DYALOG

2021

Scripting in Dyalog 18.2

John Daintree

Topics

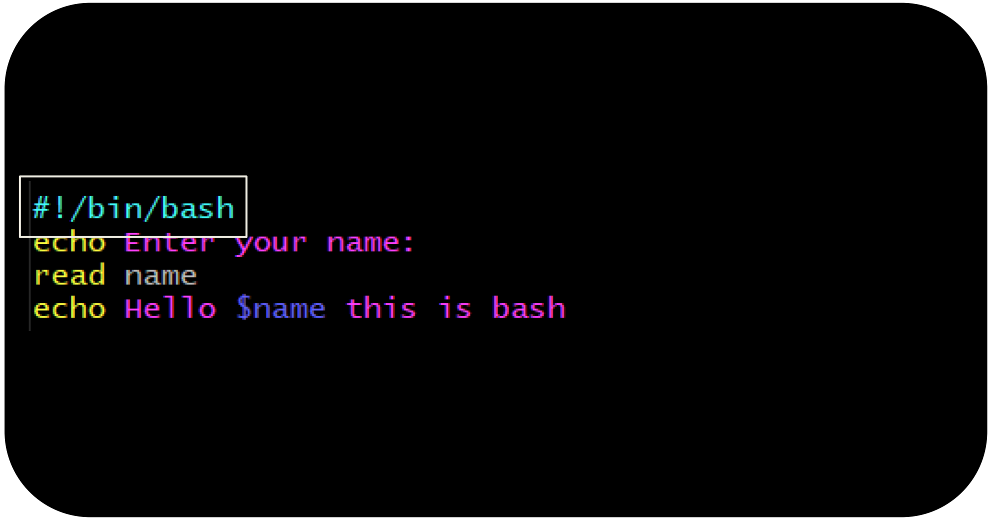
- What we mean by ”Scripting”
- What has changed since this time last year
- What may change by this time next year

What we mean by ”Scripting”

- #! (hashbang) scripting
  - shebang (or shabang) scripting
  - shell scripting
- Text files that can be executed from an interactive terminal
  - Using standard I/O (stdin/stdout/stderr and redirections)
- Why would we want to?
  - It allows APL to be used alongside other script capable languages
  - Use APL code in a pipeline

What we mean by "Scripting"

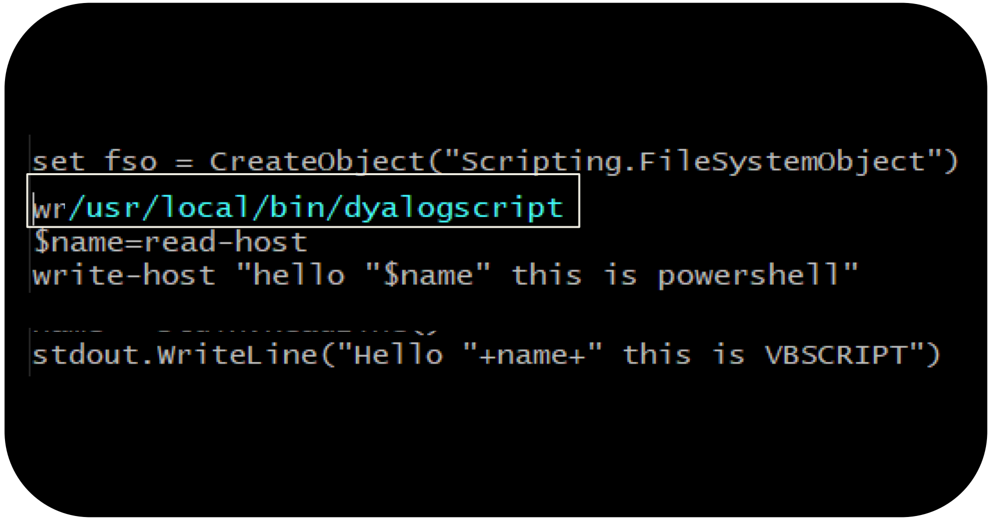
- .bat
- .bash (or .ksh etc.)



```
#!/bin/bash
echo Enter your name:
read name
echo Hello $name this is bash
```

What we mean by "Scripting"

- .bat
- .bash (or .ksh etc.)
- .ps1
- .vbs
- .apl



```
set fso = CreateObject("Scripting.FileSystemObject")
wr/usr/local/bin/dyalogscript
$name=read-host
write-host "hello "$name" this is powershell"

stdout.WriteLine("Hello "+name+" this is VBSCRIPT")
```

- Made possible by the "extended multiline session input"

First, a few examples

What has changed since this time last year

- We've changed the file extension from .dyalogscript to .apl
  - This is (sort of) only relevant on Windows
- Numerous small tweaks and fixes
- (and we will add) more samples

What has changed since this time last year

Default output:

```
#!/usr/local/bin/dyalogscript
⎕←'Enter your name: ' ⋄ name←⍞
  'hello ',name,' this is APL'
⎕←'hello ',name,' this is APL'
```

This will NOT be output from the script

Assignment is required

What has changed since this time last year

Configuration:

Settings (MAXWS etc.) are no longer retrieved from the environment or the registry

```
#!/usr/local/bin/dyalogscript
size←{
α←'b KbMbGbTb'
ω<1024:(⍕ω),2↑α
(2⌽α)∇ ω÷1024
}
⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',size ⎕WA
```

What has changed since this time last year

Configuration:

Settings (MAXWS etc.) are no longer retrieved from the environment or the registry

```
#!/usr/local/bin/dyalogscript
size←{
α←'b KbMbGbTb'
ω<1024:(⍕ω),2↑α
(2⌽α)∇ ω÷1024
}
⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',size ⎕WA
```

maxws.apl

```
{ settings:
   {
    MAXWS:"400Mb"
   }
}
```

maxws.dcfg

What has changed since this time last year

Exit on Error:

Script will exit with non-zero exit code

```
#!/usr/local/bin/dyalogscript
⎕←'Hello'
⎕←÷0
⎕←'World'
```

This generates an error

This will not be executed

Script will exit with non zero exit code

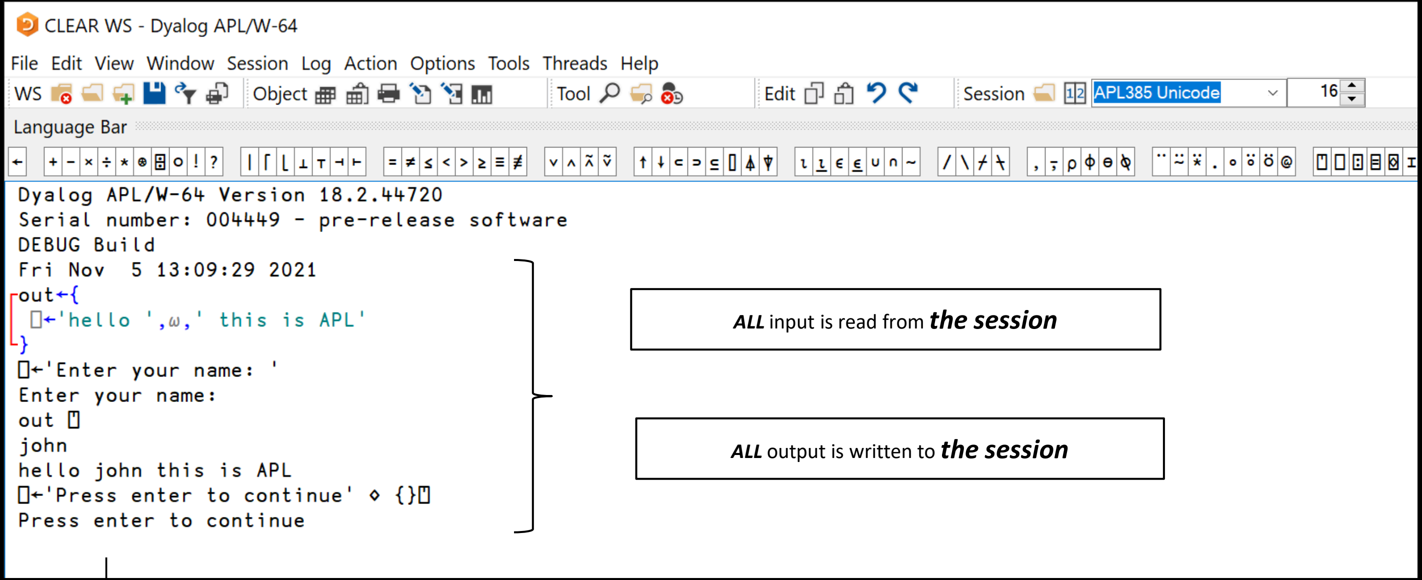
Some words about debugging with the ODE or RIDE

It’s tricky

We’re working on it

There’s a whole lot of I/O going on

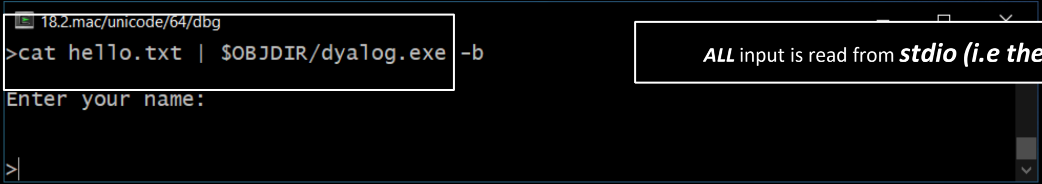
```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⎕←'Enter your name: '
out ⍞
⎕←'Press enter to continue'
{}⍞
```


CLEAR WS - Dyalog APL/W-64
File Edit View Window Session Log Action Options Tools Threads Help
WS Object Tool Edit Session APL385 Unicode 16
Language Bar

```
Dyalog APL/W-64 Version 18.2.44720
Serial number: 004449 - pre-release software
DEBUG Build
Fri Nov  5 13:09:29 2021
out←{
 ⎕←'hello ',⍵,' this is APL'
}
⎕←'Enter your name: '
Enter your name:
out ⍞
john
hello john this is APL
⍞←'Press enter to continue' ⋄ {}⍞
Press enter to continue
```

ALL input is read from ***the session***

ALL output is written to ***the session***



```
>cat hello.txt | $OBJDIR/dyalog.exe -b
Enter your name:

>
```

ALL input is read from ***stdio (i.e the file)***

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⍞←'Press enter to continue'
{}⍞
```

Read from ***script file*** (and Ignored)

Read from ***script file*** and executed in the session

Read from ***script file*** and executed in the session

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',ω,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⍞←'Press enter to continue'
{}⍞
```

Written to ***user output***, e.g. terminal

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⎕←'Enter your name: '
out ⍞
⎕←'Press enter to continue'
{}⍞
```

Read from ***script file*** and “executed” in the session

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⍞←'Press enter to continue'
{}⍞
```

Read from ***user input***, e.g. terminal

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⍞←'Press enter to continue'
{}⍞
```

Read from ***script file*** and “executed” in the session

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',⍵,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⍞←'Press enter to continue'
{}⍞
```

Written to ***user output***, e.g. terminal

```
#!/usr/local/bin/dyalogscript
out←{
⎕←'hello ',ω,' this is APL'
}

⍞←'Enter your name: '
out ⍞
⎕←'Press enter to continue'
{}⍞
```

Read from ***user input***, e.g. terminal

There’s a whole lot of I/O going on

Traditional development

There's a whole lot of I/O going on

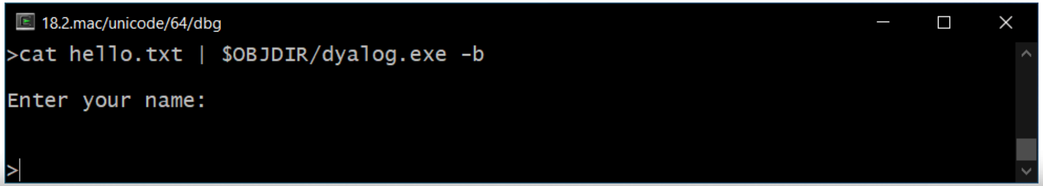
Traditional development

Code (session)

User I/O via ⎕ (session)

User I/O via ⍞ (session)

All I/O via the session



```
18.2.mac/unicode/64/dbg
>cat hello.txt | $OBJDIR/dyalog.exe -b

Enter your name:


>
```

There's a whole lot of I/O going on

Traditional development

Code (session)

User I/O via ⎕ (session)

User I/O via ⍞ (session)

All I/O via the session

#! runtime

There’s a whole lot of I/O going on

Traditional development

Code (session)

User I/O via ⎕ (session)

User I/O via ⍞ (session)

All I/O via the session

#! runtime

Code (the #! file)

User I/O via ⎕ (stdin/stdout)

User I/O via ⍞ (stdin/stderr)

They may all be different

There’s a whole lot of I/O going on

Traditional development	***#! runtime***	***#! debugging***
Code (session)	Code (the #! file)	
User I/O via ⎕ (session)	User I/O via ⎕ (stdin/stdout)	
User I/O via ⍞ (session)	User I/O via ⍞ (stdin/stderr)	
All I/O via the session	They may all be different	

There's a whole lot of I/O going on

Traditional development

Code (session)

User I/O via ⎕ (session)

User I/O via ⍞ (session)

All I/O via the session

#! runtime

Code (the #! file)

User I/O via ⎕ (stdin/stdout)

User I/O via ⍞ (stdin/stderr)

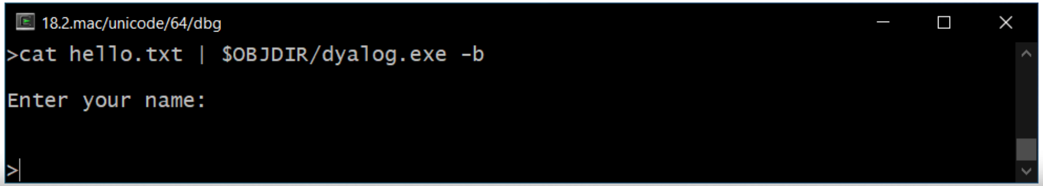
They may all be different

#! debugging

Pick from the first two columns

This choice will change as the debugger is attached or detached

And during the debug session


```
18.2.mac/unicode/64/dbg
>cat hello.txt | $OBJDIR/dyalog.exe -b

Enter your name:


>|
```

It’s tricky

We’re working on it

So in the meantime

If it's good enough for bash…

set –x

-x = "minus x" = MX = 1010

1010⌶

```
#!/bin/bash


echo Enter your name:
read name
echo Hello $name this is bash
```

```
Enter your name:
johnd
Hello johnd this is bash
```

```
#!/bin/bash
set -x

echo Enter your name:
read name
echo Hello $name this is bash
```

```
+ echo Enter your name:
Enter your name:
+ read name
johnd
+ echo Hello johnd this is bash
Hello johnd this is bash
```

```
@echo off
echo Enter your name:
set /p NAME=
echo Hello %NAME% this is CMD
```

```
Enter your name:
johnd
Hello johnd this is CMD
```

```
echo Enter your name:
set /p NAME=
echo Hello %NAME% this is CMD
```

```
M:\scripts>echo Enter your name:
Enter your name:
M:\scripts>set /p NAME=
M:\scripts>echo Hello john this is CMD
Hello john this is CMD
```

```
#!/bin/bash
set -x

echo Enter your name:
read name
echo Hello $name this is bash
```

```
+ echo Enter your name:
Enter your name:
+ read name
johnd
+ echo Hello johnd this is bash
Hello johnd this is bash
```

0

```
#!/usr/local/bin/dyalogscript
(1010⌶)  ⍝ c.f. set -x (¯x->MX->1010)

∇r←tsize w
a←'b KbMbGbTb'
:While w>1024
a←2⌽a ⋄ w←w÷1024
:End
r←(⍕w),2↑a
∇

⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',tsize ⎕WA
```

```
MAXWS is 256M
⎕WA is 255.9692993Mb
```

1

```
#!/usr/local/bin/dyalogscript
(1010⌶) ⍝ c.f. set -x (¯x->MX->1010)

∇r←tsize w
a←'b KbMbGbTb'
:While w>1024
a←2⌽a ⋄ w←w÷1024
:End
r←(⍕w),2↑a
∇

⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',tsize ⎕WA
```

```
+
+∇r←tsize w
+a←'b KbMbGbTb'
+:While w>1024
+          a←2⌽a ⋄ w←w÷1024
+:End
+r←(⍕w),2↑a
+∇
+
+⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS
MAXWS is 256M
+⎕←'⎕WA is ',tsize ⎕WA
⎕WA is 255.9692993Mb
+
#!/usr/local/bin/dyalogscript
'         '(1010⌶)1 ⍝ c.f. set -x (¯x->MX->1010)

∇r←tsize w
a←'b KbMbGbTb'
:While w>1024
a←2⌽a ⋄ w←w÷1024
:End
r←(⍕w),2↑a
∇

⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',tsize ⎕WA
```

```
      ∇r←tsize w
      a←'b KbMbGbTb'
      :While w>1024
               a←2⌽a ⋄ w←w÷1024
      :End
      r←(⍕w),2↑a
      ∇

      ⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' '
MAXWS is 256M
      ⎕←'⎕WA is ',tsize ⎕WA
⎕WA is 255.9692993Mb
```

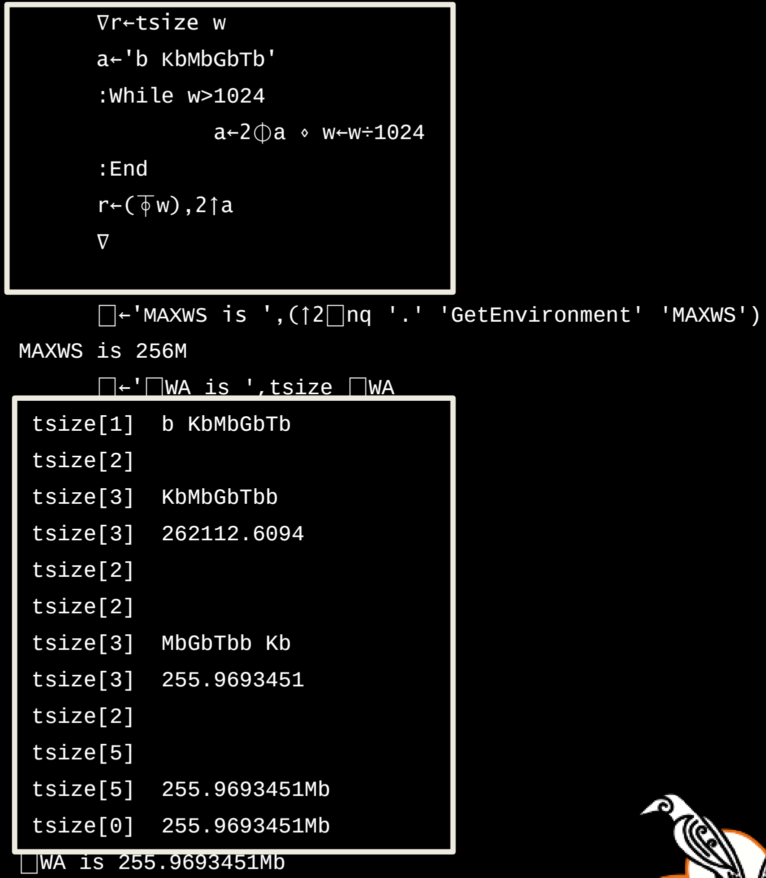
2

```
#!/usr/local/bin/dyalogscript
'      '(1010⌶)  ⍝ c.f. set -x (¯x->MX->1010)

∇r←tsize w
a←'b KbMbGbTb'
:While w>1024
a←2⌽a ⋄ w←w÷1024
:End
r←(⍕w),2↑a
∇

⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',tsize ⎕WA
```

```
MAXWS is 256M
tsize[1]  b KbMbGbTb
tsize[2]
tsize[3]  KbMbGbTbb
tsize[3]  262112.5625
tsize[2]
tsize[2]
tsize[3]  MbGbTbb Kb
tsize[3]  255.9692993
tsize[2]
tsize[5]
tsize[5]  255.9692993Mb
tsize[0]  255.9692993Mb
⎕WA is 255.9692993Mb
```

3

```
#!/usr/local/bi  yalogscript
'        ‘(1010⌶)  ⍝ c.f. set -x (¯x->MX->1010)

∇r←tsize w
a←'b KbMbGbTb'
:While w>1024
a←2⌽a ⋄ w←w÷1024
:End
r←(⍕w),2↑a
∇

⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
⎕←'⎕WA is ',tsize ⎕WA
```



```
      ∇r←tsize w
      a←'b KbMbGbTb'
      :While w>1024
               a←2⌽a ⋄ w←w÷1024
      :End
      r←(⍕w),2↑a
      ∇
      ⎕←'MAXWS is ',(↑2⎕nq '.' 'GetEnvironment' 'MAXWS')
MAXWS is 256M
      ⎕←'⎕WA is ',tsize ⎕WA
 tsize[1]  b KbMbGbTb
 tsize[2]
 tsize[3]  KbMbGbTbb
 tsize[3]  262112.6094
 tsize[2]
 tsize[2]
 tsize[3]  MbGbTbb Kb
 tsize[3]  255.9693451
 tsize[2]
 tsize[5]
 tsize[5]  255.9693451Mb
 tsize[0]  255.9693451Mb
⎕WA is 255.9693451Mb
```

While we're comparing with bash
some things that we don't do

```
#!/bin/bash
set -e

echo Enter your
rm this_does_not
echo this will
```

```
#!/usr/local/bin/dyalogscript
⎕←'Hello'
⎕←÷0
⎕←'World'
```

This generates an error

This will not be executed

Script will exit with non zero exit code

```
#!/bin/bash
set -e

echo Enter your name:
rm this_does_not_exist
echo this will not happen
```

This generates an error

This will not be executed

Script will exit with non zero exit code

Exit on Error:

Script will exit with non-zero exit code

```
#!/usr/local/bin/dyalogscript

⎕←'Hello'
⎕←÷0
⎕←'World'
```

This generates an error

This will not be executed

Script will exit with non zero exit code

Exit on Error:

Script will exit with non-zero exit code

4

```
#!/usr/local/bin/dyalogscript
(1010⌶)[illegible]⍝ c.f. set -e
⎕←'Hello'
⎕←÷0
⎕←'World'
```

This generates an error

This WOULD be executed

Script WOULD exit with a ZERO exit code

Calling scripts from scripts

hello_dup.bash

```
#!/bin/bash

hello.bash
hello.bash
```

starts a new process

starts a new process

```
Enter your name:
johnd
Hello johnd this is bash
Enter your name:
johnd
Hello johnd this is bash
```

hello_dup.bash

```
#!/bin/bash

. hello.bash
. hello.bash
```

runs in the same process

runs in the same process

```
Enter your name:
johnd
Hello johnd this is bash
Enter your name:
johnd
Hello johnd this is bash
```

hello_dup.apl

```
#!/usr/local/bin/dyalogscript

)sh hello.apl
)sh hello.apl
```

hello_dup.apl

```
#!/usr/local/bin/dyalogscript

⎕sh 'hello.apl'
⎕sh 'hello.apl'
```

starts a new process

starts a new process

```
Enter your name:
johnd
Hello johnd this is bash
Enter your name:
johnd
Hello johnd this is bash
```

But

There's a whole lot of I/O going on

hello_dup.apl

```
#!/usr/local/bin/dyalogscript
```

```
⎕sh 'hello.apl'
⎕sh 'hello.apl’
```

starts a new process

starts a new process

```
Enter your name:
jo~~h~~nd
Hello johnd this is bash
Enter your name:
jo~~h~~nd
Hello johnd this is bash
```

Might be fixable in ⎕SH

hello_dup.bash

```
#!/bin/bash

. hello.bash
. hello.bash
```

runs in the same process

runs in the same process

```
Enter your name:
johnd
Hello johnd this is bash
Enter your name:
johnd
Hello johnd this is bash
```

~~Might be fixable in ⎕SH~~

But more likely a new mechanism will be required

What may change by this time next year

- Debugging improvements (RIDE etc.)
- Ability to have scripts calling scripts
  - Address the I/O issues
- Address the encoding issues on ⎕SH output
- More samples?
- Suggestions from users
  - Do we need `1010⌶4` to change the on error behavior?
  - What else?