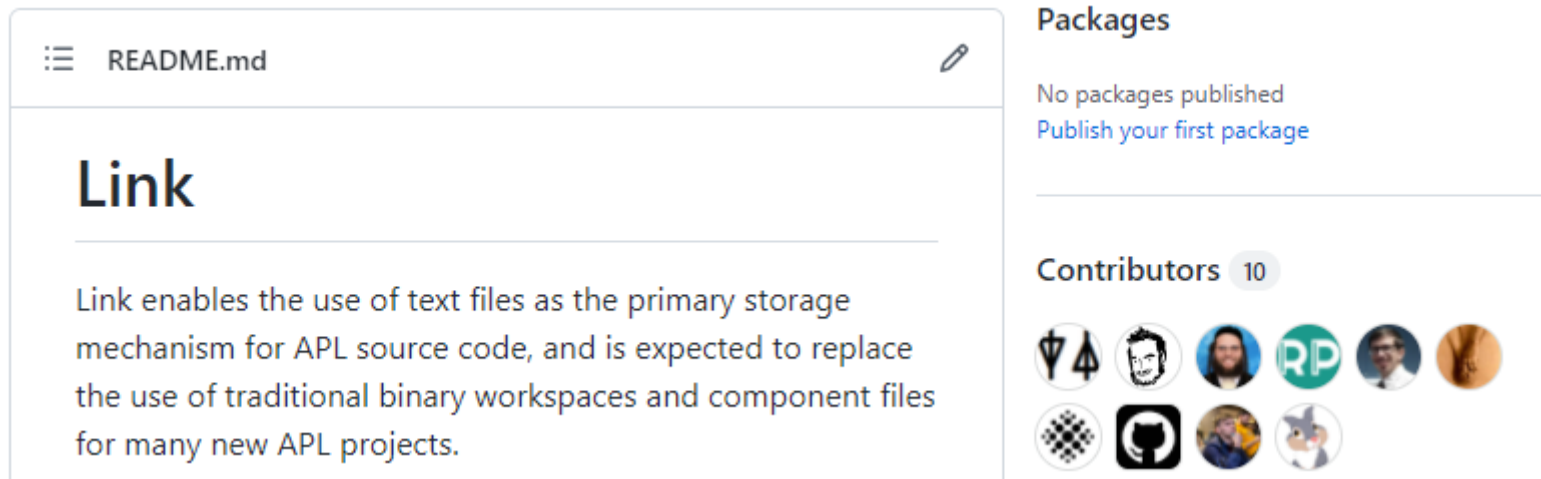
## Introduction to Link

*Morten Kromberg*



# What is Link?

According to <https://github.com/dyalog/link>



The screenshot shows the GitHub interface for the 'link' repository. The top bar indicates the file 'README.md' is open. The main content area features the title 'Link' followed by a description: 'Link enables the use of text files as the primary storage mechanism for APL source code, and is expected to replace the use of traditional binary workspaces and component files for many new APL projects.' To the right, the 'Packages' section states 'No packages published' with a link to 'Publish your first package'. Below this, the 'Contributors' section shows a count of 10 contributors, with a grid of 10 circular profile pictures displayed.

Link

Link enables the use of text files as the primary storage mechanism for APL source code, and is expected to replace the use of traditional binary workspaces and component files for many new APL projects.

Packages

No packages published  
[Publish your first package](#)

Contributors 10

# All you need to Know about Link...

To declare that your source is in `C:\linkdemo`

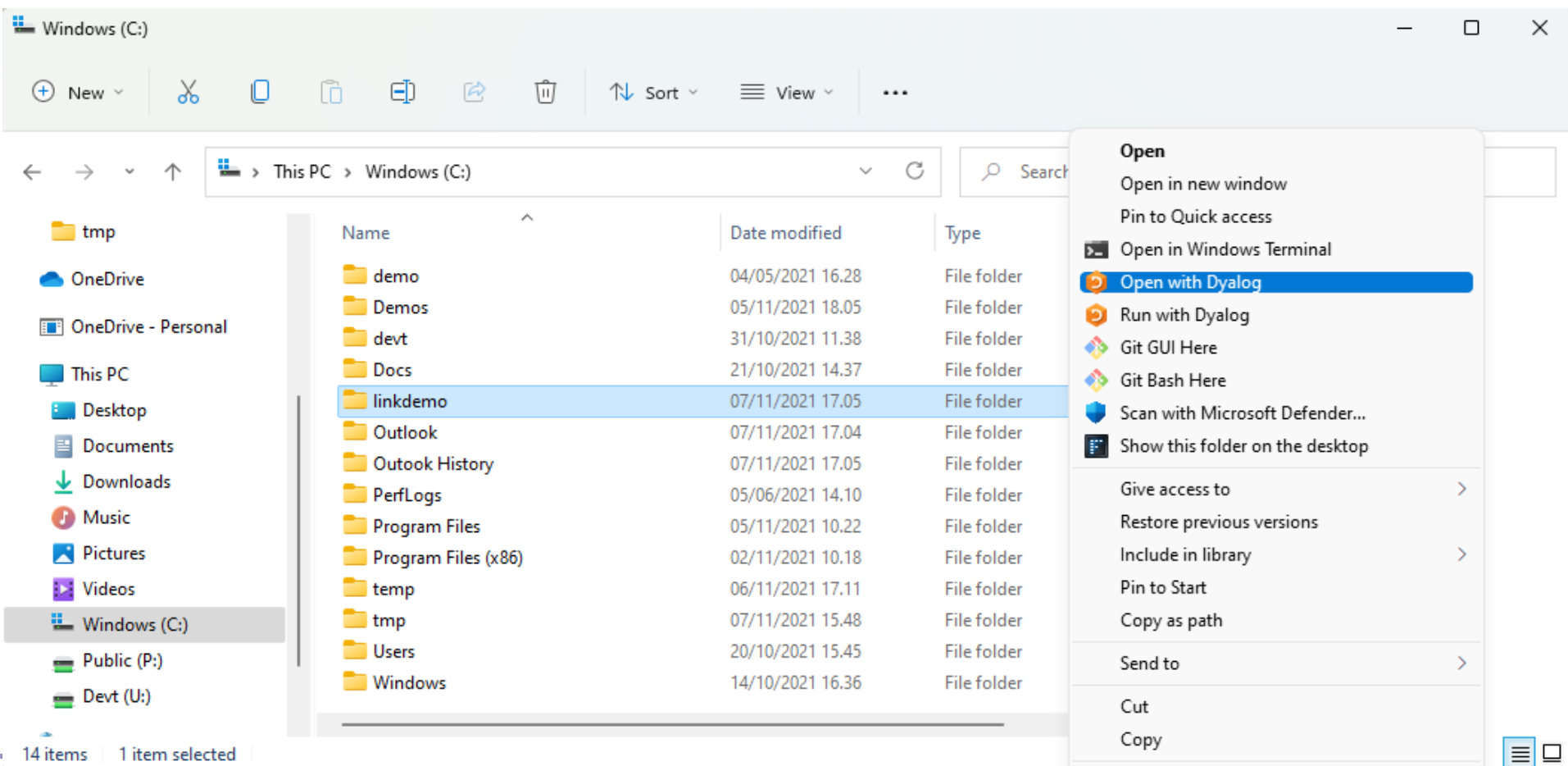
```
]link.create    #    C:\linkdemo
```

To import code that you do not intend to modify, from `C:\linkdemo\myapp` into the namespace "myapp":

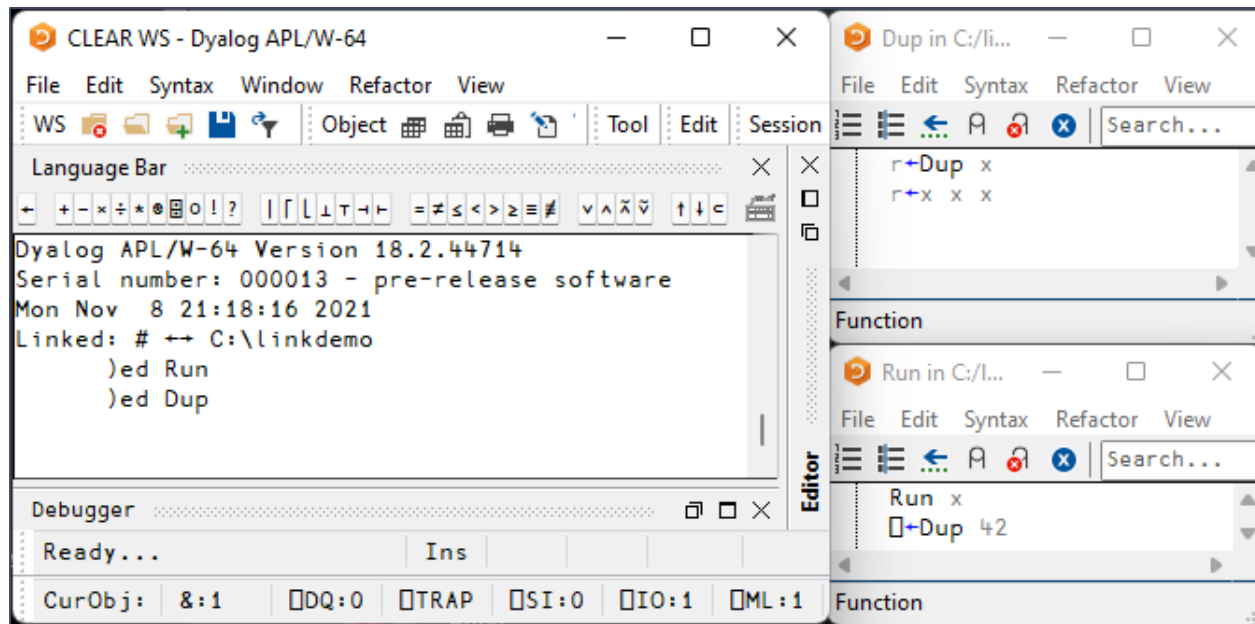
```
]link.import    myapp    C:\linkdemo\myapp
```

...and then proceed to use APL normally

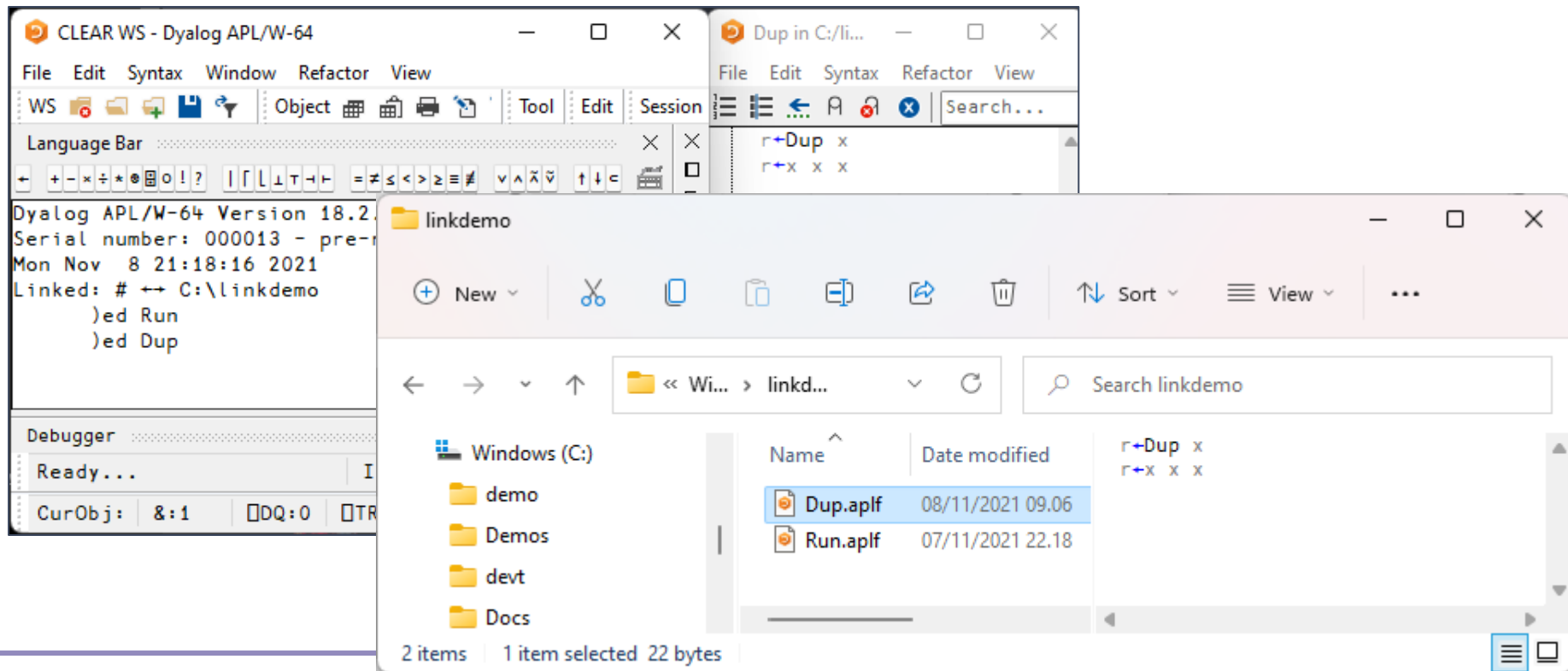
...but without having to remember to type `)SAVE` at the end of your day.



# Workspace + Source are Synchronised



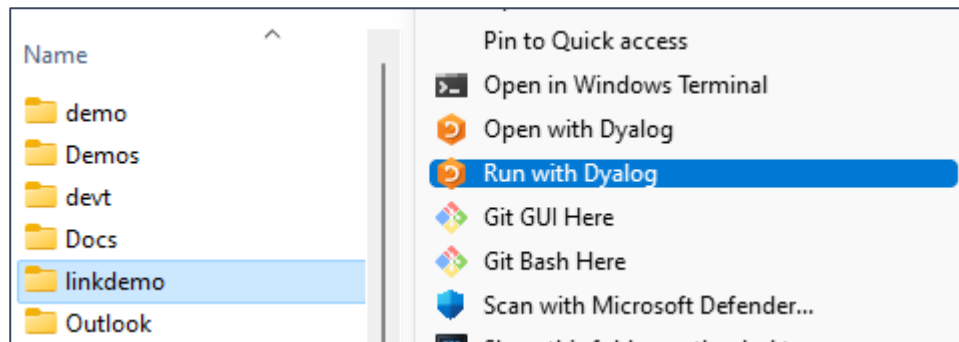
# Workspace + Source are Synchronised



# Workspace + Source are Synchronised

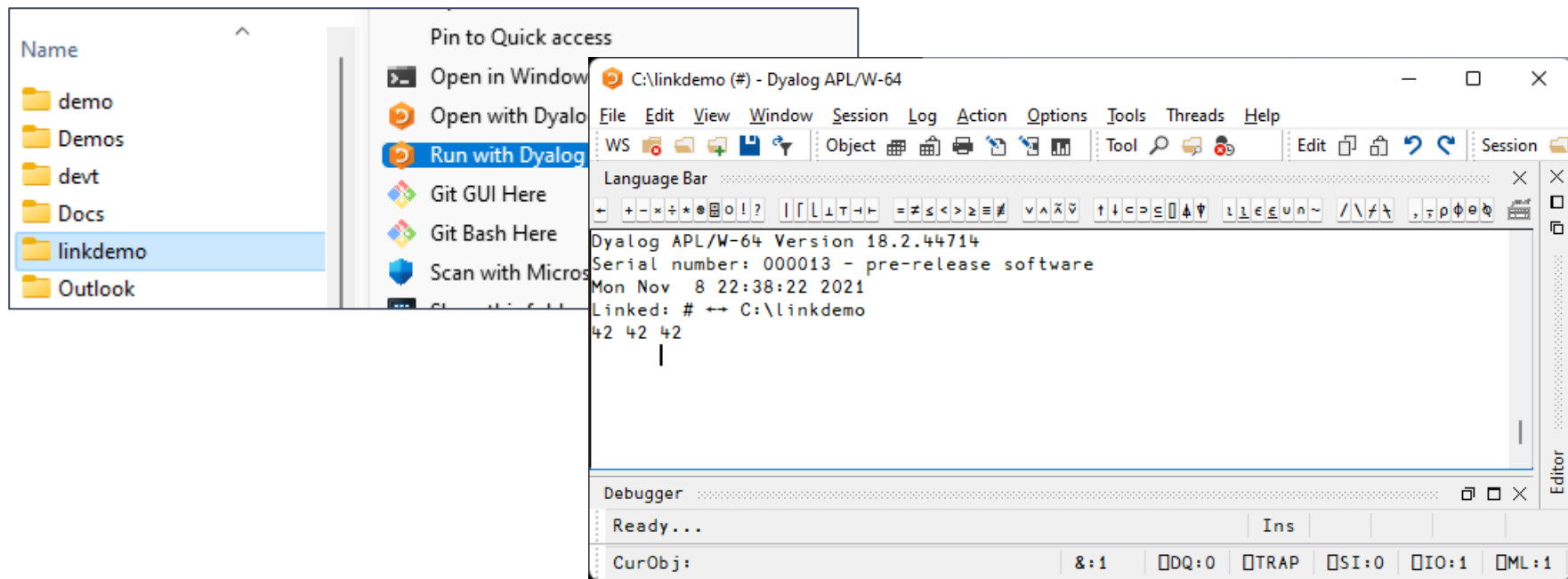
- Changes made "in APL" using the APL Editor are **immediately** written to file
- Changes made to the files using external tools (editors, source code management systems, etc) are **immediately** brought in to the workspace

# "Run with Dyalog" calls function Run



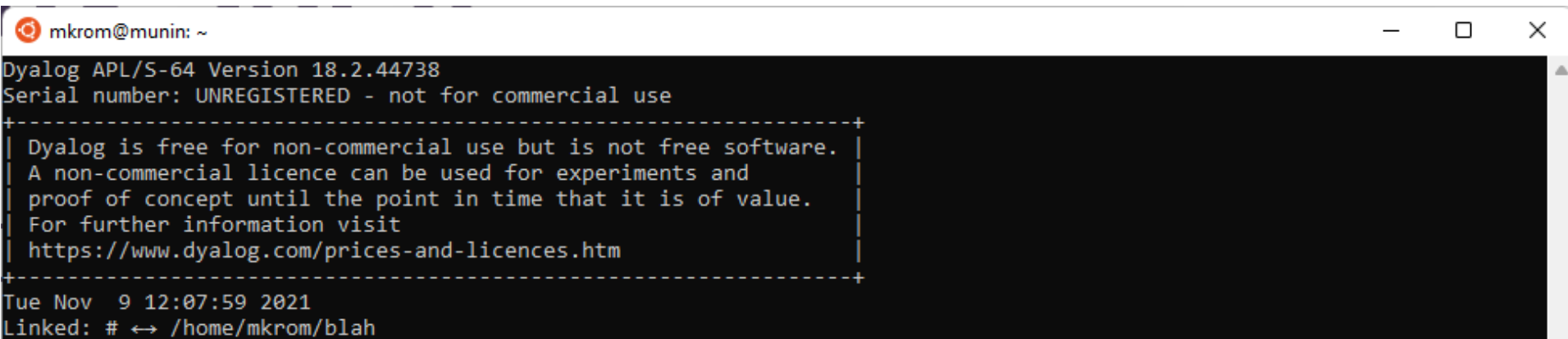


# "Run with Dyalog" calls function Run



# Launching from Shell

```
$ dyalog LOAD="blah"
```

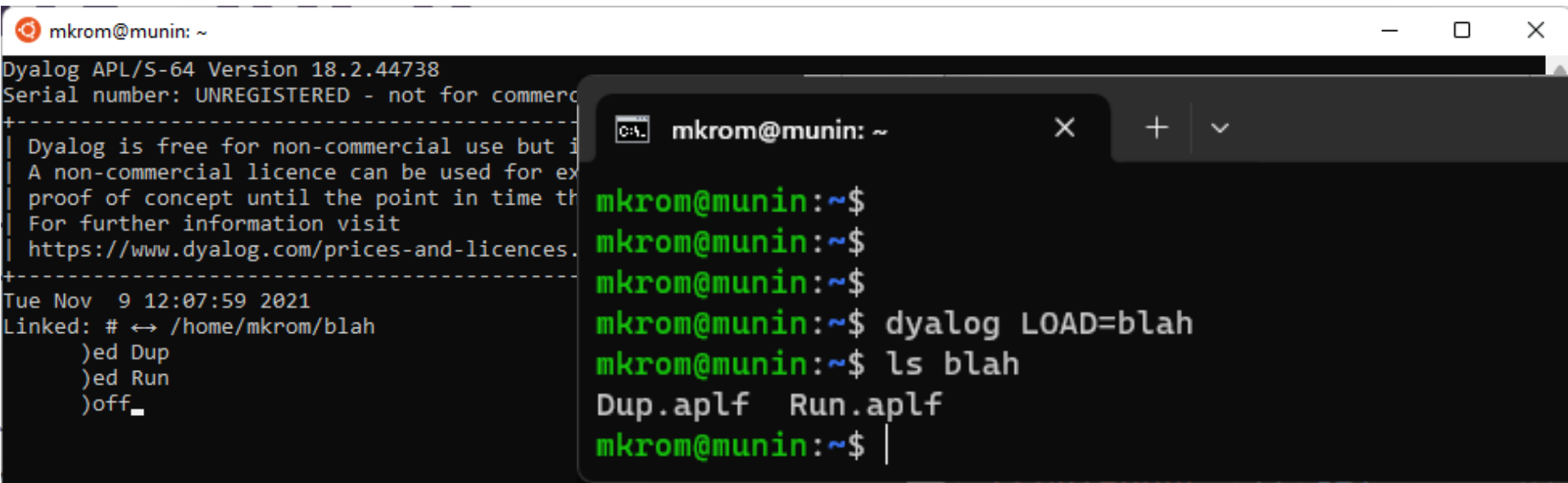


A terminal window titled 'mkrom@mumin: ~' with standard window controls. The output of the 'dyalog' command is displayed on a black background with white text. The output includes the version '18.2.44738', a notice that the software is unregistered and not for commercial use, a dashed box containing a license notice, the current date and time, and the linked directory path.

```
mkrom@mumin: ~  
Dyalog APL/S-64 Version 18.2.44738  
Serial number: UNREGISTERED - not for commercial use  
+-----+  
| Dyalog is free for non-commercial use but is not free software. |  
| A non-commercial licence can be used for experiments and      |  
| proof of concept until the point in time that it is of value.  |  
| For further information visit                                    |  
| https://www.dyalog.com/prices-and-licences.htm |  
+-----+  
Tue Nov 9 12:07:59 2021  
Linked: # ↔ /home/mkrom/blah
```

# Launching from Shell

```
$ dyalog LOAD="blah"
```



The screenshot shows a terminal window titled 'mkrom@munin: ~'. The window displays the output of the 'dyalog' command, which is a version and license notice for Dyalog APL/S-64. The output includes the version number 18.2.44738, a serial number, and a license notice stating that the software is free for non-commercial use. Below the license notice, the date and time 'Tue Nov 9 12:07:59 2021' are shown, followed by a linked command prompt '# ↔ /home/mkrom/blah'. The prompt is followed by a series of commands and their outputs: 'ed Dup', 'ed Run', and 'off'. The prompt is then followed by a series of commands and their outputs: 'mkrom@munin:~\$ dyalog LOAD=blah', 'mkrom@munin:~\$ ls blah', 'Dup.aplf Run.aplf', and 'mkrom@munin:~\$'.

```
mkrom@munin: ~
Dyalog APL/S-64 Version 18.2.44738
Serial number: UNREGISTERED - not for commercial use
-----
Dyalog is free for non-commercial use but it is not free for
A non-commercial licence can be used for example for
proof of concept until the point in time that the software
For further information visit
https://www.dyalog.com/prices-and-licences.
-----
Tue Nov 9 12:07:59 2021
Linked: # ↔ /home/mkrom/blah
)ed Dup
)ed Run
)off_

mkrom@munin:~$
mkrom@munin:~$
mkrom@munin:~$
mkrom@munin:~$ dyalog LOAD=blah
mkrom@munin:~$ ls blah
Dup.aplf Run.aplf
mkrom@munin:~$
```

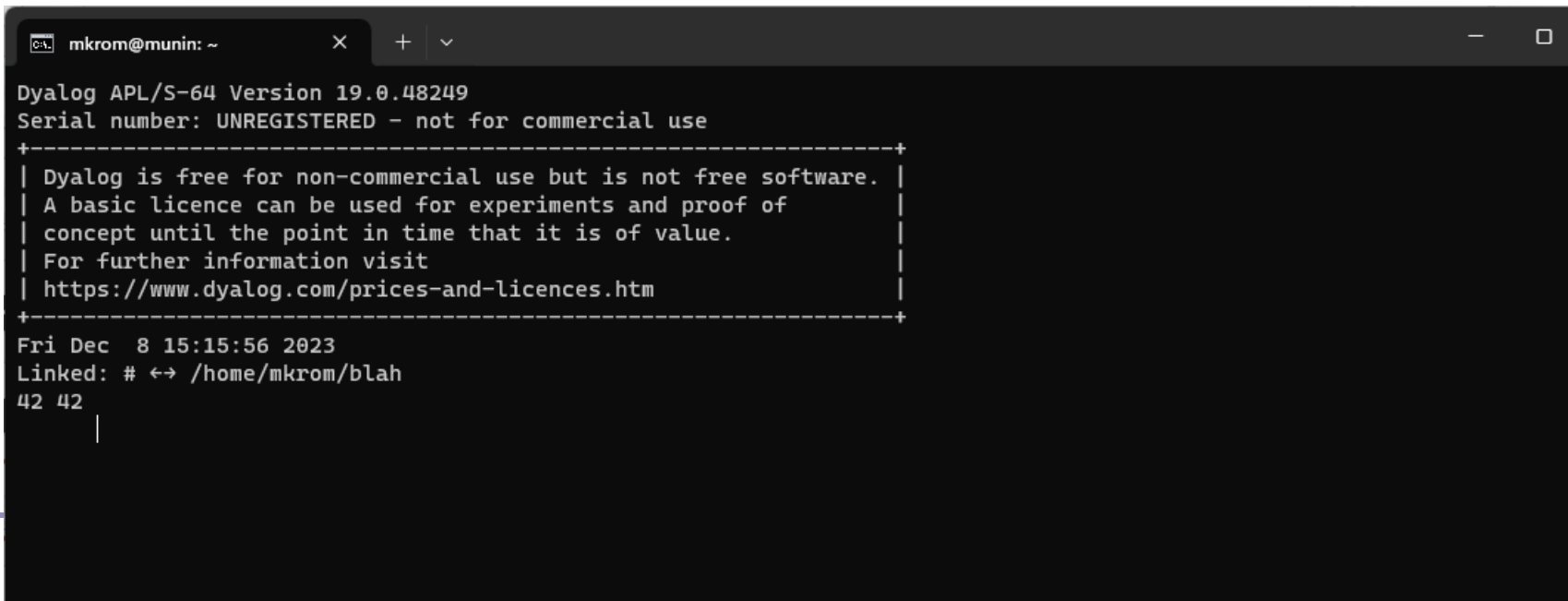
# Command Shell, Continued...

```
mkrom@munin: ~  
mkrom@munin:~$  
mkrom@munin:~$ ls blah  
Dup.aplf  Run.aplf  
mkrom@munin:~$ nano blah/Run.aplf
```

```
GNU nano 6.2      blah/Run.aplf  
Run  
Dup 42  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

# Launching from Shell

```
$ dyalog LOAD=blah
```

A terminal window with a dark background. The title bar shows 'mkrom@munin: ~' and standard window controls. The output of the 'dyalog LOAD=blah' command is displayed in a monospaced font. It includes the version '19.0.48249', a notice about unregistered commercial use, a license disclaimer in a dashed box, the date 'Fri Dec 8 15:15:56 2023', the linked path '/home/mkrom/blah', and the number '42' repeated twice.

```
mkrom@munin: ~  
Dyalog APL/S-64 Version 19.0.48249  
Serial number: UNREGISTERED - not for commercial use  
+-----+  
| Dyalog is free for non-commercial use but is not free software. |  
| A basic licence can be used for experiments and proof of      |  
| concept until the point in time that it is of value.          |  
| For further information visit                                  |  
| https://www.dyalog.com/prices-and-licences.htm |  
+-----+  
Fri Dec 8 15:15:56 2023  
Linked: # ↔ /home/mkrom/blah  
42 42  
|
```

# Load but do not Run ...

Set an empty LX= to avoid calling Run on startup

```
$ dyalog LOAD=blah LX=
```

# Converting an Existing System

```
)load c:\demos\linkdemo\myapp
```



This is a regular workspace

```
)fns
```

```
Main      Mean      Root      StdDev
```

```
; "1 1 0 0<CR"<NL -3
```

```
Main;data
A Compute Mean and StdDev until user inputs an empty array

:Repeat
  ⍵←'Enter some numbers:'
  :If 0≠pdata+⍵
    ⍵←'Mean: ',1⍴Mean data
    ⍵←'StdDev: ',1⍴StdDev data
  :EndIf
:Until 0=⍵data
```

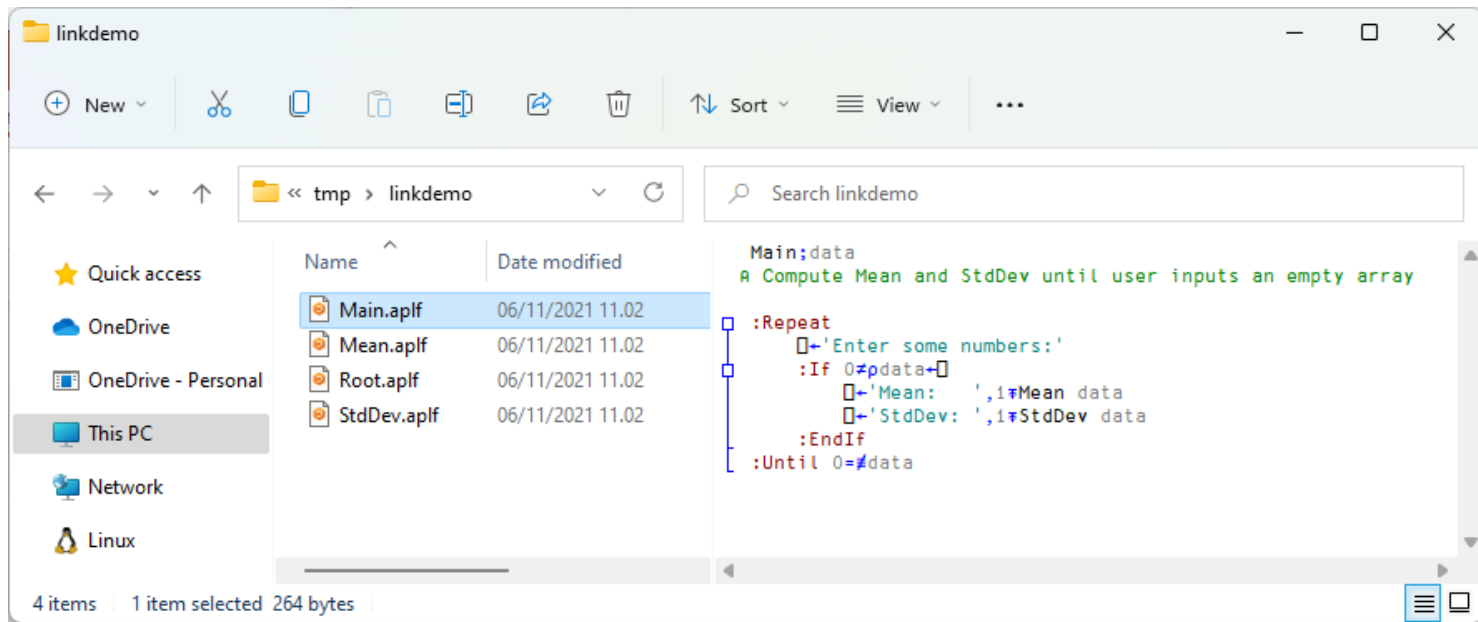
```
mean←Mean vals;sum
sum←+/vals
mean←sum÷1⍴p,vals
```

```
Root←{α+2
      ω*÷α}
```

```
StdDev←{2 Root(+.×÷÷p),ω-Mean ω}
```

# Exporting (from WS to File)

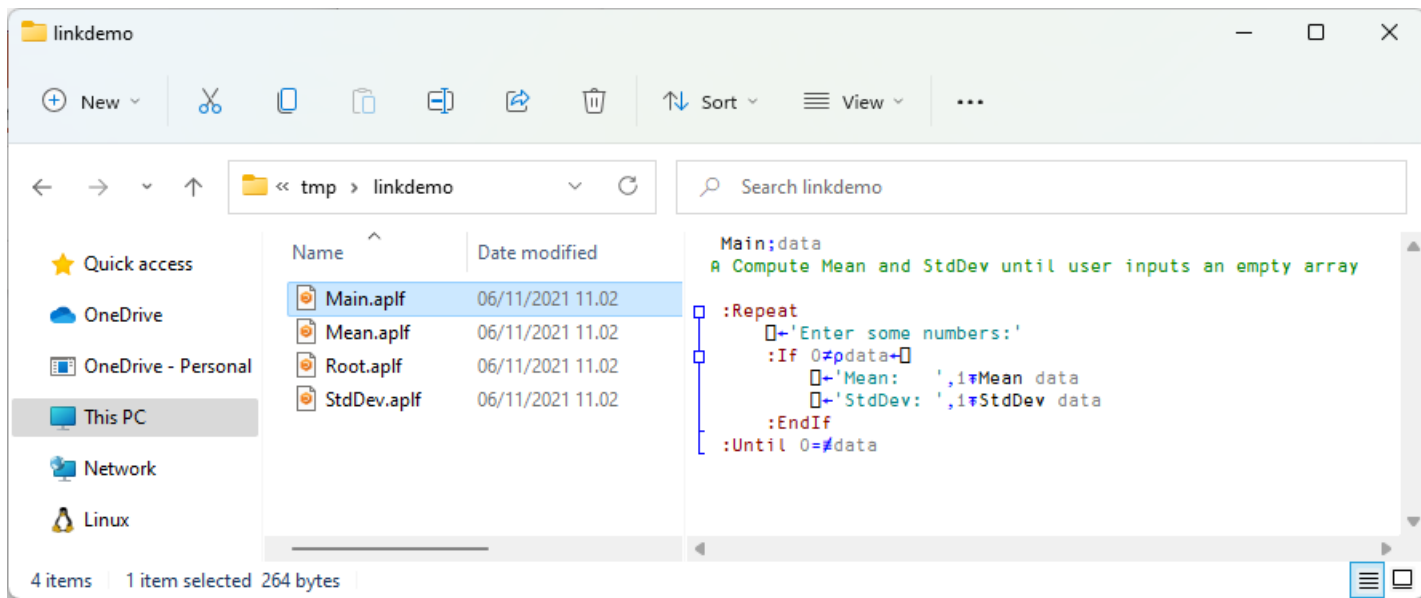
```
]link.export # c:\tmp\linkdemo  
Exported: # → c:\tmp\linkdemo
```





# [System] Variables

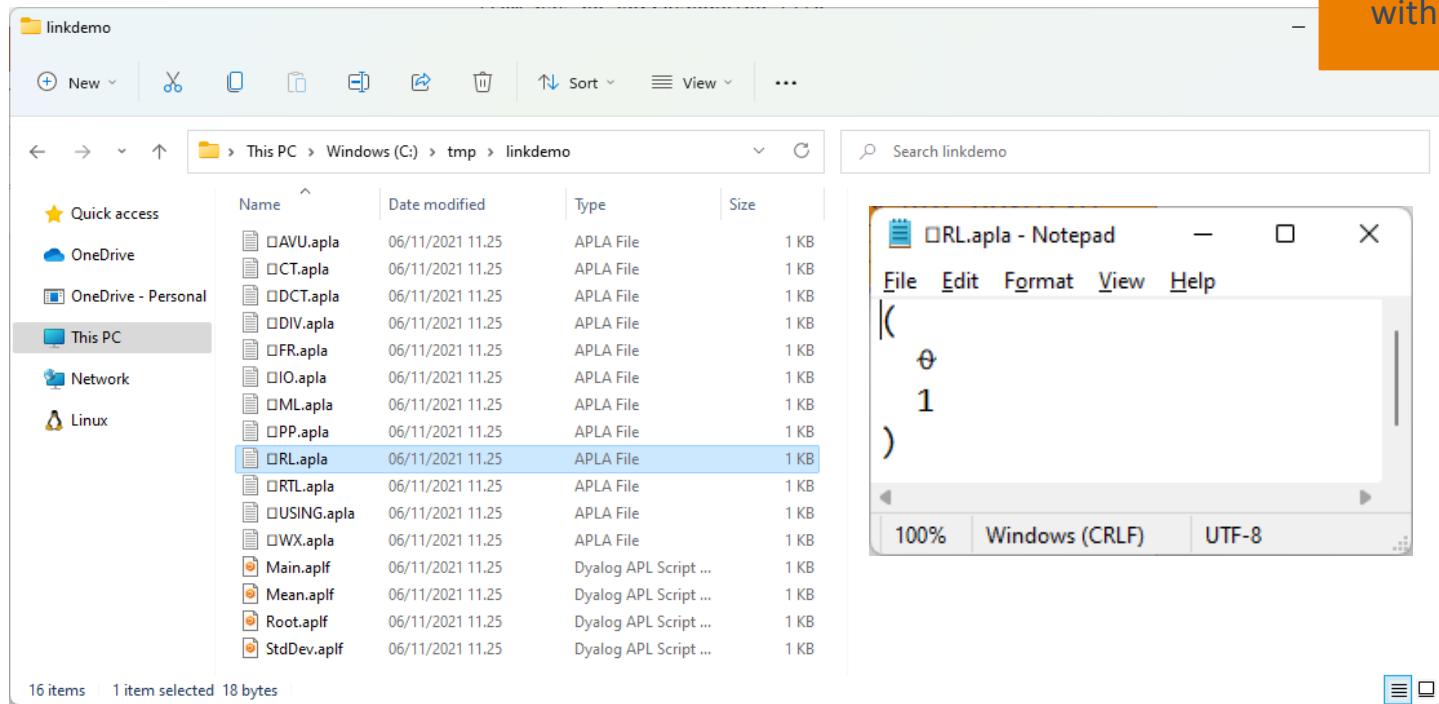
```
]link.add IO ML  
Added: #.IO #.ML
```



# -sysvars

```
]export # c:\tmp\linkdemo -sysvars  
Exported: # → c:\tmp\linkdemo
```

Exports all  
system variables  
with namespace scope



# ]link.create replaces )LOAD

```
]link.create # c:\tmp\linkdemo  
Linked: # ↔ c:\tmp\linkdemo
```

```
)fns  
Main      Mean      Root      StdDev
```

```
      Main  
Enter some numbers:  
□:
```

```
      1 2 3 4  
Mean:      2.5  
StdDev:    1.1
```

```
Enter some numbers:  
□:
```

```
0
```



Shows that link is  
Bi-directional

(Currently requires .NET)

# Adding Structure

```
)fns
Main      Mean      Root      StdDev

; "1 1 0 0<CR"<NL -3
```

```
Main;data
A Compute Mean and StdDev until user inputs an empty array

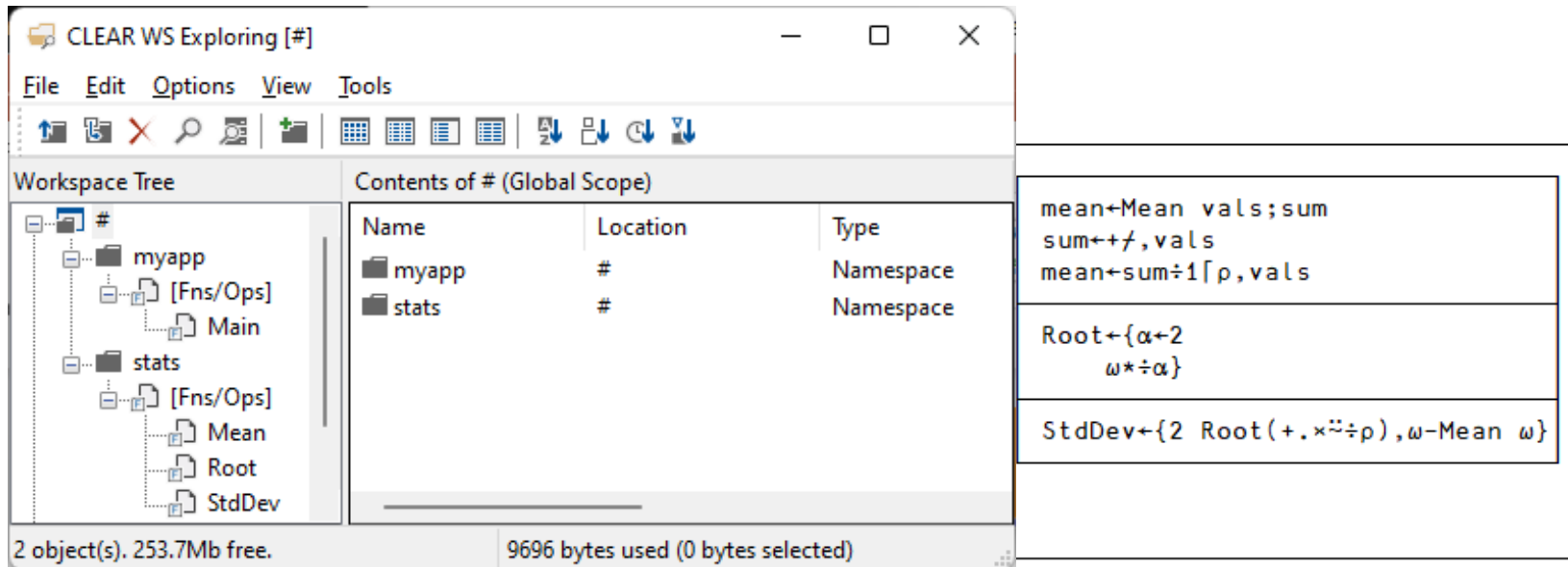
:Repeat
  ⍵←'Enter some numbers:'
  :If 0≠pdata←⍵
    ⍵←'Mean: ',1⍴Mean data
    ⍵←'StdDev: ',1⍴StdDev data
  :EndIf
:Until 0=≠data
```

```
mean←Mean vals;sum
sum←+/vals
mean←sum÷1[ρ,vals
```

```
Root←{α←2
      ω÷α}
```

```
StdDev←{2 Root(+.×÷ρ),ω-Mean ω}
```

# Adding Structure



The screenshot shows the CLEAR WS Exploring window with a menu bar (File, Edit, Options, View, Tools) and a toolbar. The Workspace Tree on the left shows a hierarchy: # (Global Scope) containing myapp (Namespace) and stats (Namespace). myapp contains [Fns/Ops] and Main. stats contains [Fns/Ops], Mean, Root, and StdDev. The Contents of # (Global Scope) table lists the following:

Name	Location	Type
myapp	#	Namespace
stats	#	Namespace

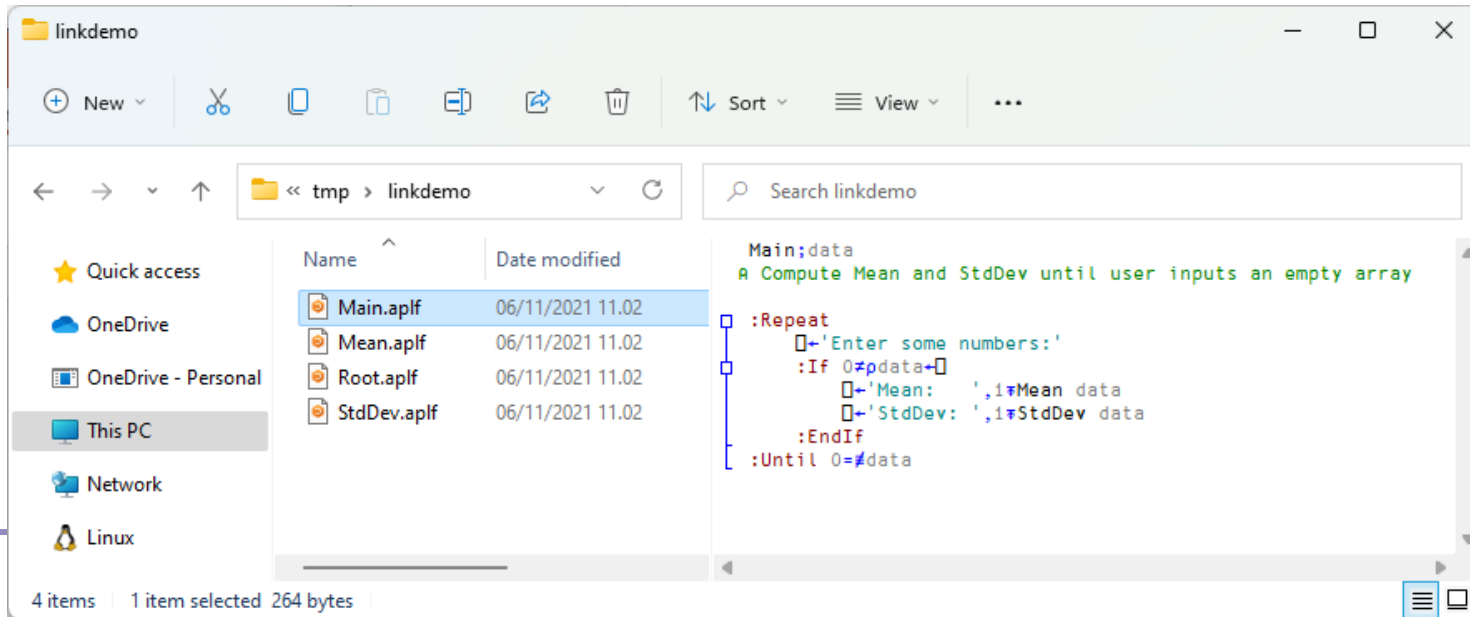
Below the table, three code snippets are displayed:

```
mean←Mean vals;sum  
sum←+/vals  
mean←sum÷1[ρ,vals  
  
Root←{α←2  
ω*÷α}  
  
StdDev←{2 Root(+.×÷÷ρ),ω-Mean ω}
```

The status bar at the bottom indicates "2 object(s). 253.7Mb free." and "9696 bytes used (0 bytes selected)".

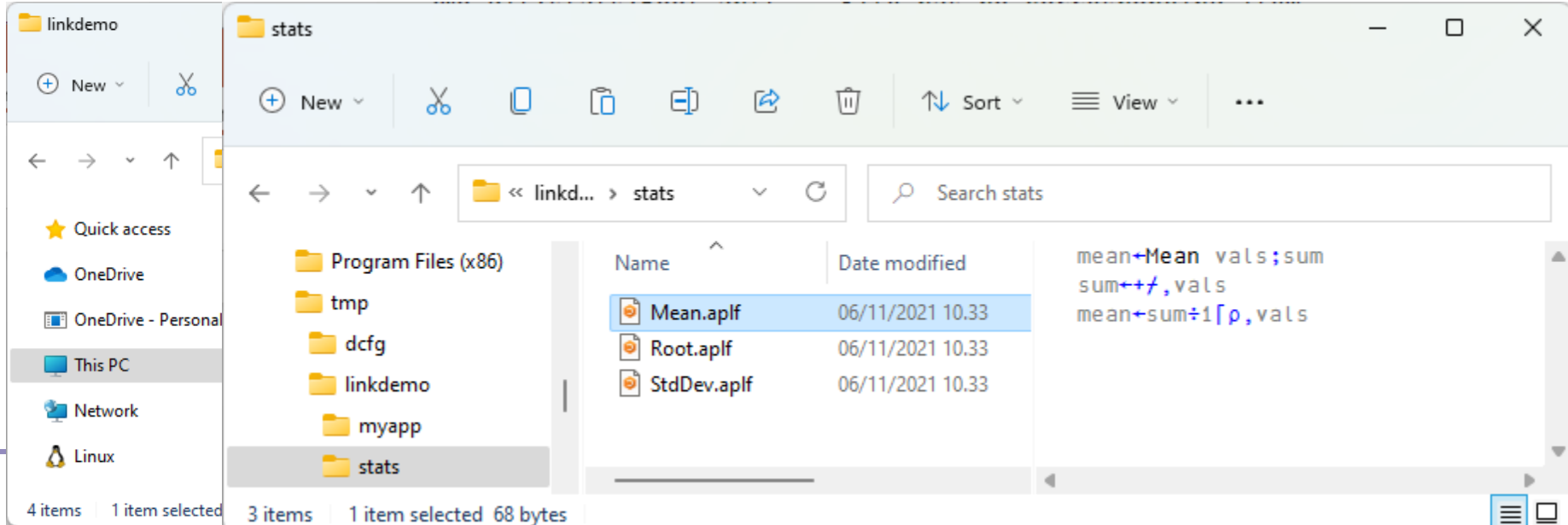
# Adding Structure

- You could change the directory structure using File Manager...



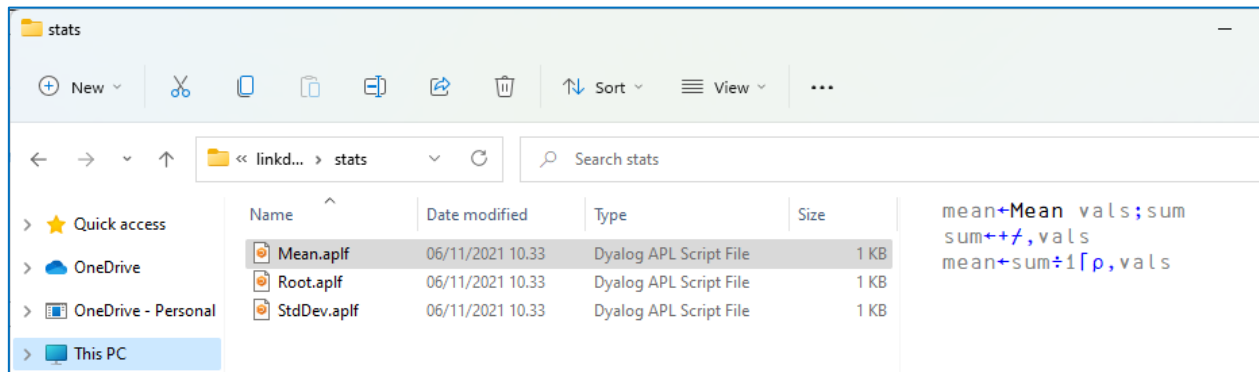
# Adding Structure

- You could change the directory structure using File Manager...



OR...

```
ⓂMKDIR 'c:\tmp\linkdemo\myapp'  
ⓂMKDIR 'c:\tmp\linkdemo\stats'  
  
]link.export Main c:\tmp\linkdemo\myapp  
Exported: #.Main → c:/tmp/linkdemo/myapp/Main.aplf  
  
statfns←'Mean' 'Root' 'StdDev'  
;{ⓂSE.Link.Export ω 'c:\tmp\linkdemo\stats'}"statfns  
Exported: #.Mean → c:/tmp/linkdemo/stats/Mean.aplf  
Exported: #.Root → c:/tmp/linkdemo/stats/Root.aplf  
Exported: #.StdDev → c:/tmp/linkdemo/stats/StdDev.aplf
```

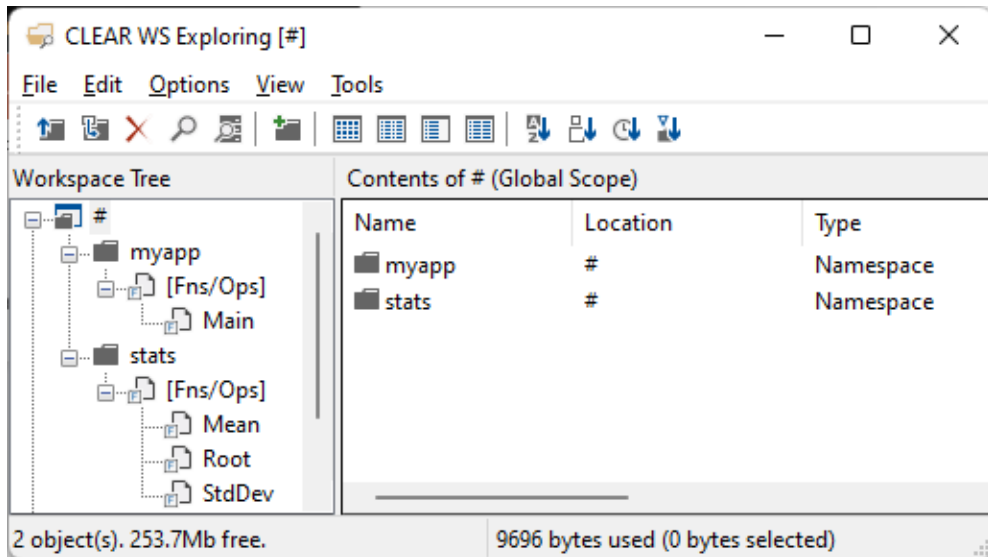


Oh, and  
delete the  
old files



# Adding Structure

```
Language Bar  
+ - * / % & @ ! ? [ ] { } ~ = < > > ≡ ≠ √ ^ ⌈ ⋄ ↓ ↑ ↵ ⊞ ⊠ ⊢ ⊣ ⊤ ⊥ ⊦ ⊧ ⊨ ⊩ ⊪ ⊫ ⊬ ⊭ ⊮ ⊯ ⊰ ⊱ ⊲ ⊳ ⊴ ⊵ ⊶ ⊷ ⊸ ⊹ ⊺ ⊻ ⊼ ⊽ ⊾ ⊿ ⊿ ⊿ ⊿  
Dyalog APL/W-64 Version 18.2.44714  
Serial number: 000013 - pre-release software  
Mon Nov 8 08:56:13 2021  
  
    ]link.create # c:\tmp\linkdemo  
Linked: # ↔ c:\tmp\linkdemo  
myapp.Main  
Enter some numbers:  
⎕:  
    10?10  
VALUE ERROR: Undefined name: Mean  
Main[6] ⎕←'Mean: ',1≠Mean data
```



Namespace  
structure  
≡  
Directory  
Structure \*

- \* Unless you use `-flatten`

# Adding Structure

The screenshot displays the Dyalog APL/W-64 Version 18.2.4714 interface. The top pane shows the execution of a program that creates a link to a demo file and enters a loop for computing statistics. The bottom pane shows the debugger with a step-through view of the same code, highlighting the loop structure and the current state of the program.

```
Dyalog APL/W-64 Version 18.2.4714
Serial number: 000013 - pre-release software
Mon Nov 8 08:56:13 2021

]link.create # c:\tmp\linkdemo
Linked: # ↔ c:\tmp\linkdemo
myapp.Main
Enter some numbers:
[]:
10?10
VALUE ERROR: Undefined name: Mean
Main[6] []←'Mean: ',1⊖Mean data
^
```

**Debugger**

Search...

Main;data;ST  
A Compute Mean and StdDev until user inputs an empty array

```
ST+#.stats
:Repeat
  []←'Enter some numbers:'
  :If 0≠pdata+[]
    []←'Mean: ',1⊖ST.Mean data
    []←'StdDev: ',1⊖ST.StdDev data
  :EndIf
:Until 0=ndata
```

Modified Function Pos: 8/11,27

Ready...

CurObj: STStdDev (Undefined) 8:1 DDQ:0 TRAP SI:

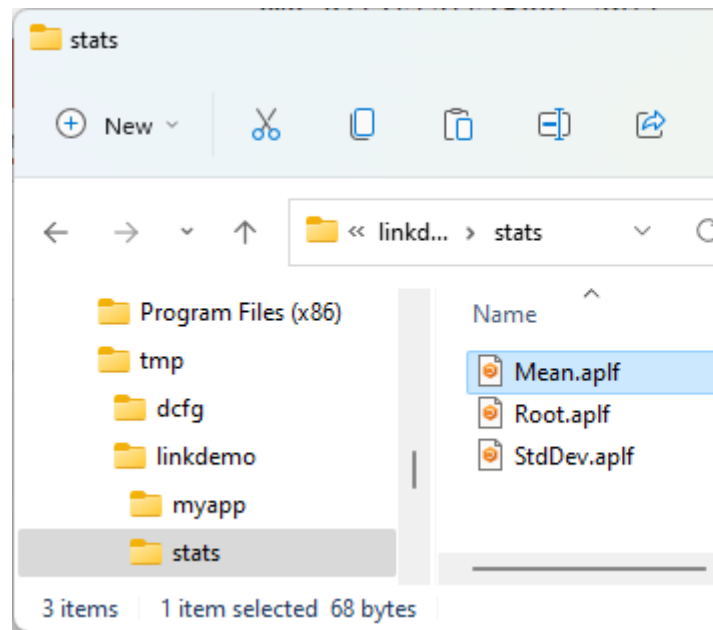
STstack (Tid: Ttd:0)

Main[6]\*[]←'Mean: '

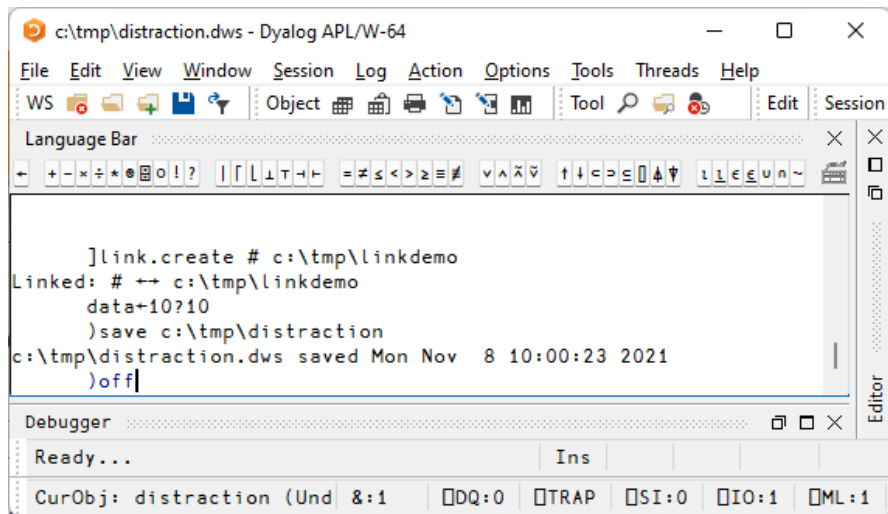
# The -flatten switch

- You can have a directory structure to organize your code
  - ...but ignore it when loading into the workspace
- No namespace structure is created

```
]link.create # c:\tmp\linkdemo -flatten
Linked: # ↔ c:\tmp\linkdemo
)fn
Main      Mean      Root      StdDev
```



# Link vs. Workspaces



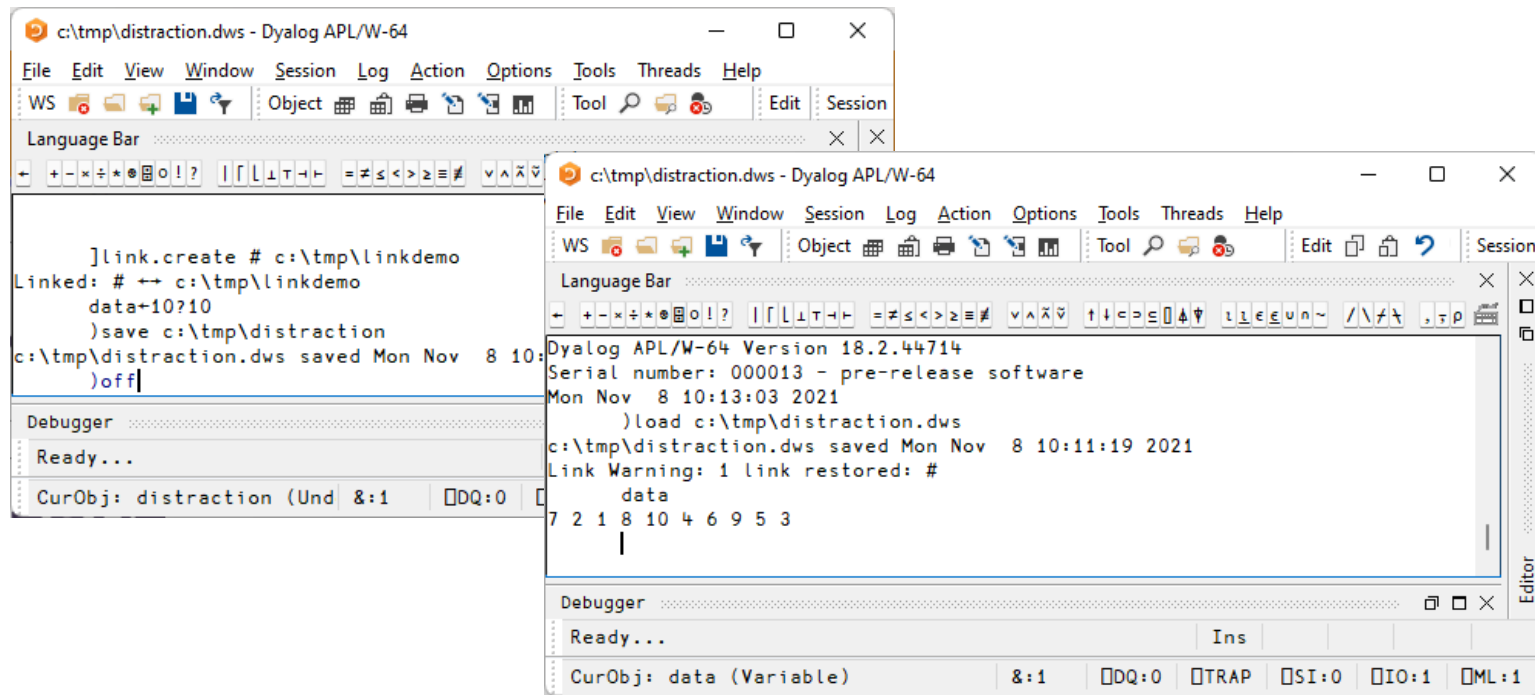
The screenshot shows the Dyalog APL/W-64 workspace editor. The title bar indicates the file is 'c:\tmp\distraction.dws'. The menu bar includes File, Edit, View, Window, Session, Log, Action, Options, Tools, Threads, and Help. The toolbar contains icons for workspace operations. The Language Bar shows various APL symbols. The main editor area contains the following code:

```
]link.create # c:\tmp\linkdemo  
Linked: # ↔ c:\tmp\linkdemo  
data←10?10  
)save c:\tmp\distraction  
c:\tmp\distraction.dws saved Mon Nov 8 10:00:23 2021  
)off|
```

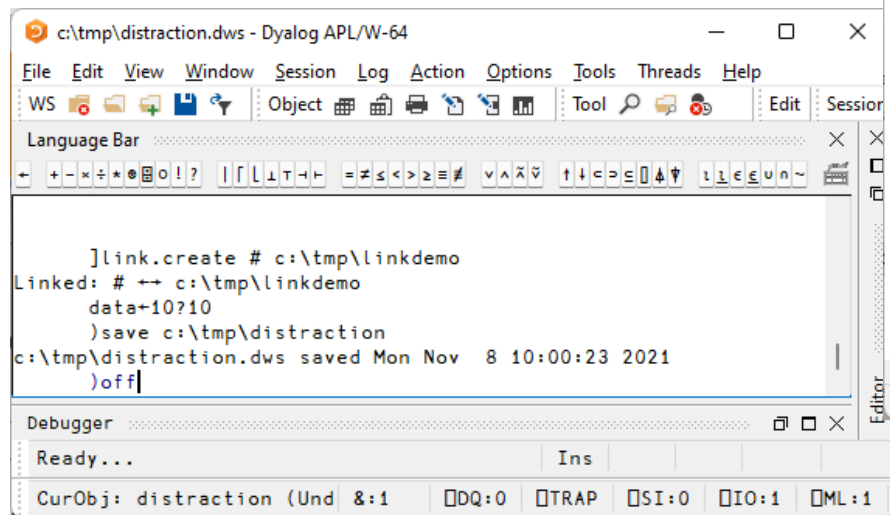
The Debugger window at the bottom shows 'Ready...' and 'Ins' buttons. The status bar at the bottom displays 'CurObj: distraction (Und &:1)' and various flags: ☐DQ:0, ☐TRAP, ☐SI:0, ☐IO:1, ☐ML:1.



# Link vs. Workspaces



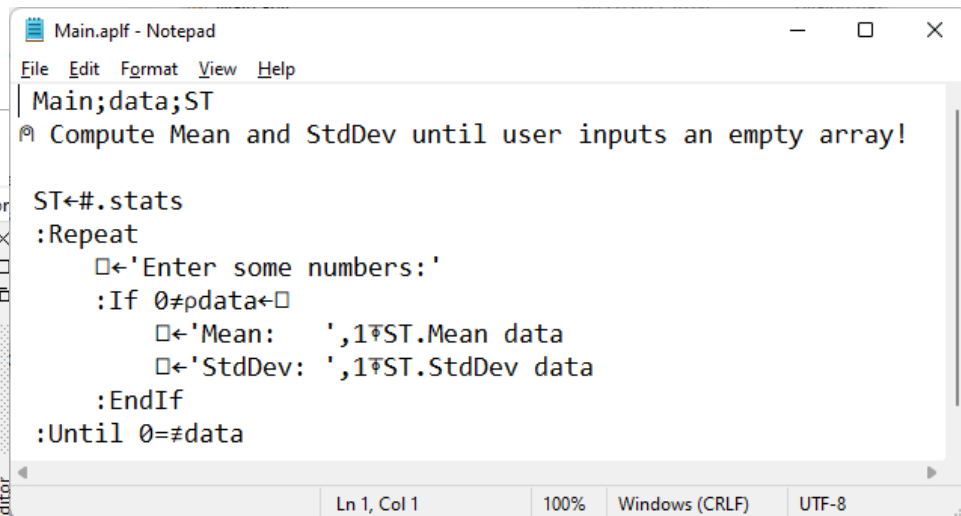
# Saved Workspaces



The screenshot shows the Dyalog APL/W-64 workspace editor. The title bar indicates the file is 'c:\tmp\distraction.dws'. The menu bar includes File, Edit, View, Window, Session, Log, Action, Options, Tools, Threads, and Help. The toolbar contains icons for workspace operations. The main text area displays the following APL code:

```
]link.create # c:\tmp\linkdemo  
Linked: # ↔ c:\tmp\linkdemo  
data←10?10  
)save c:\tmp\distraction  
c:\tmp\distraction.dws saved Mon Nov 8 10:00:23 2021  
)off|
```

At the bottom, a debugger window shows 'Ready...' and status information: 'CurObj: distraction (Und &:1)' and various flags (DQ, TRAP, SI, IO, ML) all set to 0 or 1.

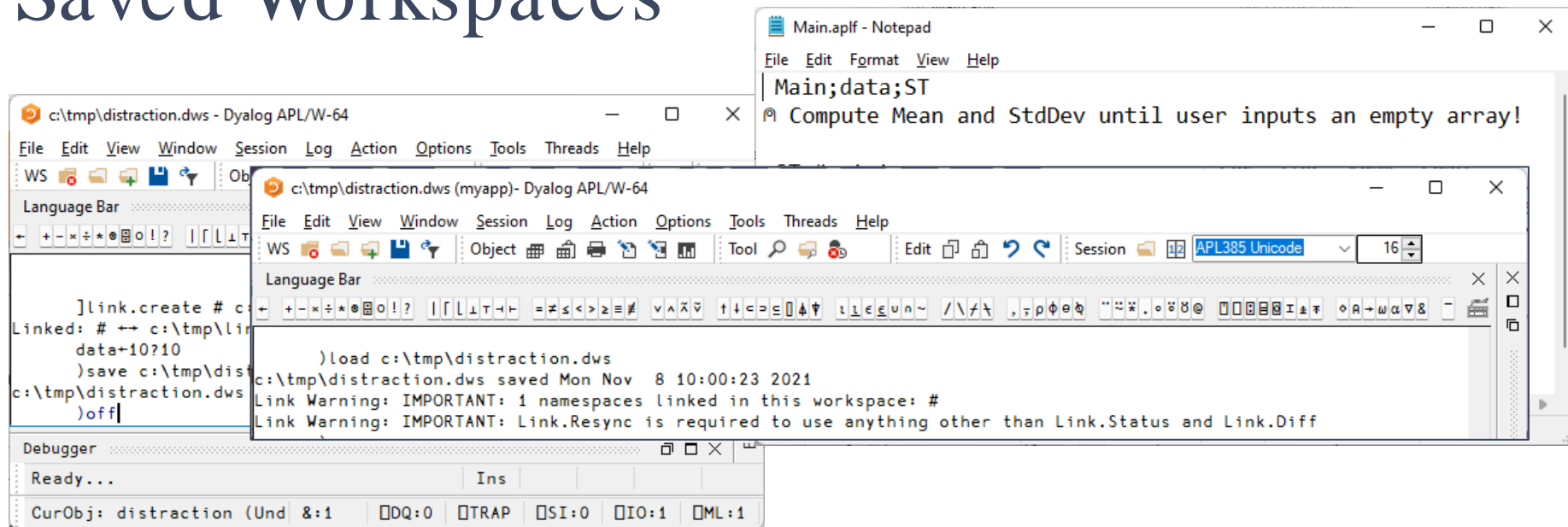


The screenshot shows a Notepad window titled 'Main.aplf - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The text area contains the following APL code:

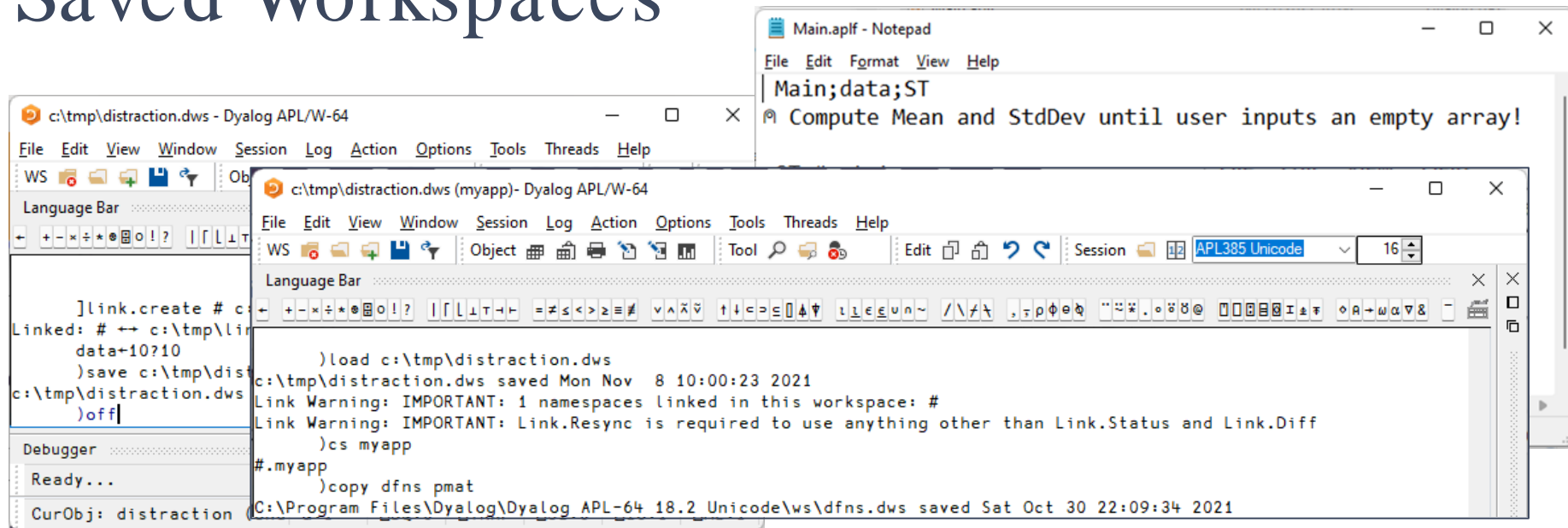
```
Main;data;ST  
Ⓜ Compute Mean and StdDev until user inputs an empty array!  
  
ST←#.stats  
:Repeat  
  Ⓜ←'Enter some numbers:'  
  :If 0≠pdata←Ⓜ  
    Ⓜ←'Mean: ',1ⓂST.Mean data  
    Ⓜ←'StdDev: ',1ⓂST.StdDev data  
  :EndIf  
:Until 0=#data
```

The status bar at the bottom indicates 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

# Saved Workspaces

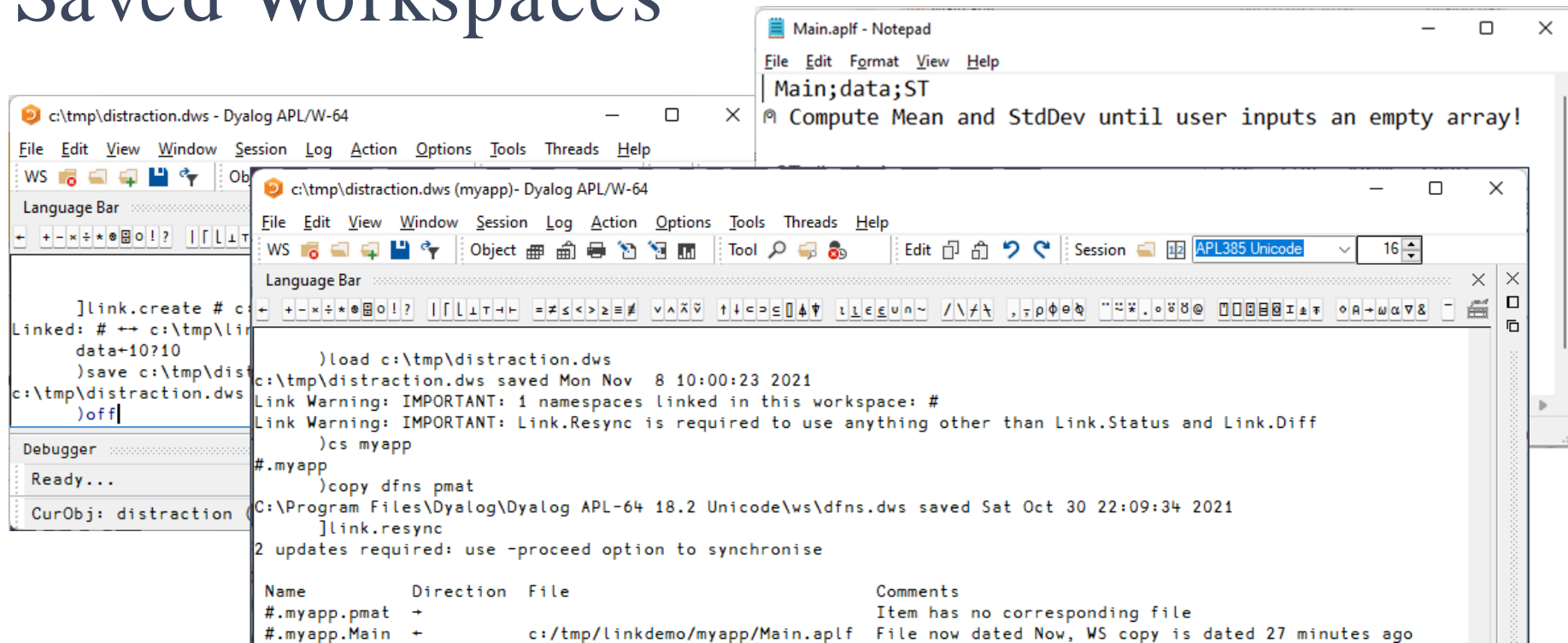


# Saved Workspaces

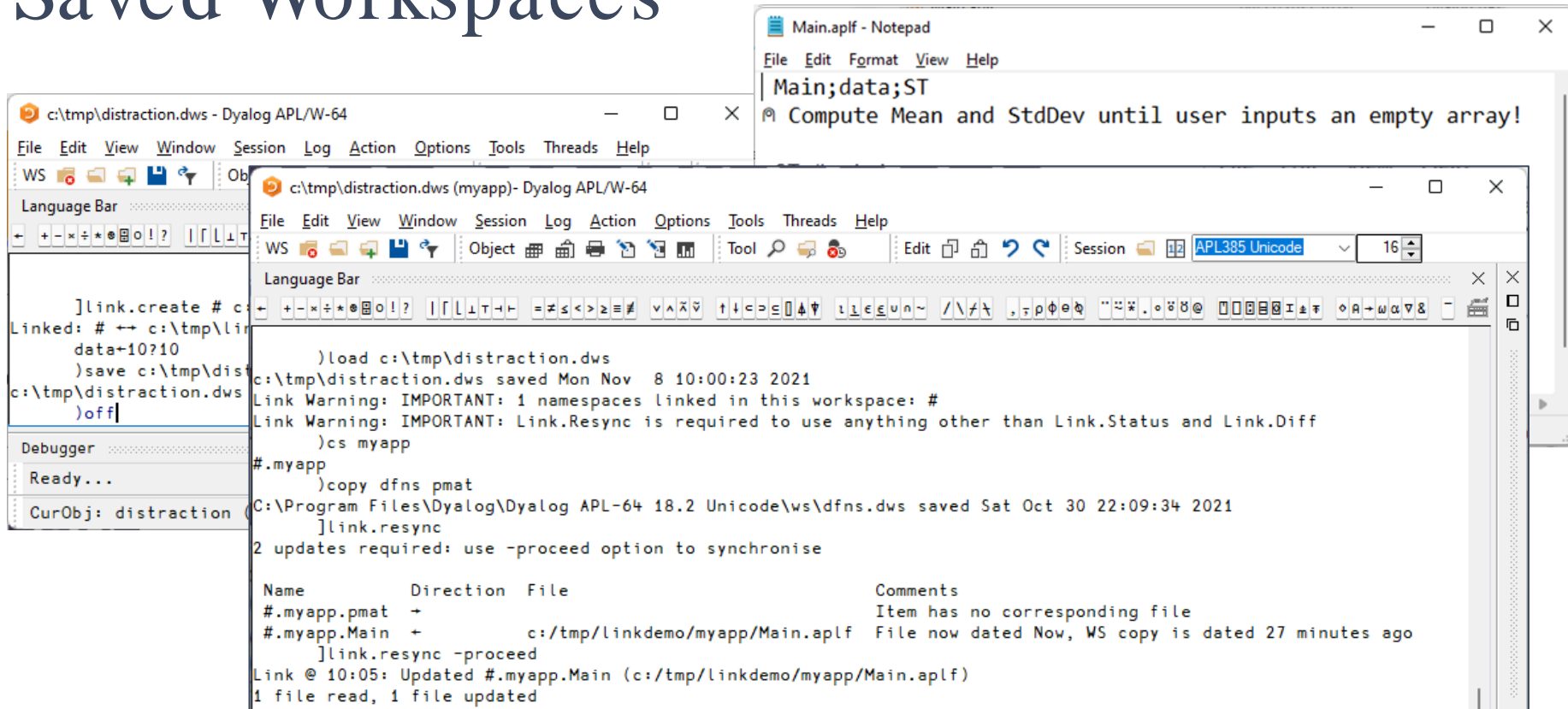




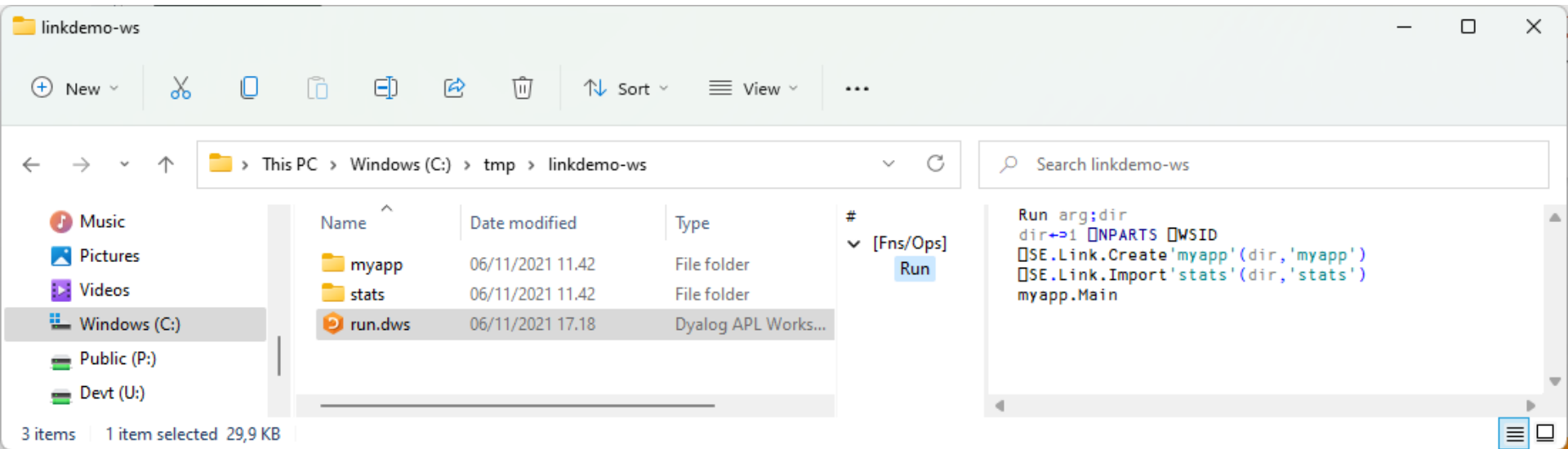
# Saved Workspaces



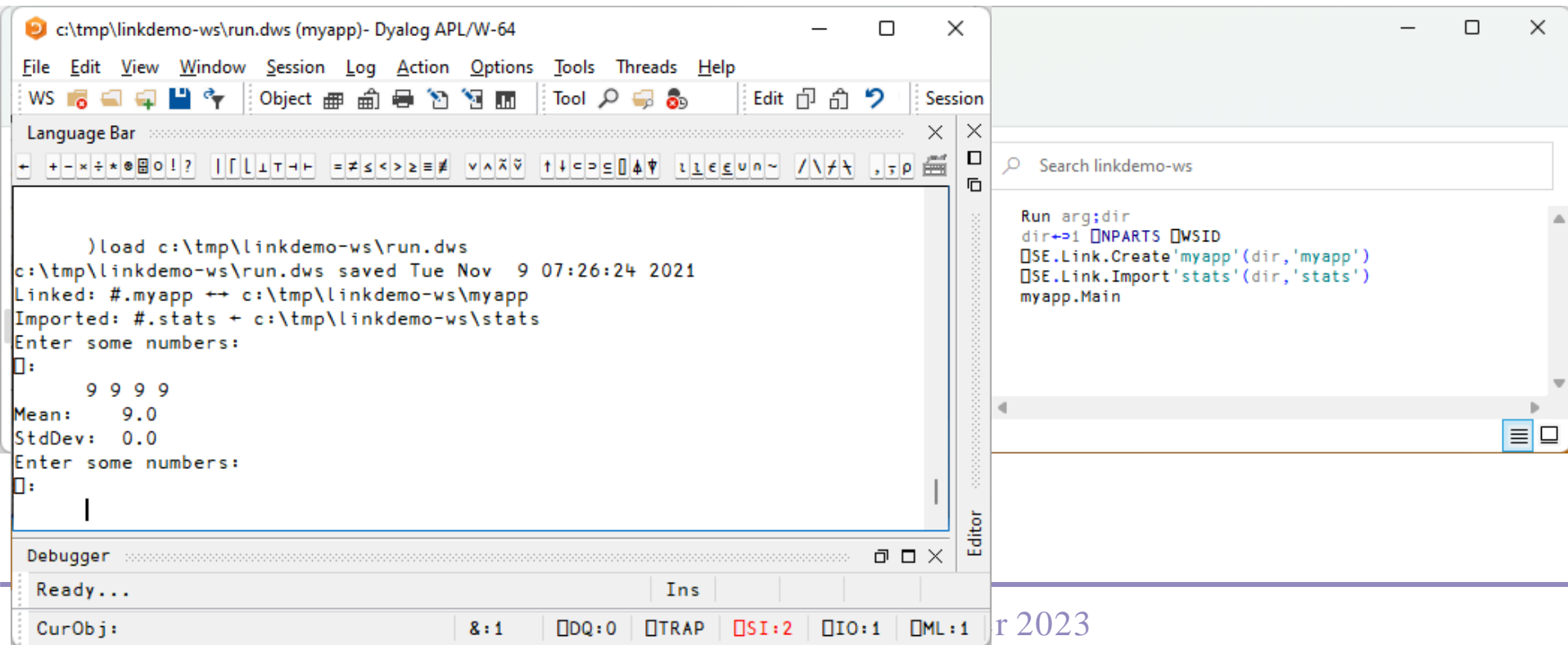
# Saved Workspaces



# Optional "Bootstrap" Workspace



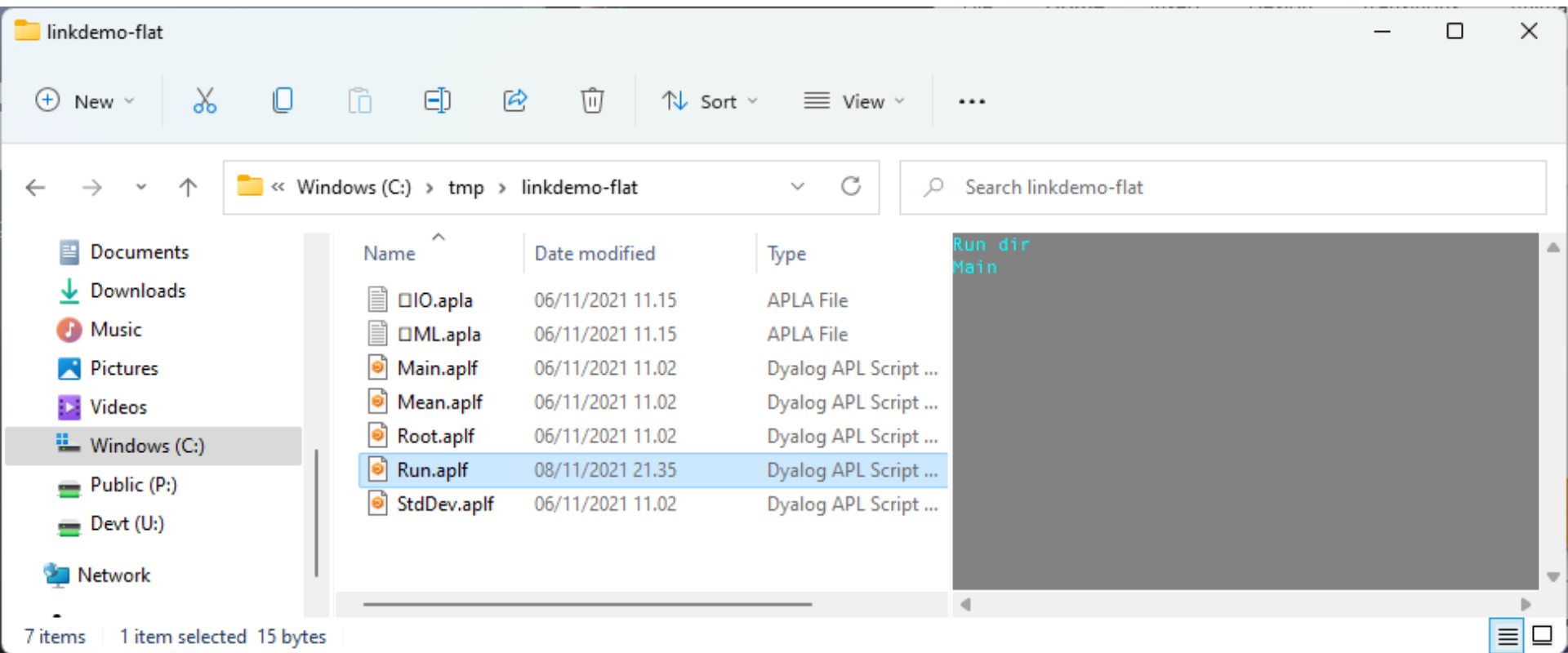
# Optional "Bootstrap" Workspace



# Saved Workspaces

- ✧ OK for
  - ✧ Pausing work
  - ✧ Crash analysis
- ✧ Not recommended as
  - ✧ Storage for source code

# Launching using Text Only



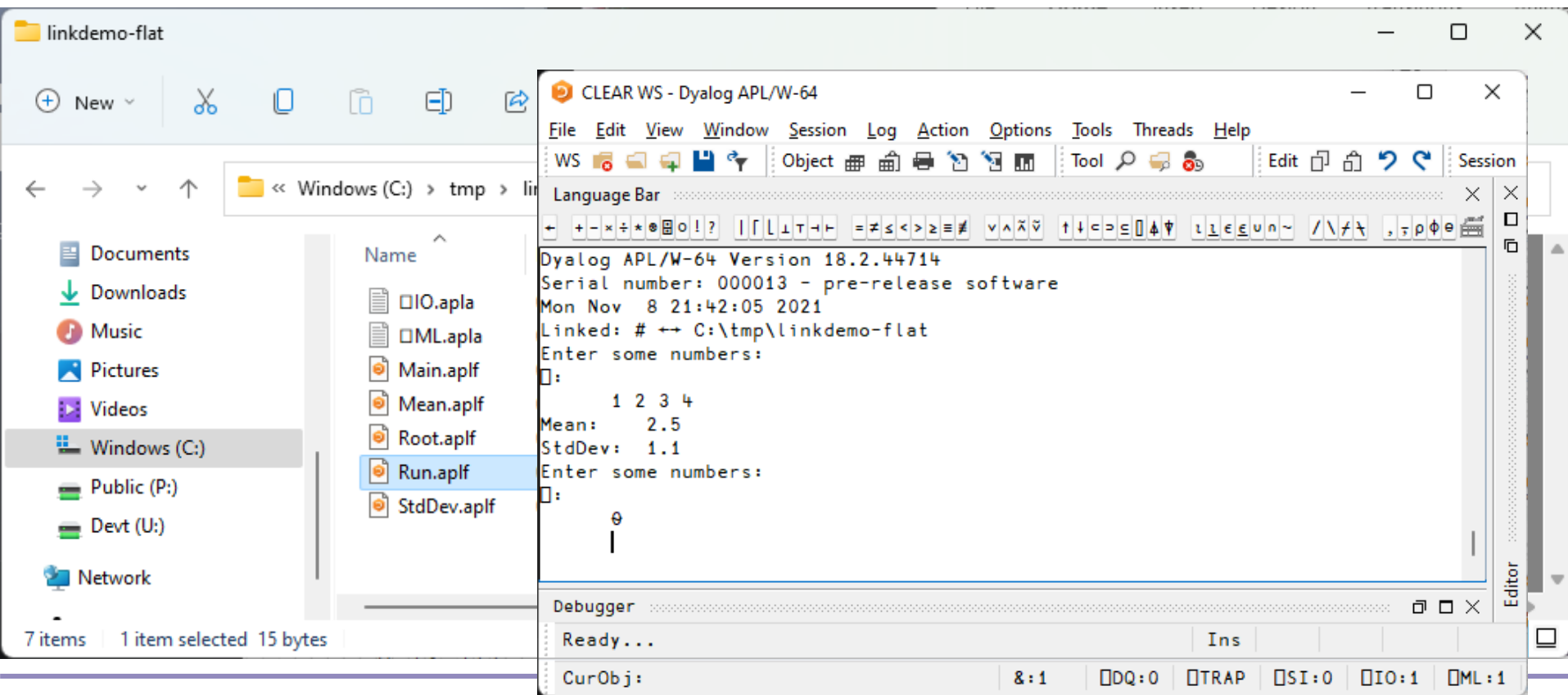
# Launching using Text Only

The screenshot shows a Windows File Explorer window for the folder 'linkdemo-flat'. The address bar shows the path 'Windows (C:) > tmp > linkdemo-flat'. The left sidebar shows the 'Windows (C:)' drive selected. The main area displays a table of files:

Name	Date modified	Type
IO.apla	06/11/2021 11.15	APLA File
ML.apla	06/11/2021 11.15	APLA File
Main.aplf	06/11/2021 11.02	Dyalog APL Script ...
Mean.aplf	06/11/2021 11.02	Dyalog APL Script ...
Root.aplf	06/11/2021 11.02	Dyalog APL Script ...
Run.aplf	08/11/2021 21.35	Dyalog APL Script ...
StdDev.aplf	06/11/2021 11.02	Dyalog APL Script ...

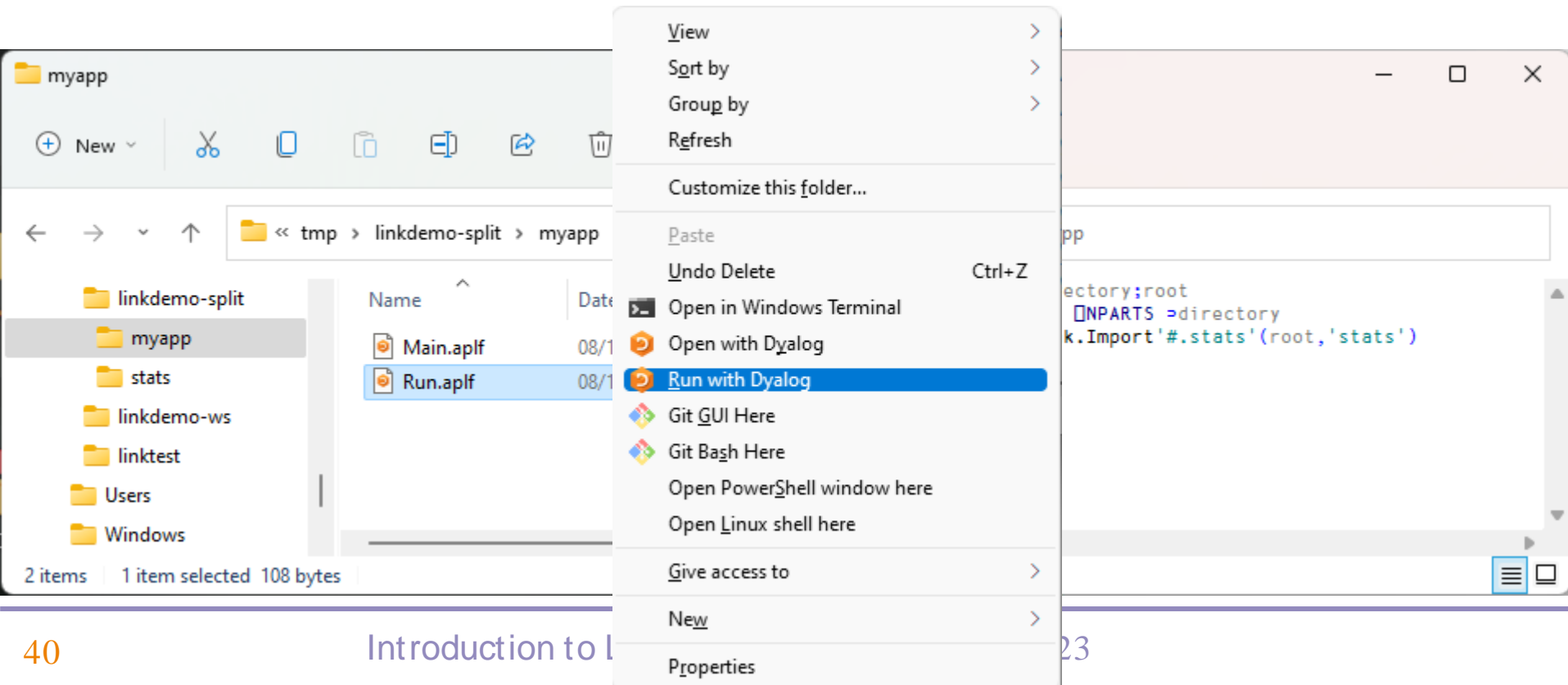
The 'Run.aplf' file is selected. A context menu is open, showing options like 'Run with Dyalog' and 'Open in Windows Terminal'. The status bar at the bottom shows '7 items | 1 item selected 15 bytes'.

# Launching using Text Only

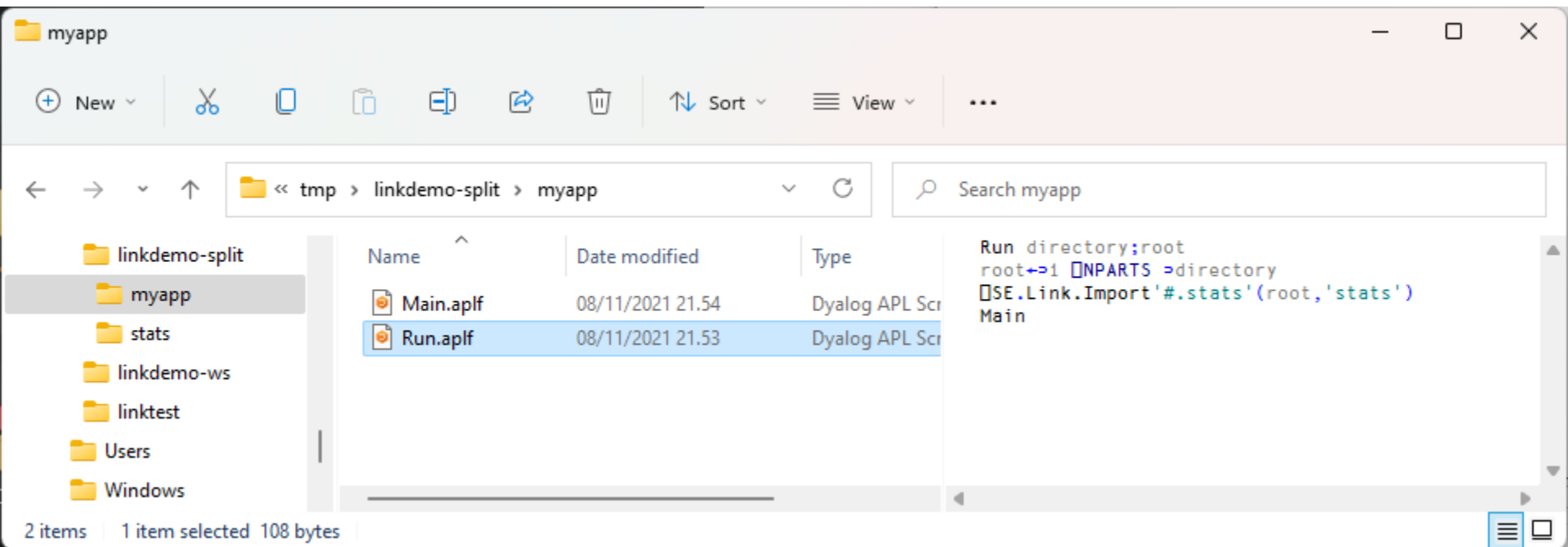




# Launch Directory is passed to Run



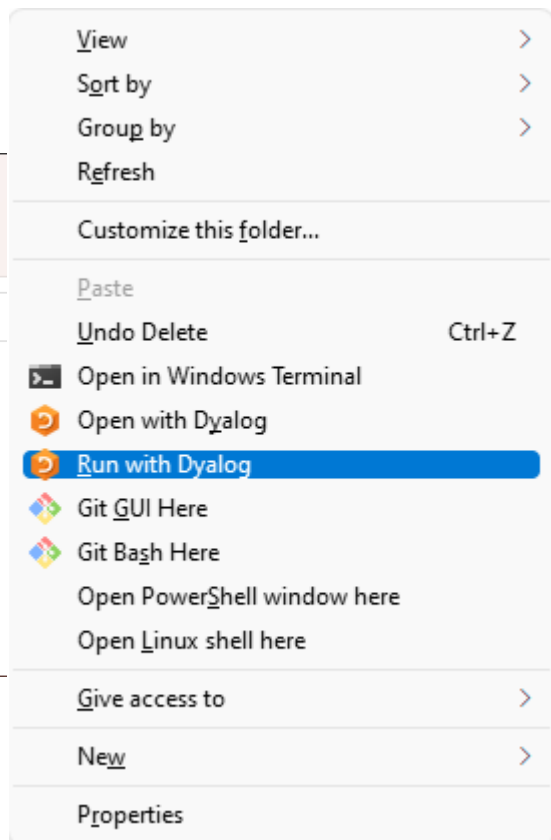
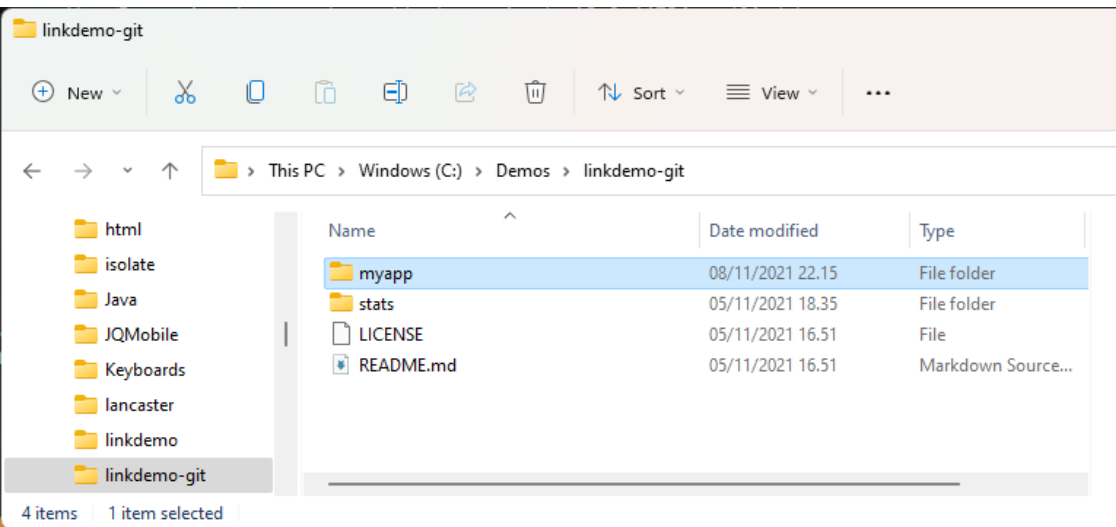
# Launch Directory is passed to Run



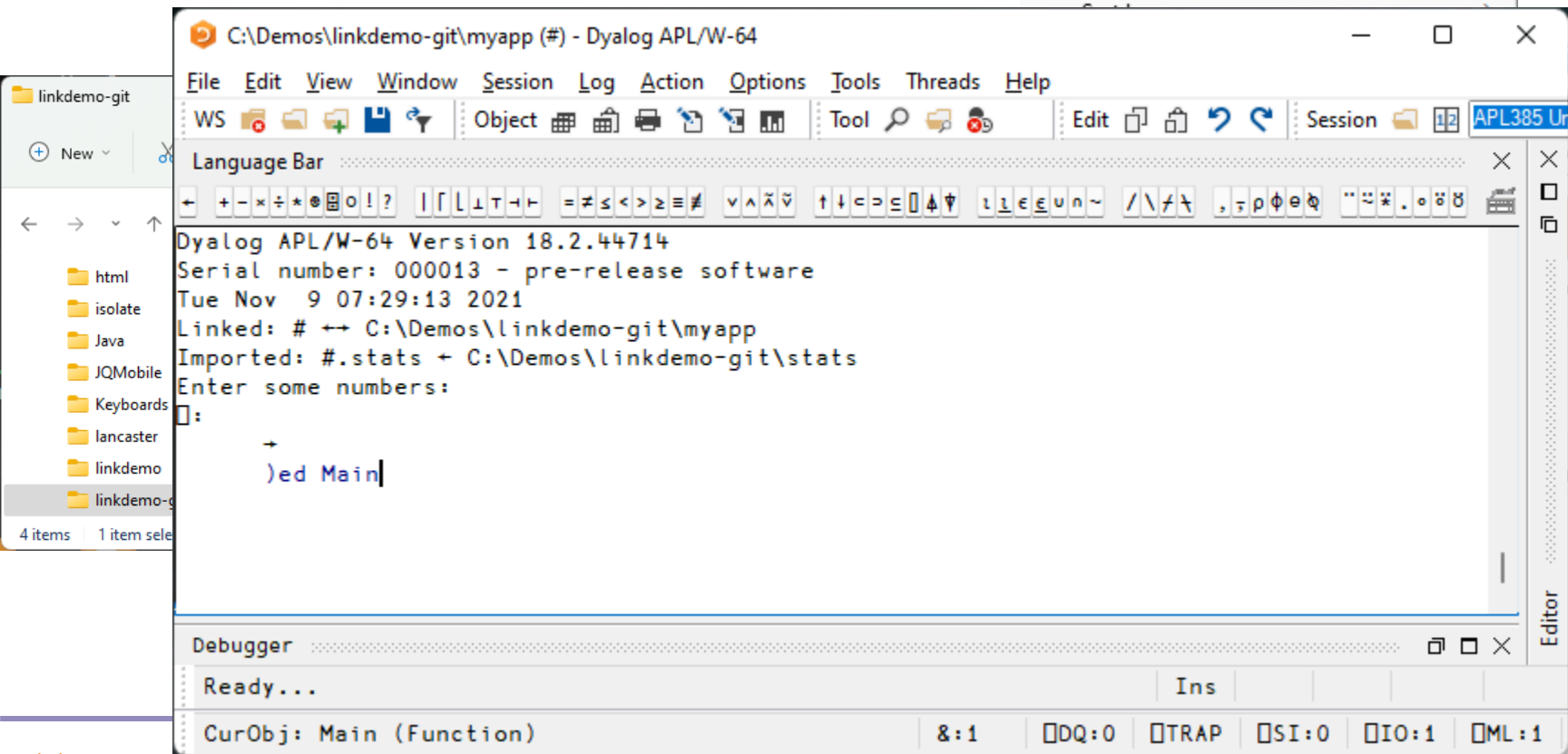
# Reminder: Launching from Shell

```
$ dyalog LOAD="/mnt/c/linkdemo"
```

# Git Hub, VS Code + Link



# Git Hub, VS Code + Link



Visual Studio Code interface showing the Source Control panel on the left and the Main.aplf file in the editor.

**SOURCE CONTROL**

Message (Ctrl+Enter to commit on 'main')

Changes (1)

Main.aplf myapp

**COMMITTS** 0: 2t • main • Last fetch... ↑ ↓ ↺ ↻ ↺ ↻

- Compare Working Tree with <branch, tag, or ref>
- Changes to push to origin on GitHub 2 commits
- Use LaunchDir if not Run You, 4 minutes ago
- Take advantage of LINK\_DIR and LINK\_RUN You, 8 minutes ago
- ⌂ origin ► Add sysvars, remove Directions You, 3 days ago
- Clean up pre-Dyalog'21 You, 3 days ago

**FILE HISTORY**

**BRANCHES**

**REOTES**

**STASHES**

**TAGS**

**SEARCH & COMPARE**

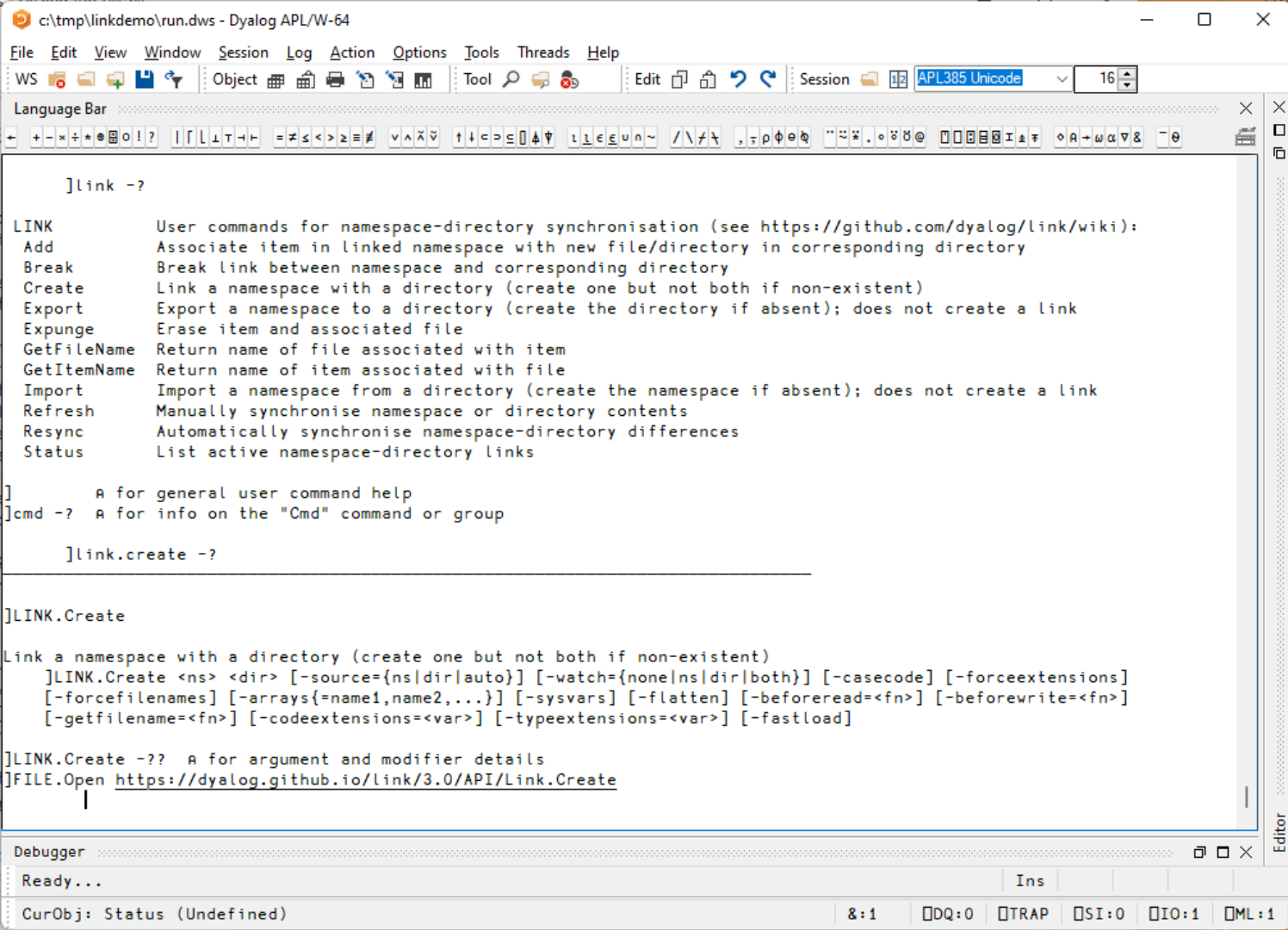
**Main.aplf (Working Tree) M X**

```

1 Main;data
2 A Compute Mean and StdDev until user inputs an empty ar
3
4 ST←#.stats
5 :Repeat
6   □←'Enter some numbers:'
7   :If 0≠pdata+□
8     □←'Mean: ',1⌈ST.Mean data
9     □←'StdDev: ',1⌈ST.StdDev data
10    :EndIf
11 :Until 0=≠data
12
13+ □←'Thank you and Goodbye!'
14+

```

Visual Studio Code status bar: main\* 0: 2t 0 0 A 0 You, seconds ago Ln 13, Col 1 Spaces: 4 UTF-8 CRLF APL



Link.Create - Link User Guide

dialog.github.io/link/3.0/API/Link.Create/

AppsLinkFlying & SailingDyalogCloudExercisesSBOTravelLinuxSportAPLProductivityGit

Reading list

DYALOG

Link User Guide

Search

Dyalog/Link  
6 Stars · 8 Forks

Link User Guide

Overview

Introduction

Technical Details and Limitations

Workspaces

History of source files as text in Dyalog

Install and Upgrade

Installation

Upgrading to Link 3.0

Change History

Working with Link

Basic Usage

Setting Up Your Environment

Converting an Existing Workspace to use Link

Migrating from SALT to Link

API & Command Reference

API Overview

Link.Add

Link.Break

Link.CaseCode

Link.Create

Link.Export

Link.Expunge

Link.Create

Syntax

```
]LINK.Create <ns> <dir> [-source={ns|dir|auto}] [-watch={none|ns|dir|both}] [-case: message + {options} []SE.Link.Create (namespace directory)
```

Arguments

- `namespace` is either a reference to, or a simple character containing the name of a namespace.  
In the user command `<ns>` is simply the name of the namespace. If a reference is used, it must refer to a namespace which has a display form which has name class 9 and can be used to locate the namespace (as opposed to an "anonymous" space with a name containing `[namespace]` or similar segments).
- `directory` is a simple character vector containing the path to a file system directory without any trailing slash or backslash.  
In the user command, `<dir>` is the path to the file system directory.

Result

- `message` is a simple character vector describing the established link, along with possible failures

Table of contents

Syntax

Arguments

Result

Common options

source

watch

arrays

sysVars

forceExtensions

forceFileNames

Advanced Options

flatten

caseCode

beforeWrite

beforeRead

getFilename

codeExtensions

customExtensions

typeExtensions

fastLoad



Index - Link User Guide

← → ↺ dyalog.github.io/link/4.0/ 🔍 📄 ⭐ 🗨 ⚙ 📖 👤 ⋮

Apps Link JSWC APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Ferie 2022

DYALOG Link User Guide 🔍 Search Dyalog/Link v3.0.19 ⭐ 15 🗨 7

Link User Guide

Overview

Index

Technical Details and Limitations

Workspaces

History of source files as text in Dyalog

Install and Upgrade

Installation

Version 4.0 Release Notes

Working with Link

Basic Usage

Configuration Files

Setting Up Your Application

Converting an Existing Workspace to use Link

API & Command Reference

API Overview

Link.Add

Link.Break

Link.CaseCode

Link.Create

Link.Configure

Link.Export

Link.Expunge

Link.Fix

Link.GetFileName

Link.GetItemName

Link.Import

Note

Link version 4.0 and its documentation are under development! You can track our progress [on GitHub](#).

Audience

It is assumed the reader is a user of Dyalog APL using Dyalog APL version 19.0 or later. If you are currently managing text source using SALT or Link versions 1 or 2 and considering moving to Link 4.0, you might want to review documentation which describes differences between early versions of SALT and Link - these can be found in the [Link 3.0 documentation](#).

Link 4.0 is designed to be upwards compatible with 3.0. If you are migrating from Link 3.0 to 4.0, you may want to begin with a review of the [new features of Link 4.0](#).

Introduction

Link

 allows you to use Unicode text files to store APL source code, rather than "traditional" binary workspaces. The benefits of using Link and text files include:

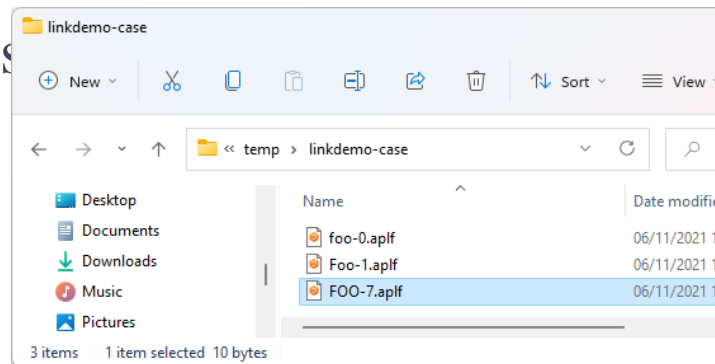
- It is easy to use source code management (SCM) tools like Git or Subversion to manage your code. Although an SCM is not a requirement for Link, Dyalog *highly* recommends using Git or similar systems to manage source code that Link will load into your APL session.
- Changes to your code are **immediately** written to file: there is no need to remember to save your work. The assumption is that you will make the record permanent with a *commit* to your source code management system, when the time is right.
- Unlike binary workspaces, text source can usually be shared between different versions of APL - or even with human readers or writers who don't have APL installed at all.

# Highlights of Link 3.0 (Dyalog 18.2)

- ◆ LOAD=
- ◆ Handles "Resync" of saved workspaces
- ◆ Supports names that differ only in case using "case coding"

# Highlights of Link 3.0 (Dyalog 18.2)

- LOAD=
- Handles "Resync" of saved workspaces
- Supports names that differ only in case using "case coding"



# Link v4.0 (Dyalog v19.0)

## Link v4.0 Highlights

- ◆ **Configuration Files (incl "Global" config)**
- ◆ Link a single Class or Namespace file
  - ◆ As opposed to a folder
- ◆ Create/Export/Import default to current namespace if none supplied
- ◆ Support for character vectors, matrices and vec-of-vecs in simple text files
- ◆ Link now being used by APL interpreter to load user code at startup
- ◆ Array Notation processing done in C

## Link User Guide

### Overview

[Index](#)[Technical Details and Limitations](#)[Workspaces](#)[History of source files as text in Dyalog](#)

### Install and Upgrade

[Installation](#)[Version 4.0 Release Notes](#)

### Working with Link

[Basic Usage](#)[Configuration Files](#)[Setting Up Your Application](#)[Converting an Existing Workspace to use Link](#)

### API & Command Reference

[API Overview](#)[Link.Add](#)[Link.Break](#)[Link.CaseCode](#)

## User Configuration Files

You can have a user configuration file called `.linkconfig`, which records preferences that apply to all links, for example link creation options like `-watch=` or the `notify` setting. Under Microsoft Windows, this file is stored in the `Documents` folder, on other systems in the user's home directory.

A simple example of a user configuration file would be:

```
{
  LinkVersion: { ID: "4.0.11" },
  Debug: {
    notify: 1,
  },
  Settings: {
    watch: "ns",
  },
}
```

With the above user configuration file contents, Link will display notifications when processing changes to the source (`notify:1`), and default to only propagating changes from the namespace to source files (`watch:ns`). If a call to `Link.Create` includes an explicit setting of the `watch` option, that will override the default.

Note that a configuration contains a `LinkVersion` section, which identifies the version of Link that wrote the file.

## Table of contents

[Introduction](#)[User Configuration Files](#)[Directory Configuration Files](#)[Stop and Trace flags](#)[Link.Configure](#)[The Configuration File Format](#)[The ignoreconfig switch](#)

## Link User Guide

## Overview

Index

Technical Details and Limitations

Workspaces

History of source files as text in  
Dyalog

## Install and Upgrade

Installation

Version 4.0 Release Notes

## Working with Link

Basic Usage

## Configuration Files

Setting Up Your Application

Converting an Existing  
Workspace to use Link

## API &amp; Command Reference

API Overview

Link.Add

Link.Break

Link.CaseCode

## Directory Configuration Files

Each linked directory may contain a `.linkconfig` file containing defaults that will apply when a link is created to that directory, or if the directory is imported. When [Link.Create](#) or [Link.Export](#) create a directory and create files in it, any non-default switch settings provided to the API function will be recorded in a configuration file within the directory. This means that you no longer need to remember the options used to re-create the original link when continuing work, or importing the link into a runtime environment.

If you already have a Link folder which was created by an earlier version of Link, you can add a `.linkconfig` file using [Link.Configure](#). For example:

```
]Link.Create linkdemo /tmp/linkdemo  
]Link.Configure linkdemo flatten:1  
Was flatten:
```

The result documents that there was no previous setting for the option. A file called `/tmp/linkdemo/.linkconfig` is created, with the following contents:

```
{  
  LinkVersion: { ID: "4.0.11"},  
  Settings: {  
    flatten: 1,  
  },  
}
```

## Table of contents

Introduction

User Configuration Files

[Directory Configuration Files](#)

Stop and Trace flags

Link.Configure

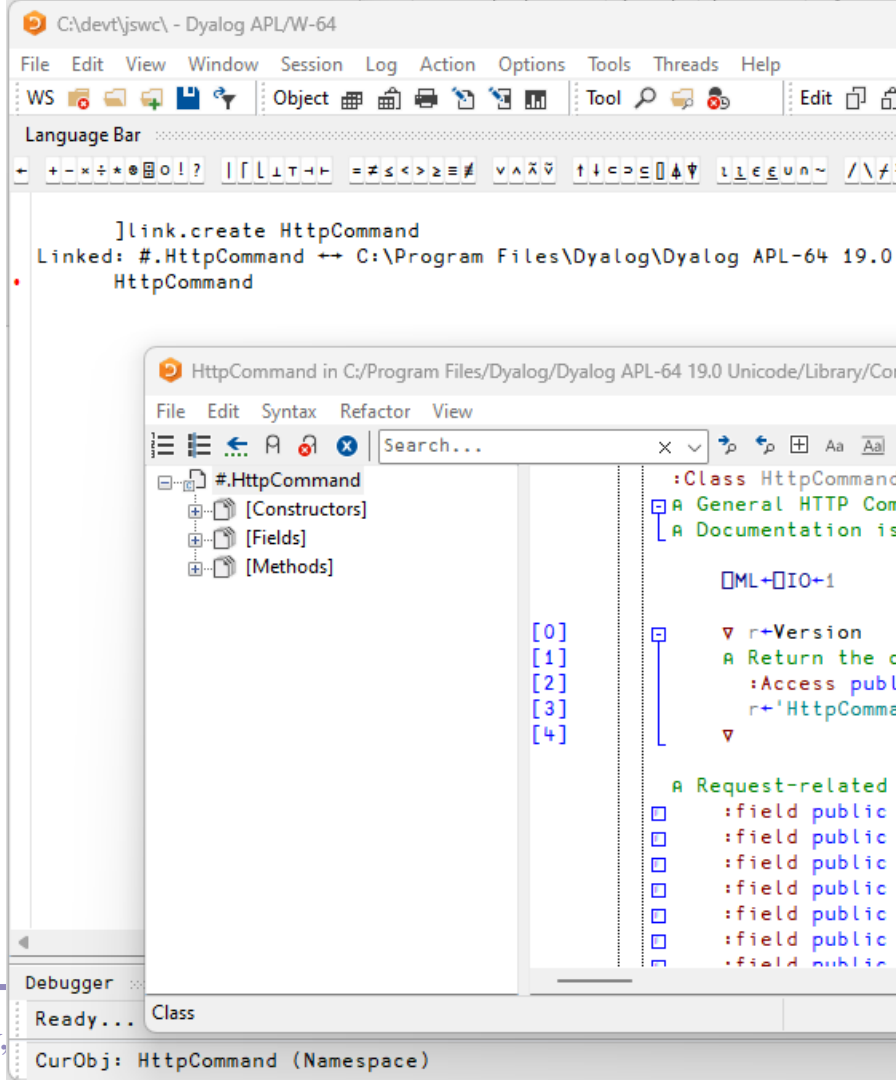
The Configuration File Format

The ignoreconfig switch

# Link v4.0

## Link v4.0 Highlights

- Configuration Files (incl "Global" config)
- Link a single Class or Namespace file
  - As opposed to a folder
- Create/Export/Import default to current namespace if none supplied
- Support for character vectors, matrices and vec-of-vecs in simple text files
- Link now being used by APL interpreter to load user code at startup
- Array Notation processing done in C



# Link v4.0 (Dyalog v19.0)

## Link v4.0 Highlights

- ◆ Configuration Files (incl "Global" config)
- ◆ Link a single Class or Namespace file
  - ◆ As opposed to a folder
- ◆ Create/Export/Import default to current namespace if none supplied
- ◆ **Support for character vectors, matrices and vec-of-vecs in simple text files**
- ◆ Link now being used by APL interpreter to load user code at startup
- ◆ Array Notation processing done in C



# Character Arrays in "Simple" Files

The screenshot shows the Dyalog APL/W-64 environment. The main workspace displays a variable named 'Supported' as a table with 8 columns: Type, Items, Size, Text, Posn, SelItems, Event, and Visible. Below the table, the cursor is positioned at the first column. The bottom status bar indicates 'CurObj: Supported (Variable)'.

Two overlapping windows show the APL code for the 'Supported' variable. The 'Supported.apla' window displays the code in a structured format with single quotes around each element name. The 'Supported.vec.apla' window displays the same code in a more compact, list-like format. An orange arrow points from the 'Supported.apla' window to the 'Supported.vec.apla' window, indicating a transformation or comparison of the two representations.

**Supported.apla**

```
(  
  'Type'  
  'Items'  
  'Size'  
  'Text'  
  'Posn'  
  'SelItems'  
  'Event'  
  'Visible'  
)
```

**Supported.vec.apla**

```
Type  
Items  
Size  
Text  
Posn  
SelItems  
Event  
Visible
```

# Link v4.0 (Dyalog v19.0)

## Link v4.0 Highlights

- ◆ Configuration Files (incl "Global" config)
- ◆ Link a single Class or Namespace file
  - ◆ As opposed to a folder
- ◆ Create/Export/Import default to current namespace if none supplied
- ◆ Support for character vectors, matrices and vec-of-vecs in simple text files
- ◆ **Link now being used by APL interpreter to load user code at start up**
- ◆ **Array Notation processing done in C**

# Link Road Map

## Link 5 & 6

- Crawler which will periodically compare workspace to source folders
  - Reduces / avoids need for .NET "File System Watcher"
  - Postponed from v3.0 to v4.0, and now from v4.0 to v5.0
- "Create a proper API"
  - Probably v6.0

Link 5.0 Milestone

github.com/Dyalog/link/milestone/4

AppsLinkJSWCAPLFlying & SailingCarDyalogCloudSBOTravelLinuxSportProductivityFerie 2022

Dyalog / link

Type to search

+>⌕🔗📧🔍

<> CodeIssues66Pull requests1DiscussionsActionsProjectsWikiSecurityInsightsSettings

Labels

Milestones

Edit milestone

New issue

Link 5.0

Due by March 31, 20240% complete

A Spring 2024 release.

16 Open0 Closed

Create a more regular API with clear return codes and error messagesenhancement

#125 opened on May 14, 2020 by nicolas-dyalog

Add a file crawler in background as backup for FileSystemWatcherenhancement

#88 opened on Jul 30, 2019 by mkromberg

Add a namespace crawler background process to check for APL names that have been changed by APL primitivesenhancement

#51 opened on Apr 30, 2019 by stampes

Allow linking source files for functions, operators and arraysenhancement

#559 opened on Jul 7 by mkromberg

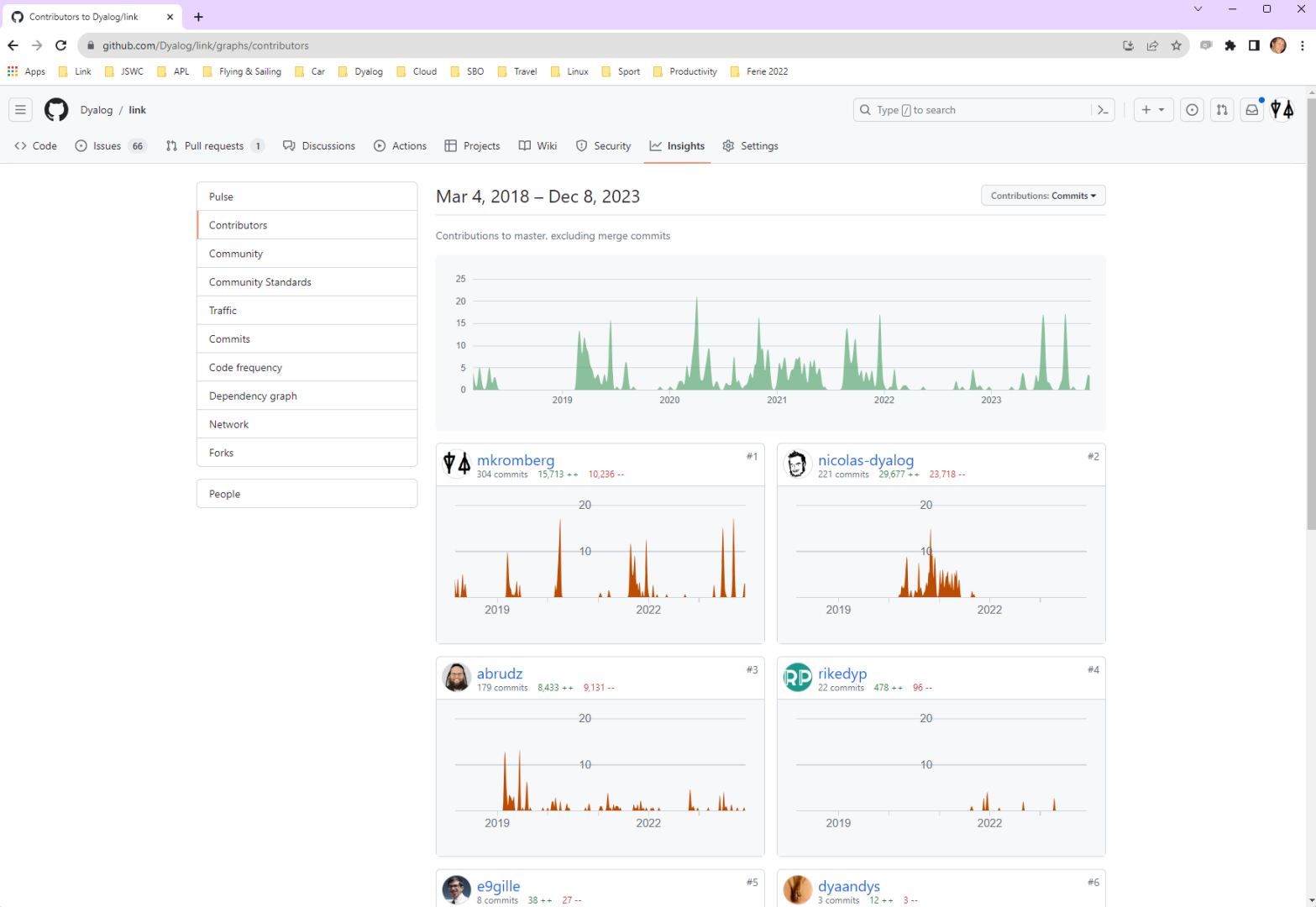
Clean up the QAenhancement

#545 opened on Jun 29 by mkromberg

Add a -browse switch for Create and Importenhancement

#573 opened on Sep 3 by bpbecker

Test Cleanup: allow assert to take a boolean argumentenhancement



# Recommended Viewing

Welcome to Dyalog Videos Page x +

dyalog.tv/Dyalog23/?v=FIUH-306Gns

Relaunch to update

Apps Link JSWC APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity

**DYALOC**  
Elsinore 2023

Dyalog '23 APL Seeds '23 Dyalog '22 Webinar APL Seeds '22 Dyalog '21 APL Seeds '21 Dyalog '20 Dyalog '19 Dyalog '18 Dyalog '17

Dyalog '16 Dyalog '15 Dyalog '14 Dyalog '13 Dyalog '12 Dyalog '11 APL Berlin 2010 Dyalog '09 Dyalog '08

Using Packages // Morten Kromberg // Dyalog '23

Watch Later Share

**DYALOC**  
Elsinore 2023

**Using Packages**  
Morten Kromberg

**Using Packages**  
Morten Kromberg

**Dyalog Version 20.0 - Part 2**  
John Daintree

**Statistical Libraries for Dyalog**  
Josh David

**Vega Charts with Dyalog**  
Rich Park

**Research and Education**

quAPL - A Quantum Computing Library in APL

MORE VIDEOS

# Conclusion

- Link enables the use of many valuable 3rd party tools with APL source
- These tools can give you MUCH better control of your source code
- ...all without losing any of our Inalienable Rights

