Tool Talk

Free Tools for APL+Win Programmers

Rex Swain
Independent Consultant
8 South Street
Washington, CT 06793
Tel 860-868-0131
rex@rexswain.com
http://www.rexswain.com

April 2010

Revised June 2010

This is a consolidated version of presentations made at the APL2000 Conferences from 2003 through 2006, plus additional tools created since then.

Table of Contents

Instructions

- To download the tools, visit

  http://www.rexswain.com/tooltalk.html

  Also check there periodically for updates.

- Add UCMDSREX.SF to your user command files path:

```
]UFILE [1.5] UCMDSREX
```

  If you already have UCMDSREX.SF, just replace it with this one; all the previous tools are here too.

- Ask for help! Each tool provides a summary of arguments and options. For example,

```
]SEEK ?
```

- Many tools display forms using the APL font. After the first use of one of these tools, the defaults are written into your APLW.INI file:

```
[UCMDSREX]
FontName=APLFONT
FontSize=11
FontStyle=0
FontCharset=default
```

  If you wish, you may change these values.

- SEEK and SUPERSEEK both use the "APLNext" Unicode font, which may be downloaded from http://www.rexswain.com/tooltalk.html.

Acknowledgements

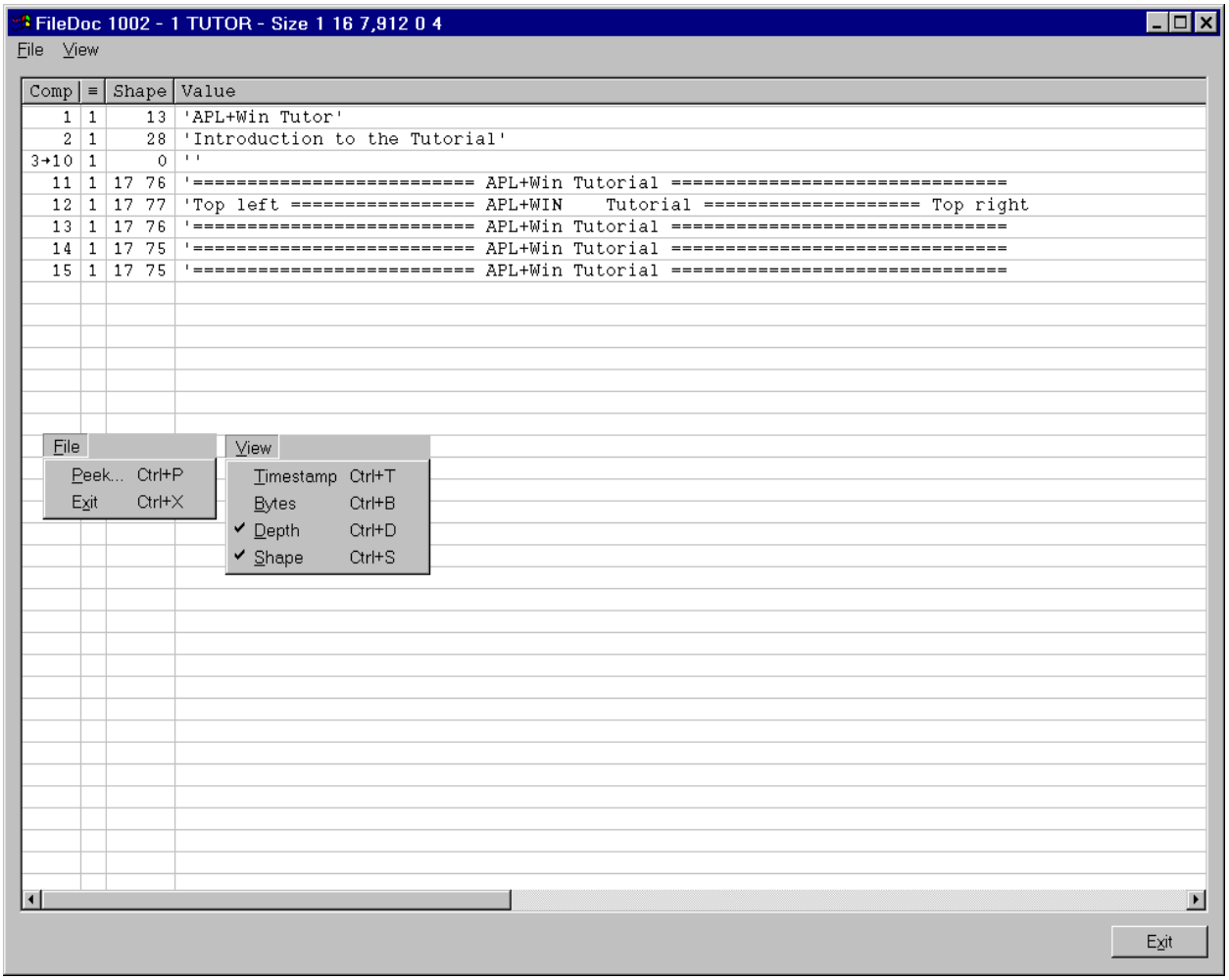
Many thanks to all those who have helped me to hurdle countless Windows obstacles -- notably Bob Smith, Patrick Parks, J. Merrill, Eric Lescasse, Pierre Gilbert, and Brent Hildebrand. I have enormous respect for these gurus who have explored the world outside of the APL, returned to tell the tale, and saved me countless hours of flailing around.

Thanks also to Gary Bergquist and Don Lagosz-Sinclair for their excellent PARSEVR function.

FILEDOC: Look at a Component File

Argument may be a file name or tie number

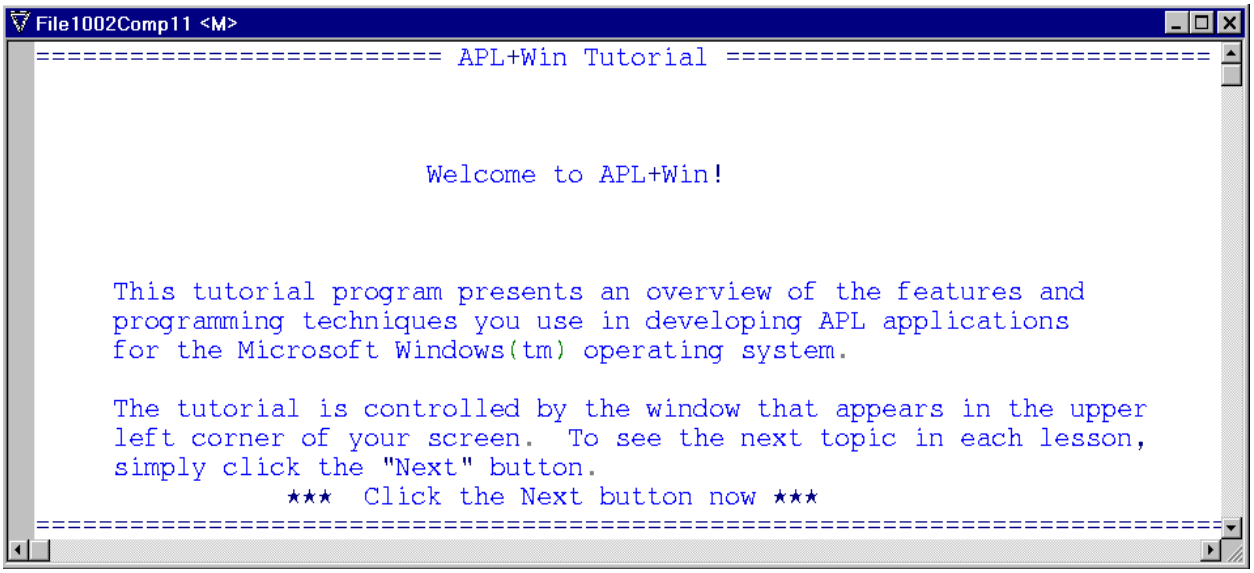
```
]filedoc 1 tutor
```



Note summarization of a series of identical values in components 3 through 10.

Use the View menu to show component timestamp and size.

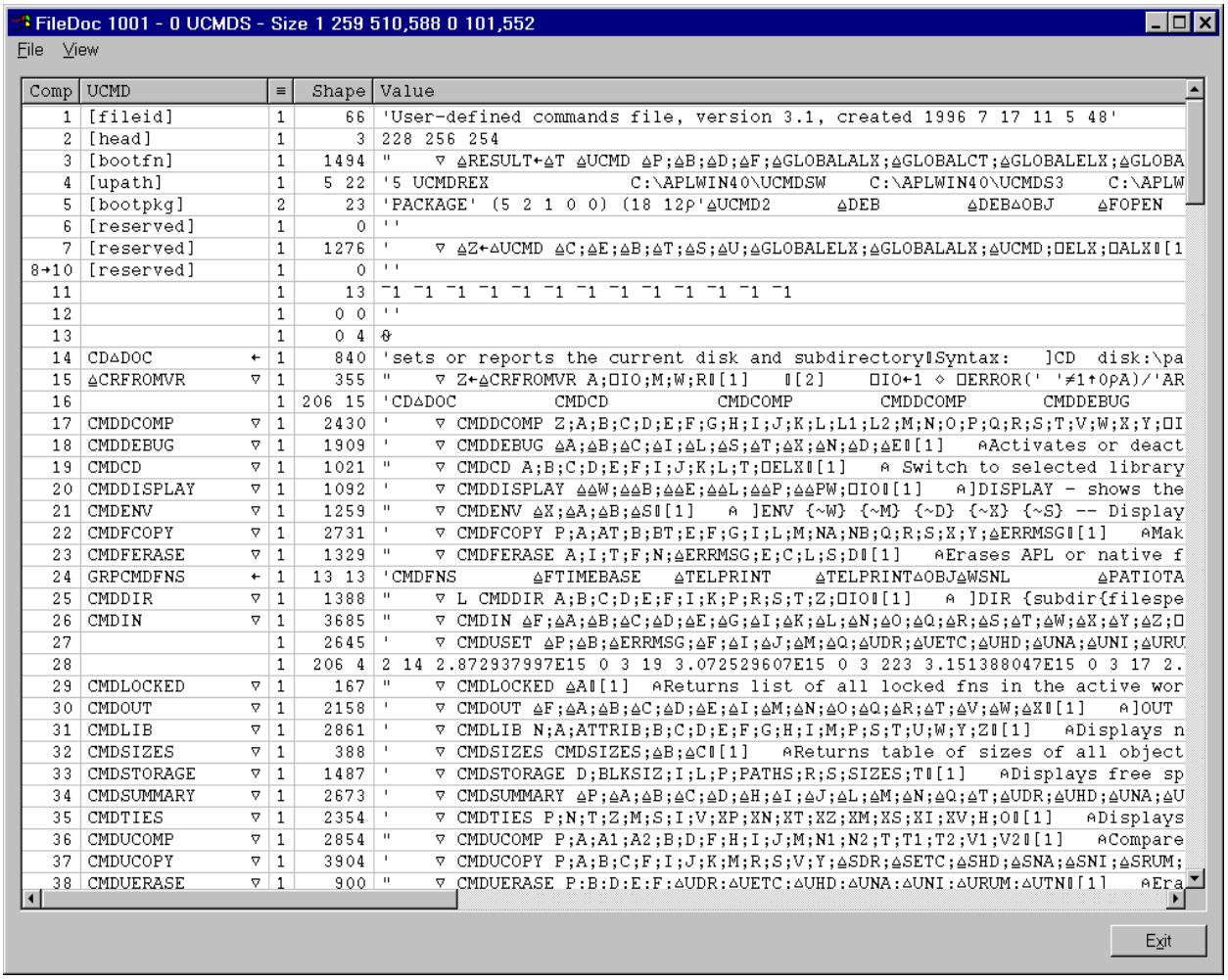
Use File|Peek or double-click to view a component in an APL edit window:



Any changes are *not* saved back into the file.

Note special handing of user-command files:

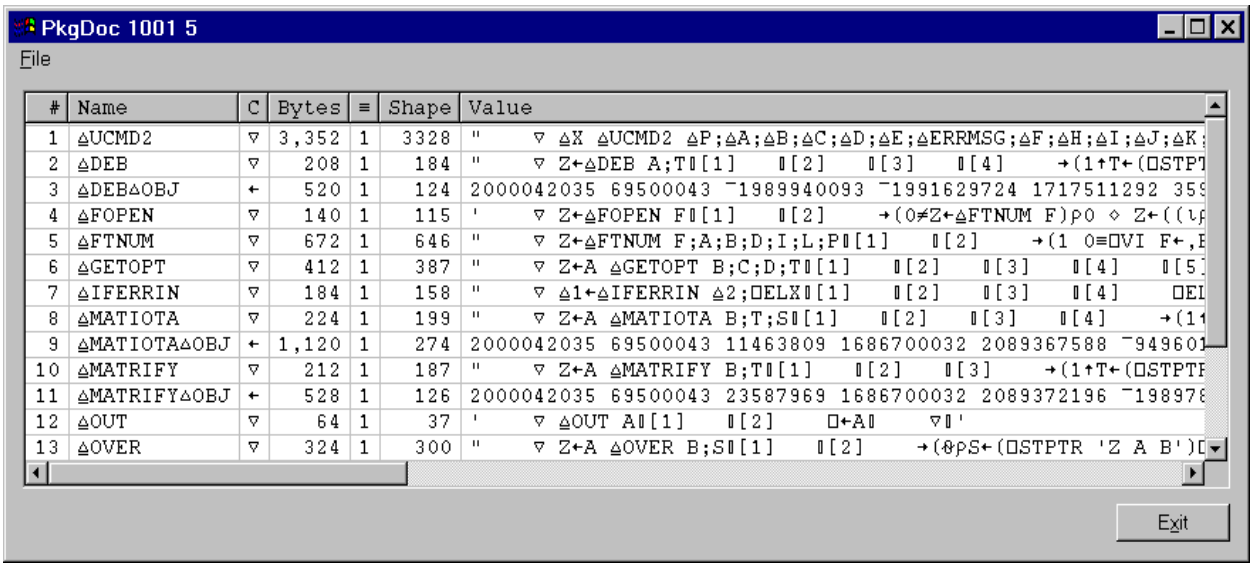
```
]filedoc 0 ucmds
```



Comp	UCMD	≡	Shape	Value
1	[fileid]	1	66	'User-defined commands file, version 3.1, created 1996 7 17 11 5 48'
2	[head]	1	3	228 256 254
3	[bootfn]	1	1494	" ∇ ⍙RESULT←⍙T ⍙UCMD ⍙P;⍙B;⍙D;⍙F;⍙GLOBALALX;⍙GLOBALCT;⍙GLOBALELX;⍙GLOBA
4	[upath]	1	5 22	'5 UCMDREX C:\APLWIN40\UCMDSW C:\APLWIN40\UCMDS3 C:\APLW
5	[bootpkg]	2	23	'PACKAGE' (5 2 1 0 0) (18 12⍴'⍙UCMD2 ⍙DEB ⍙DEB⍙OBJ ⍙FOPEN
6	[reserved]	1	0	''
7	[reserved]	1	1276	' ∇ ⍙Z←⍙UCMD ⍙C;⍙E;⍙B;⍙T;⍙S;⍙U;⍙GLOBALELX;⍙GLOBALALX;⍙UCMD;⎕ELX;⎕ALX⍮[1
8→10	[reserved]	1	0	''
11		1	13	¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1
12		1	0 0	''
13		1	0 4	⍬
14	CD⍙DOC ←	1	840	'sets or reports the current disk and subdirectory⍮Syntax:]CD disk:\pa
15	⍙CRFROMVR ∇	1	355	" ∇ Z←⍙CRFROMVR A;⎕IO;M;W;R⍮[1] ⍮[2] ⎕IO←1 ⋄ ⎕ERROR(' '≠1↑0⍴A)/'AR
16		1	206 15	'CD⍙DOC CMDCD CMDCOMP CMDDCOMP CMDDEBUG
17	CMDDCOMP ∇	1	2430	' ∇ CMDDCOMP Z;A;B;C;D;E;F;G;H;I;J;K;L;L1;L2;M;N;O;P;Q;R;S;T;V;W;X;Y;⎕I
18	CMDDEBUG ∇	1	1909	' ∇ CMDDEBUG ⍙A;⍙B;⍙C;⍙I;⍙L;⍙S;⍙T;⍙X;⍙N;⍙D;⍙E⍮[1] ⍝Activates or deact
19	CMDCD ∇	1	1021	" ∇ CMDCD A;B;C;D;E;F;I;J;K;L;T;⎕ELX⍮[1] ⍝ Switch to selected library
20	CMDDISPLAY ∇	1	1092	' ∇ CMDDISPLAY ⍙⍙W;⍙⍙B;⍙⍙E;⍙⍙L;⍙⍙P;⍙⍙PW;⎕IO⍮[1] ⍝]DISPLAY - shows the
21	CMDENV ∇	1	1259	" ∇ CMDENV ⍙X;⍙A;⍙B;⍙S⍮[1] ⍝]ENV {~W} {~M} {~D} {~X} {~S} -- Display
22	CMDFCOPY ∇	1	2731	' ∇ CMDFCOPY P;A;AT;B;BT;E;F;G;I;L;M;NA;NB;Q;R;S;X;Y;⍙ERRMSG⍮[1] ⍝Mak
23	CMDFERASE ∇	1	1329	" ∇ CMDFERASE A;I;T;F;N;⍙ERRMSG;E;C;L;S;D⍮[1] ⍝Erases APL or native f
24	GRPCMDFNS ←	1	13 13	'CMDFNS ⍙FTIMEBASE ⍙TELPRINT ⍙TELPRINT⍙OBJ⍙WSNL ⍙PATIOTA
25	CMDDIR ∇	1	1388	" ∇ L CMDDIR A;B;C;D;E;F;I;K;P;R;S;T;Z;⎕IO⍮[1] ⍝]DIR {subdir{filespe
26	CMDIN ∇	1	3685	" ∇ CMDIN ⍙F;⍙A;⍙B;⍙C;⍙D;⍙E;⍙G;⍙I;⍙K;⍙L;⍙N;⍙O;⍙Q;⍙R;⍙S;⍙T;⍙W;⍙X;⍙Y;⍙Z;⎕
27		1	2645	' ∇ CMDUSET ⍙P;⍙B;⍙ERRMSG;⍙F;⍙I;⍙J;⍙M;⍙Q;⍙UDR;⍙UETC;⍙UHD;⍙UNA;⍙UNI;⍙URU
28		1	206 4	2 14 2.872937997E15 0 3 19 3.072529607E15 0 3 223 3.151388047E15 0 3 17 2.
29	CMDLOCKED ∇	1	167	" ∇ CMDLOCKED ⍙A⍮[1] ⍝Returns list of all locked fns in the active wor
30	CMDOUT ∇	1	2158	' ∇ CMDOUT ⍙F;⍙A;⍙B;⍙C;⍙D;⍙E;⍙I;⍙M;⍙N;⍙O;⍙Q;⍙R;⍙T;⍙V;⍙W;⍙X⍮[1] ⍝]OUT
31	CMDLIB ∇	1	2861	' ∇ CMDLIB N;A;ATTRIB;B;C;D;E;F;G;H;I;M;P;S;T;U;W;Y;Z⍮[1] ⍝Displays n
32	CMDSIZES ∇	1	388	' ∇ CMDSIZES CMDSIZES;⍙B;⍙C⍮[1] ⍝Returns table of sizes of all object
33	CMDSTORAGE ∇	1	1487	' ∇ CMDSTORAGE D;BLKSIZ;I;L;P;PATHS;R;S;SIZES;T⍮[1] ⍝Displays free sp
34	CMDSUMMARY ∇	1	2673	' ∇ CMDSUMMARY ⍙P;⍙A;⍙B;⍙C;⍙D;⍙H;⍙I;⍙J;⍙L;⍙M;⍙N;⍙Q;⍙T;⍙UDR;⍙UHD;⍙UNA;⍙U
35	CMDTIES ∇	1	2354	' ∇ CMDTIES P;N;T;Z;M;S;I;V;XP;XN;XT;XZ;XM;XS;XI;XV;H;O⍮[1] ⍝Displays
36	CMDUCOMP ∇	1	2854	" ∇ CMDUCOMP P;A;A1;A2;B;D;F;H;I;J;M;N1;N2;T;T1;T2;V1;V2⍮[1] ⍝Compare
37	CMDUCOPY ∇	1	3904	' ∇ CMDUCOPY P;A;B;C;F;I;J;K;M;R;S;V;Y;⍙SDR;⍙SETC;⍙SHD;⍙SNA;⍙SNI;⍙SRUM;
38	CMDUERASE ∇	1	900	" ∇ CMDUERASE P:B:D:E:F:⍙UDR:⍙UETC:⍙UHD:⍙UNA:⍙UNI:⍙URUM:⍙UTN⍮[1] ⍝Era

Note special handing of PACKAGE components in user-command files:

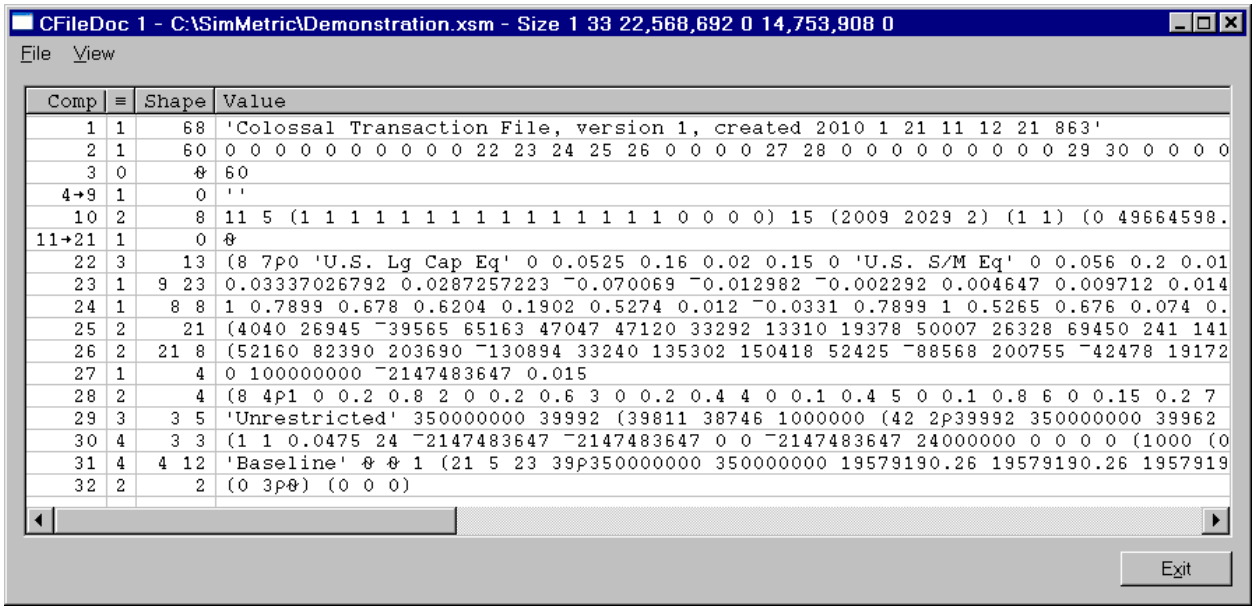
Double-click component 5:



CFILEDOC: Colossal Files

Same tool for colossal files.

```
]cfiledoc C:\SimMetric\Demonstration.xsm
```

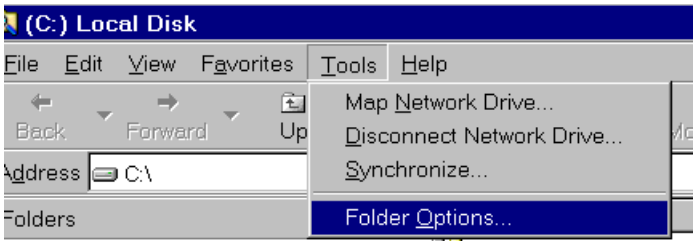


Associating]FileDoc with .SF Files

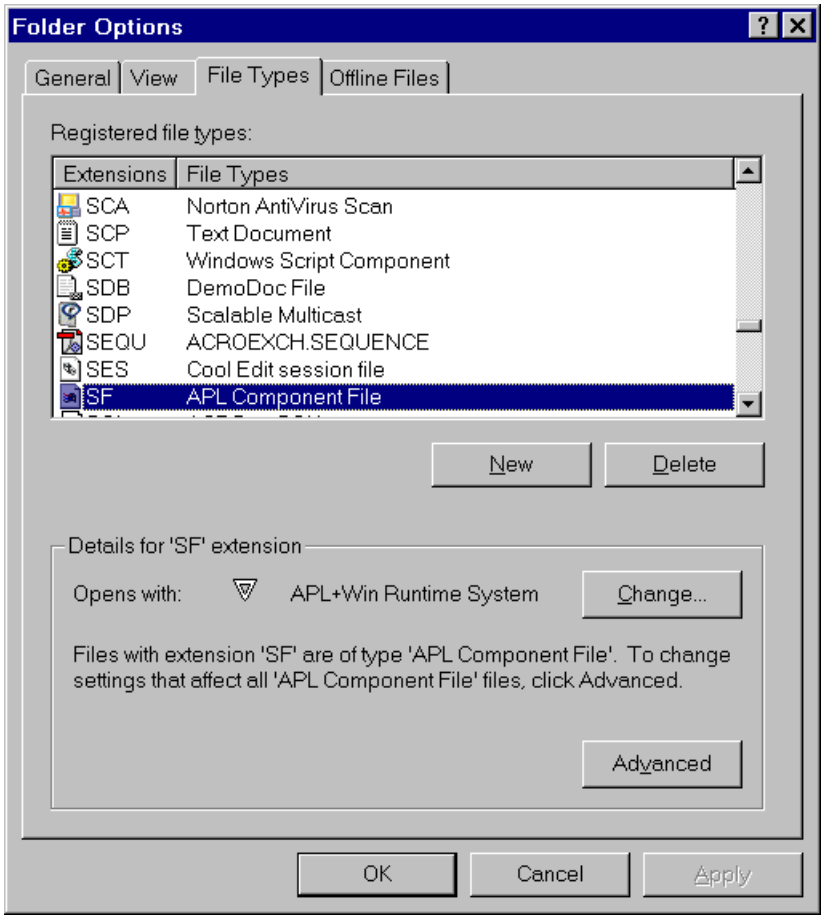
The following is for Windows XP, and assumes your APL+Win executables are in directory C:\APLWIN ; adjust accordingly.

`)LOAD` the workspace `FILEDOC` and then `)RSAVE` it into C:\APLWIN
Note! `)RSAVE` not `)SAVE`

Start Windows Explorer, Tools menu, Folder Options



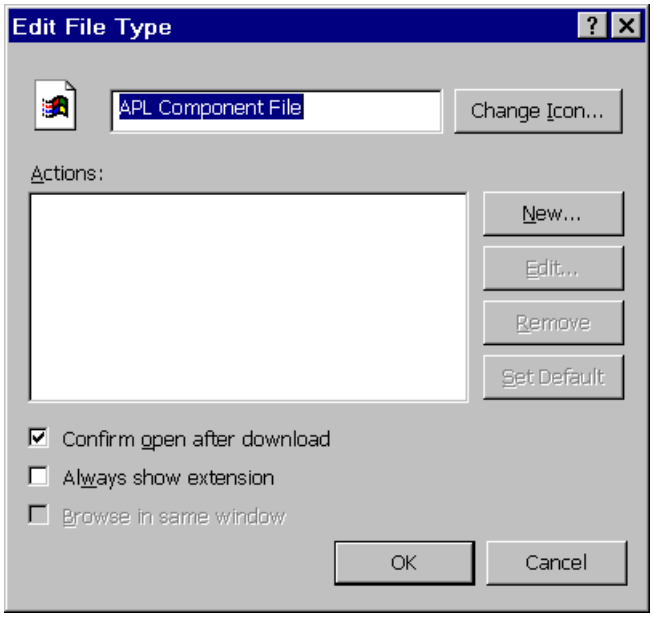
Click the File Types tab, and scroll down to SF files



Note: the File Types tab is missing in Windows 7!

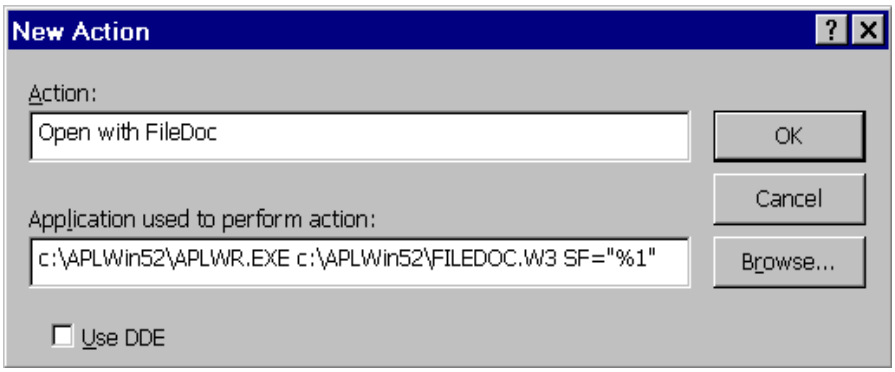
To get this functionality back, download a nice little utility **Default Programs Editor** at defaultprogramseditor.com

Click the Advanced button

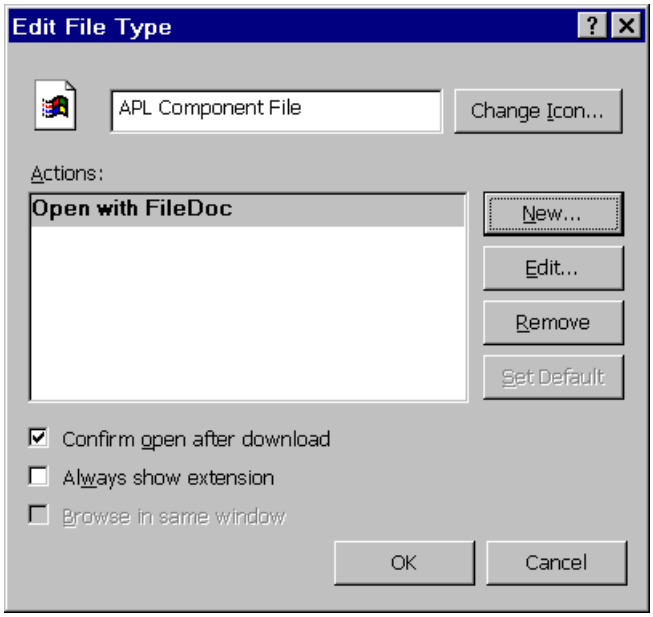


Click the New button

Type `Open with FileDoc`
and `C:\APLWIN\APLWR.EXE C:\APLWIN\FILEDOC.W3 SF="%1"`
Note! `APLWR` not `APLW`

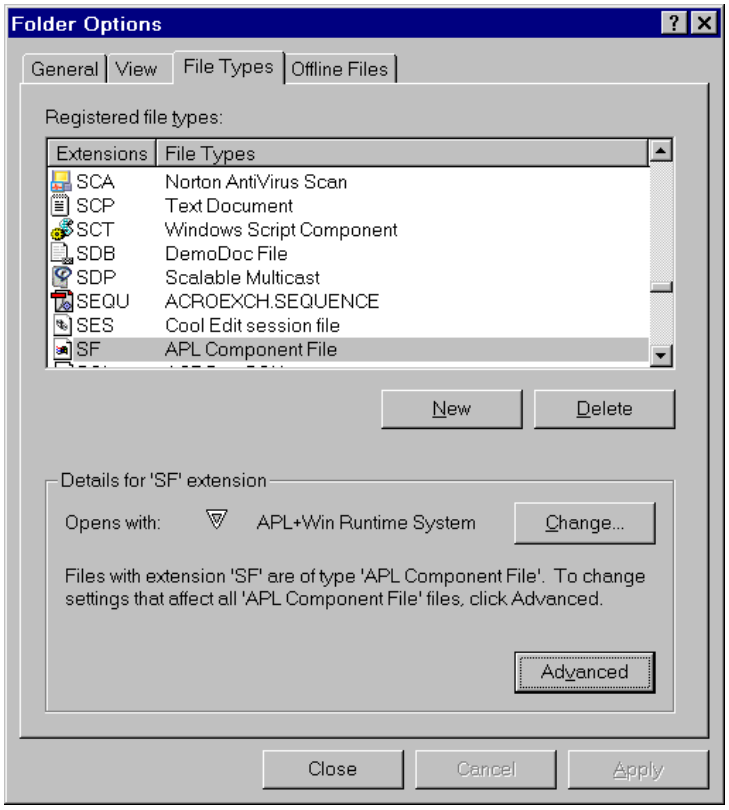


Click OK



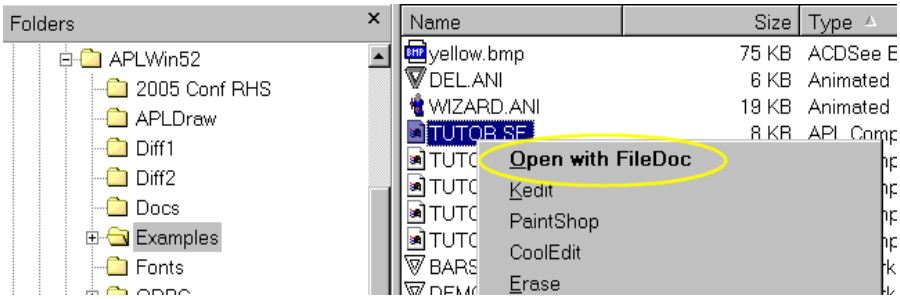
Note that Open with FileDoc is in bold face. That means it's the default action for the .SF file type.

Click OK.

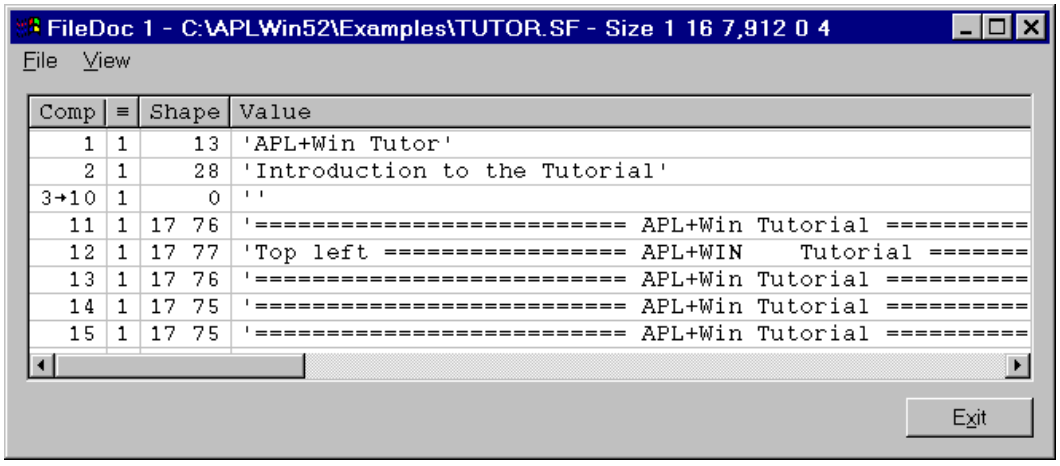


Click Close.

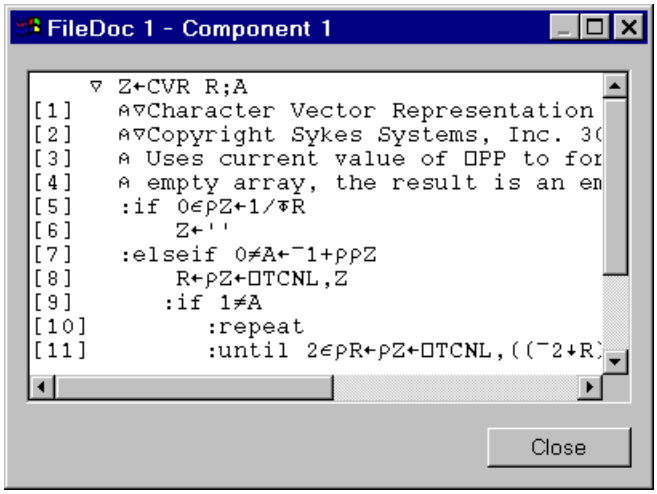
Now if you right-click on an .SF file, you should see Open with FileDoc on the context menu.



If you choose that menu item, or just double-click on the file, FileDoc should come up. And if someone emails an SF file to you, you should be able to take a quick look at it just by double-clicking the attachment.



Note that since this runs the APL+Win runtime engine, `)EDIT` is not possible. So when you peek at a file component, an Edit box is used.



NOTLOCAL: Search for unlocalized references

When you search for references to a variable, you may not care about localized references.]NOTLOCAL ignores any hits in functions where the name is localized.

```
      ]notlocal ⎕io

Searching 320 functions ...
==> AlwaysOnTop (1)
AlwaysOnTop[5]   b←(⎕io+b)⊃'hwnd_notopmost' 'hwnd_topmost'
==> AutoResize (4)
AutoResize[15]          w[;⎕io]←w[;⎕io]×(↑⍴w)⍴s
AutoResize[37]          w[;⎕io]←w[;⎕io]×(↑⍴w)⍴s
==> FFAW (3)
FFAW[87]   P←' ,'[⎕IO+P]                  ⍝ Else space before ⍝ 07 Jul 2002
FFAW[110]  F←F-D×⎕IO                      ⍝ Adjust first for display origin
FFAW[121]  →(⎕IO+2⊥''''''⍷A)⊃L90,L91,L92,L93
==> GetWinVer (2)
GetWinVer[17]       struc[3+⎕IO]←65536⊤struc[3+⎕IO]
==> goo (1)
[1]   ⎕io←1 ⋄ foo
[2]   (⎕io ⎕ct)←0 ⋄ foo
[3]   :for ⎕io :in 0 1
```

Let's say a global variable has changed unexpectedly. Include an assignment arrow to find unlocalized *assignments*. Note that all forms of assignment are detected -- not just those where an assignment arrow directly follows the name.

```
      ]notlocal ⎕io←

Searching 320 functions ...
==> goo (1)
[1]   ⎕io←1 ⋄ foo
[2]   (⎕io ⎕ct)←0 ⋄ foo
[3]   :for ⎕io :in 0 1
```

FNSS and VARSS: Search function/variable names

These are simple but I use them all the time. Was that variable named ProxyUser or UserProxy?

```
      ]varss prox
 ∆ProxyHost ∆ProxyPass ∆ProxyPort ∆ProxyUse ∆ProxyUser
```

Use a semi-colon to search for more than one substring at once:

```
      ]varss dir;path
 AAmixdir ∆AAmixdir ∆ExeDir ∆Path ∆SysDir ∆WinDir
```

Use /C to match case:

```
      ]fnss help
HtmlHelp help help1 help2 helpg helpw
      ]fnss Help /c
HtmlHelp
```

Use a leading or trailing blank and a semi-colon to match the string at the beginning or end of a name:

```
      ]fnss date
CheckForUpdates DATEBASE DATEREP EditDate
      ]fnss ; date
DATEBASE DATEREP

      ]fnss num
EditNum EditNumFmt EditNumVifi FMTNUM GETNUMFMT
      ]fnss num ;
EditNum FMTNUM
```

VARFIND: Search variables for a value

```
      ]varfind 'Fixed'

Searching 880 variables for 'Fixed'
((⊂1 1)⊃AAmixdir) ← '100% Fixed Income'
griddoc <162> ← 'xFixedCols property:'
griddoc <163> ← "  Value@Long ← ⎕WI 'xFixedCols'"
griddoc <164> ← "  ⎕WI 'xFixedCols' Value@Long"
griddoc <169> ← 'xFixedRows property:'
griddoc <170> ← "  Value@Long ← ⎕WI 'xFixedRows'"
griddoc <171> ← "  ⎕WI 'xFixedRows' Value@Long"
```

Note that unlike most user commands, the argument is executed! So above,
Use: `]varfind 'FOO'`
Not: `]varfind FOO`

```
      ]varfind 10

Searching 880 variables for 10
AAptiles ← 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
BBS_RAISED ← 10
BORDER_RAISED ← 10
CHART_Y2LEGFT ← 10
CPI_3DINFO ← 10
CSA_XGAP ← 10
HIT_RIGHTTITLE ← 10
MK_CUBE ← 10
RIFdata[1;] ← 10 1.5
SURFACE ← 10
(14 2 (13 4) 1⊃VARG0) ← ,10
(15 2 (13 4) 1⊃VARG2) ← ,10
(63 2 (13 4) 1⊃VARG2) ← ,10
```

```
      ]varfind 1 2

Searching 880 variables for 1 2
AAmixes ← 1 2 3
EForder[2;] ← 1 2 'Market Drop'
PPorder[2;] ← 1 2 'Medium Growth'
```

SI: Enhanced)SI

After an error, rather than see just the stack of function names and line numbers...

```
DOMAIN ERROR
TextSizePix[4]    z←1⊃⎕wi 'Draw' ('Scale' 5-1) ('?Text' t)

      )si

TextSizePix[4] *
chart3a[219]
sel3[329]
⍎
sel0[102]
>[fmTK.fMain.sMain.pgHome.fWork.wb;XBeforeNavigate2]
```

Let's see the code too...

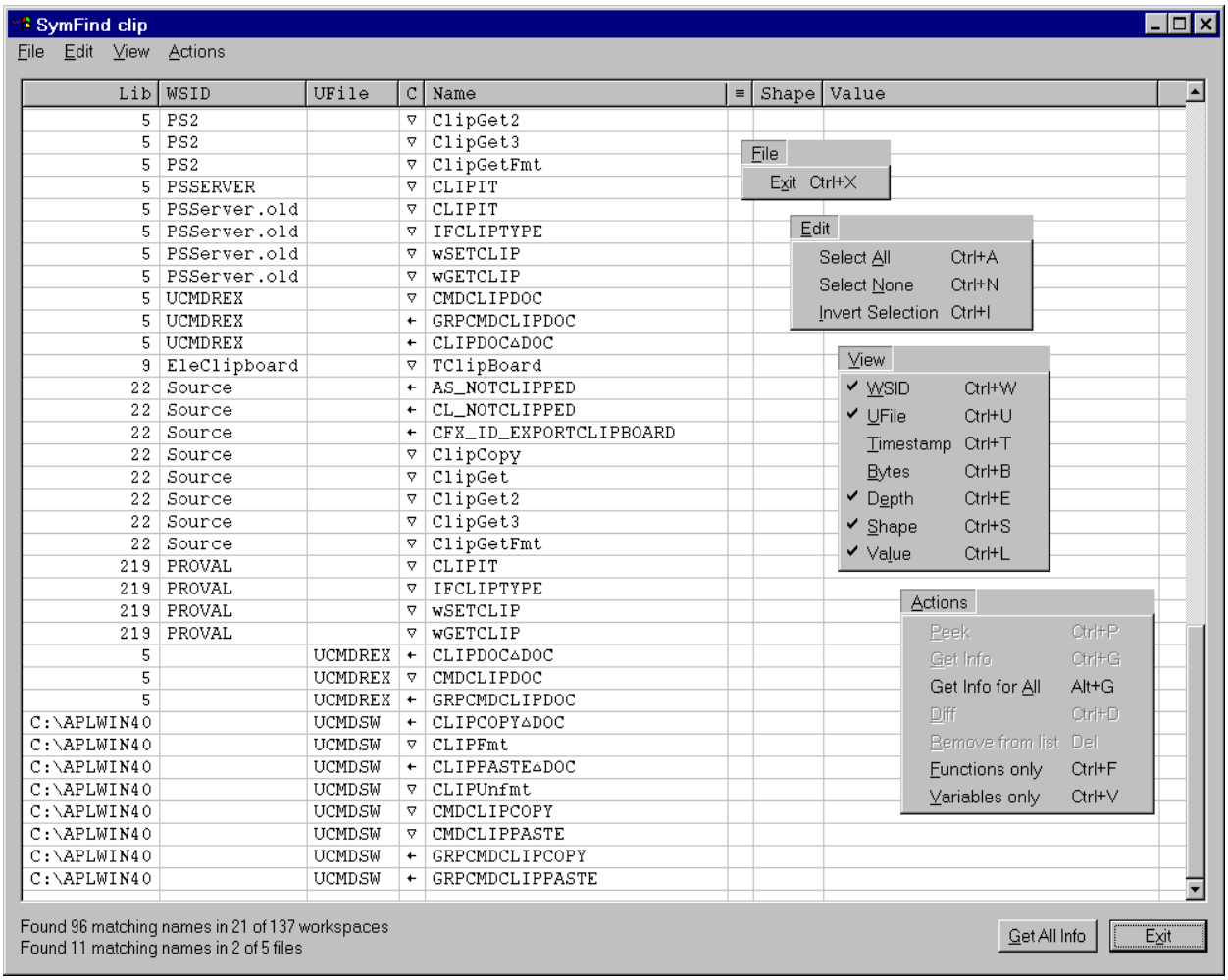
```
      ]si

TextSizePix[4] * z←1⊃⎕wi 'Draw' ('Scale' 5-1) ('?Text' t)
chart3a[219] (h w)←p TextSizePix (i⊃xl)  ⍝ p is the chart's parent frame
sel3[329] chart3a 0              ⍝ 0=make new chart from scratch
⍎
sel0[102] ⍎x
>[fmTK.fMain.sMain.pgHome.fWork.wb;XBeforeNavigate2]
```

SYMFIND: Search saved workspaces and ucmd files

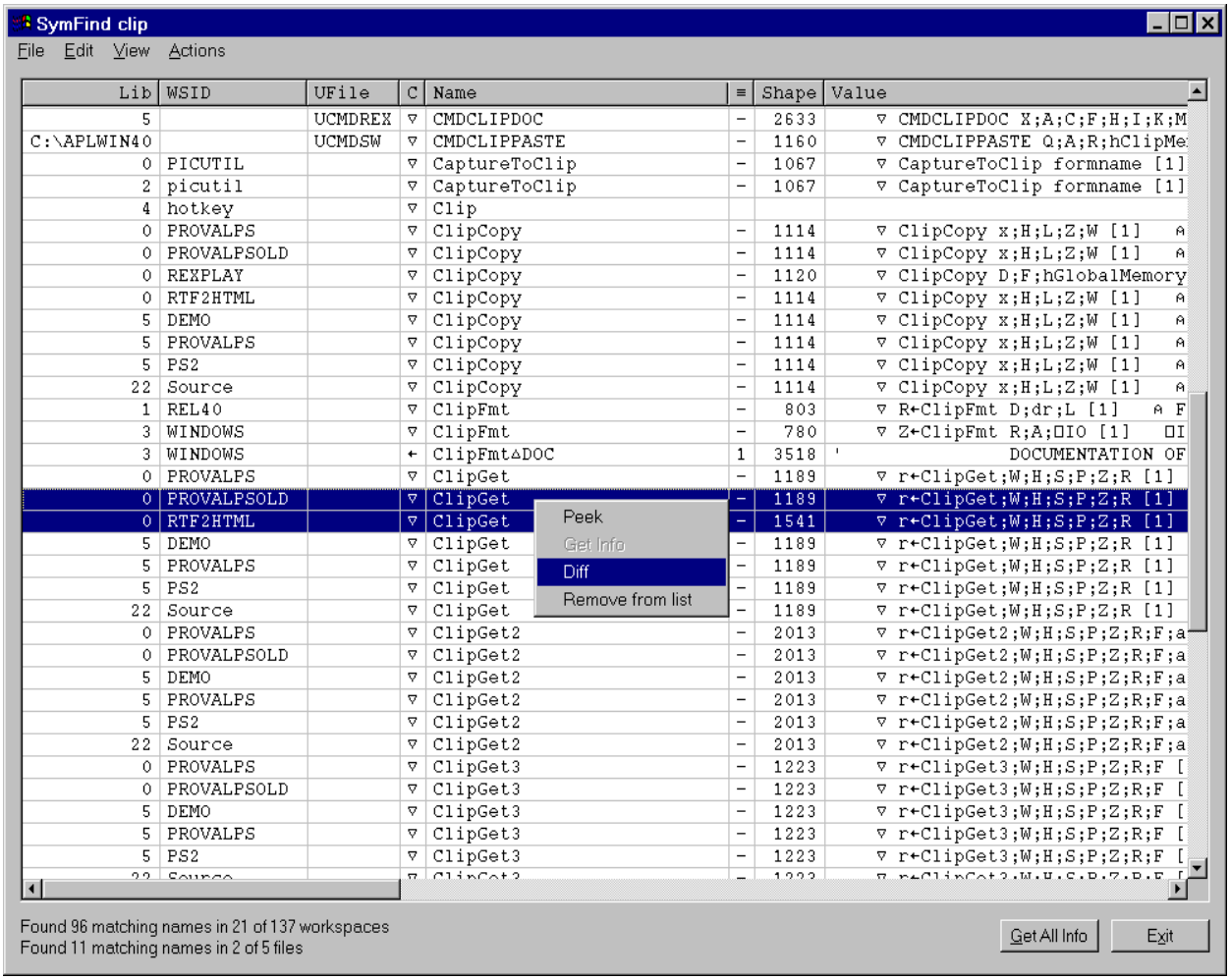
Let's say you remember having a function named Clip-something or Something-Clip, and you've already used]FNSS to ascertain that it's not in your active workspace. This tool searches saved workspaces and user command files for objects with a certain name.

And it does it without loading any workspaces! It reads the raw .w3 files and searches the symbol tables.

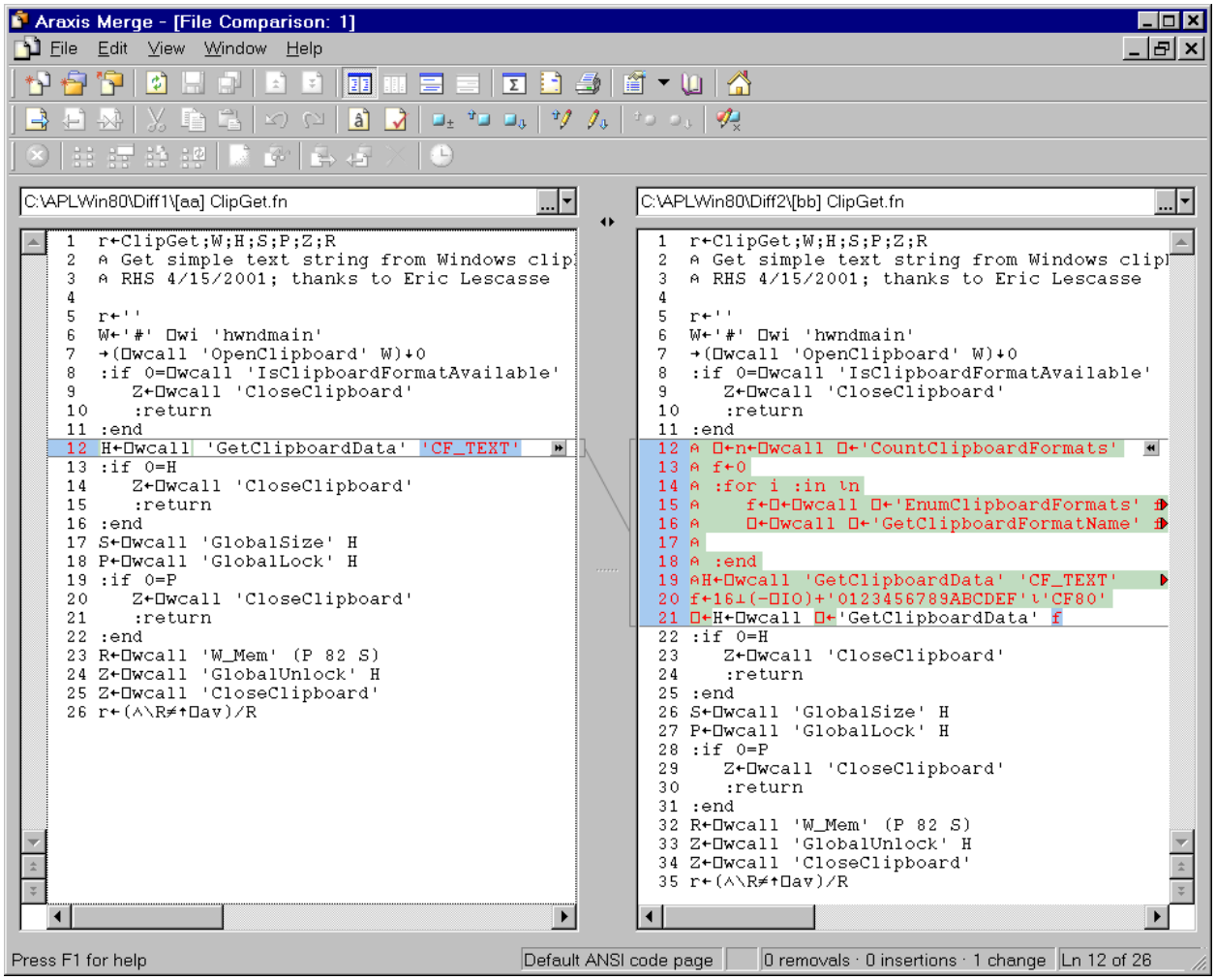
```
]symfind clip
```



If you select one or more rows and right-click, a context menu appears:



And if you choose the DIFF option...



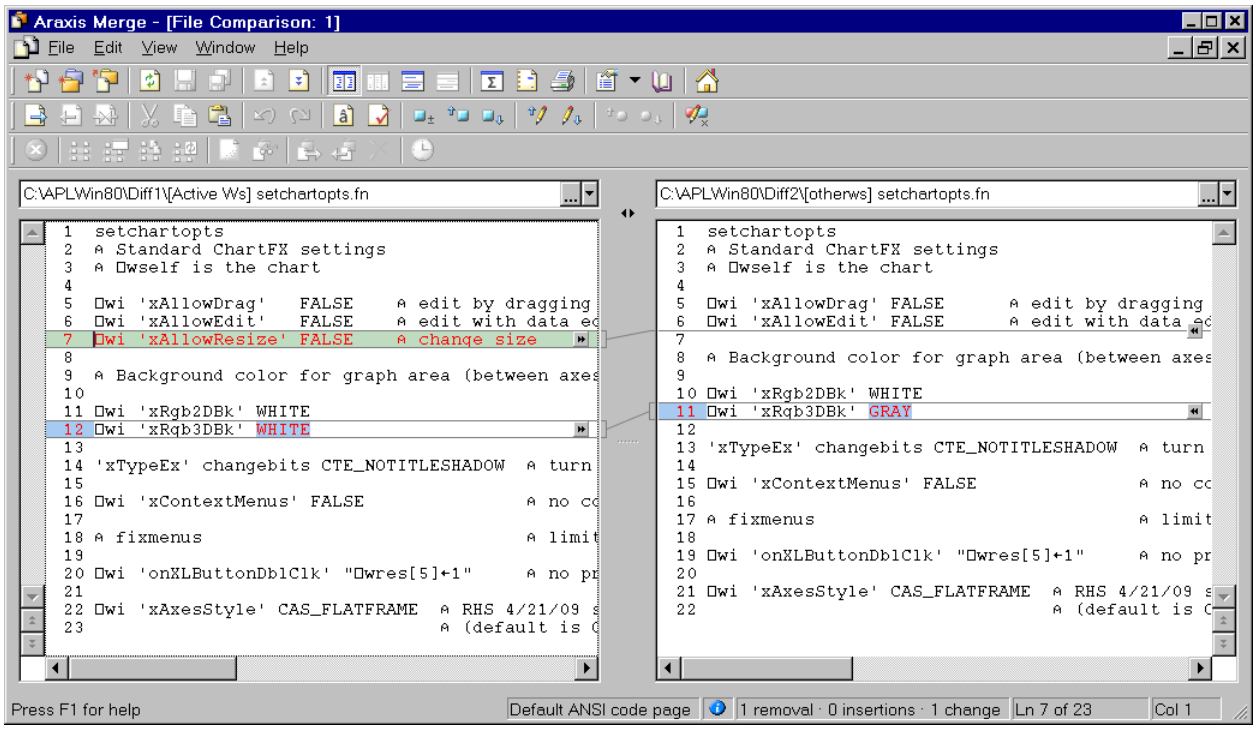
This display comes from a tool called **Araxis Merge**. This is the best $129 I ever spent. See www.araxis.com/merge/ for more information. My DIFF tool sends APL functions and variables to it (see following page).

DIFF: Show differences between two similar objects

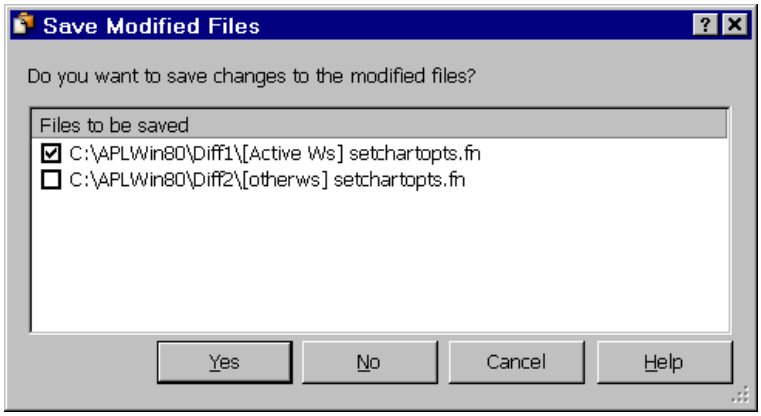
The display on the previous page comes from a tool called **Araxis Merge**. This is the best $129 I ever spent. See www.araxis.com/merge/ for more information. My DIFF tool sends APL functions and variables to it.

```
      ]diff ?

Compare (show the DIFFerences between) two functions or variables

Syntax:  ]DIFF OBJECT1 ; OBJECT2
Where each object may be:
  Active ws:
    FOO                      ⍝ object name (function or variable)
  Saved ws:
    UTILS FOO                ⍝ wsid, object name
    9 STUFF FOO              ⍝ library number, wsid, object name
    \EUDORA\ATTACH\UTIL FOO  ⍝ path, wsid, object name
  User command file:
    FOO /F=UCMDS             ⍝ object name and ucmd file to get it from
    FOO /F=19 UCMDS          ⍝ object name and ucmd file with lib number
    FOO /F                   ⍝ object name, from first ]UFILE file
  APL file component:
   tn,cn                     ⍝ tie number, component number
   tn#cn                     ⍝ tie number, component number
You may use "=" for the second wsid or object name
   Example: ]DIFF FOO;UTILS =

Alternate syntax (borrowed from ]COMP) with no semicolon:
   ]DIFF  object1  object2
Each object may be one of the following types:
   fnname  [/F[=cmdfile]]    ⍝ function
   varname [/F[=cmdfile]]    ⍝ variable
   tn,cn                     ⍝ APL file: tie number, component number
   tn#cn                     ⍝ APL file: tie number, component number
Examples:
   ]DIFF FUN GUN
   ]DIFF FUN FUN /F=5 UCMDS
   ]DIFF FUN /F=5 UCMDS GUN
   ]DIFF FUN /F=5 UCMDS GUN /F=9 MYUCMDS
You may use "=" for the second object name
Note that you may not specify WSIDs when using this no-semicolon syntax

Execute   ]DIFF /ERASE   occasionally to clean up temp files
(Sorry, this feature does not work with Win98)

You may establish a list of user command files with
   ]DIFF /ULIST=file1;file2;file3
and query it with
   ]DIFF /ULIST
Then use
   ]DIFF FOO;= /U
/U triggers a search through the list of files established above
for the first object named FOO
```

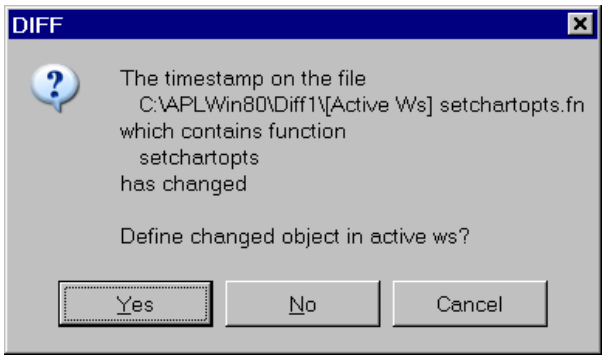
```
]diff setchartopts ; otherws =
```



If you use Araxis Merge to change one of the objects (e.g., merge line 11 from right window into left) and then close Araxis Merge, saving the file (click Yes)...

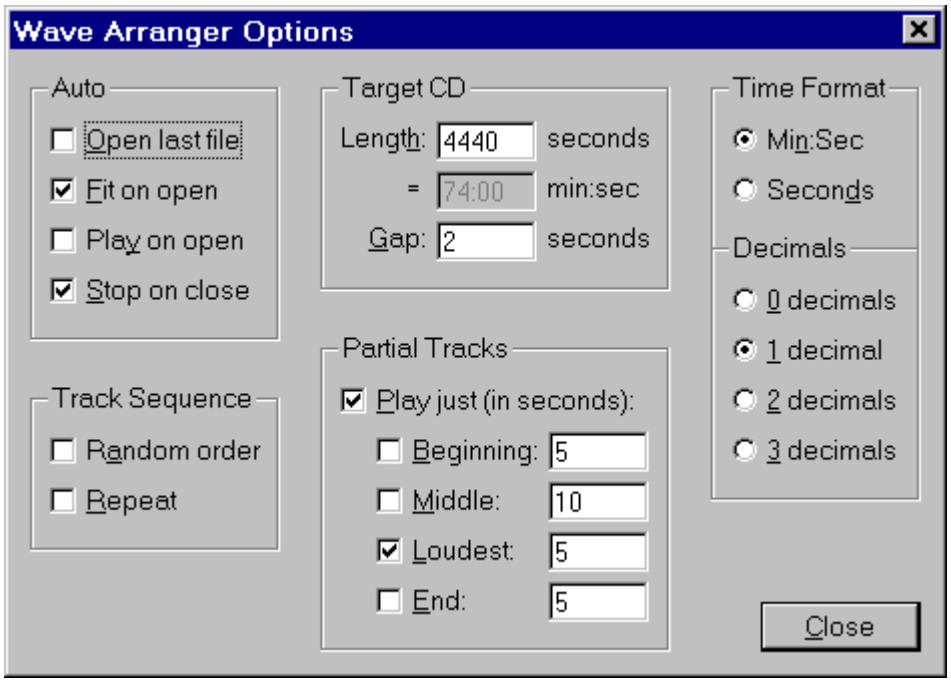


]DIFF will notice the changed file and offer to redefine the changed object in your active ws.



SHORTKEYS: Shortcut Keys

I confess I don't use this very much. But the idea is that you have a Form with a bunch of shortcut keys and you want a list of them so you find letters that aren't used yet before creating a new control.




```
]shortkeys fmWVopts
```

```
Shortcut Keys in: fmWVopts [Form]:
 Alt+0  Option  fmWVopts.fDec.op0d          &0 decimals
 Alt+1  Option  fmWVopts.fDec.op1d          &1 decimal
 Alt+2  Option  fmWVopts.fDec.op2d          &2 decimals
 Alt+3  Option  fmWVopts.fDec.op3d          &3 decimals
 Alt+A  Check   fmWVopts.fTracks.ckRandom   R&andom order
 Alt+B  Check   fmWVopts.fPartial.ckSeg1    &Beginning:
 Alt+C  Button  fmWVopts.bnClose            &Close
 Alt+D  Option  fmWVopts.fTime.opSec        Secon&ds
 Alt+E  Check   fmWVopts.fPartial.ckSeg4    &End:
 Alt+F  Check   fmWVopts.fAuto.ckFit        &Fit on open
 Alt+L  Check   fmWVopts.fPartial.ckSeg3    &Loudest:
 Alt+M  Check   fmWVopts.fPartial.ckSeg2    &Middle:
 Alt+N  Option  fmWVopts.fTime.opMin        Mi&n:Sec
 Alt+O  Check   fmWVopts.fAuto.ckOpen       &Open last file
 Alt+P  Check   fmWVopts.fPartial.ckPartial &Play just (in seconds):
 Alt+R  Check   fmWVopts.fTracks.ckRepeat   &Repeat
 Alt+S  Check   fmWVopts.fAuto.ckStop       &Stop on close
 Alt+Y  Check   fmWVopts.fAuto.ckPlay       Pla&y on open
 Ctrl+K SubMenu fmWVopts.mSegs              mSegs
 Ctrl+T SubMenu fmWVopts.mOpts              mOpts
Available: Ctrl+ A B C D E F G H I J L M N O P Q R S U V W X Y Z
Available: Alt+  G H I J K Q T U V W X Z
```

PEM: Properties, Events, and Methods

A simple tool that I use every day!

What are the properties, events, and methods of a specific control?

```
lpem fmWVopts.fAuto.ckFit
```

```
    Object: fmWVopts.fAuto.ckFit
     Class: Check
Properties: border caption children class color data def deferexit
            edge enabled events extent font help helpcontext hwnd
            instance keys links methods mode modified modifystop name
            noredraw opened order place pointer prompt properties
            scale scrollaccel scrollmargin self size state style
            suppress tabgroup tabparent tabstop targetformats tooltip
            tooltipwidth translate value visible where ++ Hwnd Self
            _data dragimage limitwhere nograychildren onAction
            siblings sourceformats transparent wedProps xdef
    Events: Action Click Close Delete Destroy DragEnter DragLeave
            DragOver Drop Exit ExitError Focus Help Hide KeyDown KeyUp
            Modified MouseDouble MouseDown MouseDrag MouseEnter
            MouseLeave MouseMove MouseUp Move Open Paint Reopen Resize
            Send Show Unfocus ++ onDragContinue onDragEnd
            onDragFeedback onDragStart onPreCreate onWindowPosChanged
            onWindowPosChanging
   Methods: Close Create Defer Delete Draw Event Exec Focus Help Hide
            Info Modify New Open Paint Popup Ref Resize Send Set
            SetLinks Show ++ Layout Msg
```

What about a generic Button control? And just show me /p (properties), please.

```
lpem Button /p
```

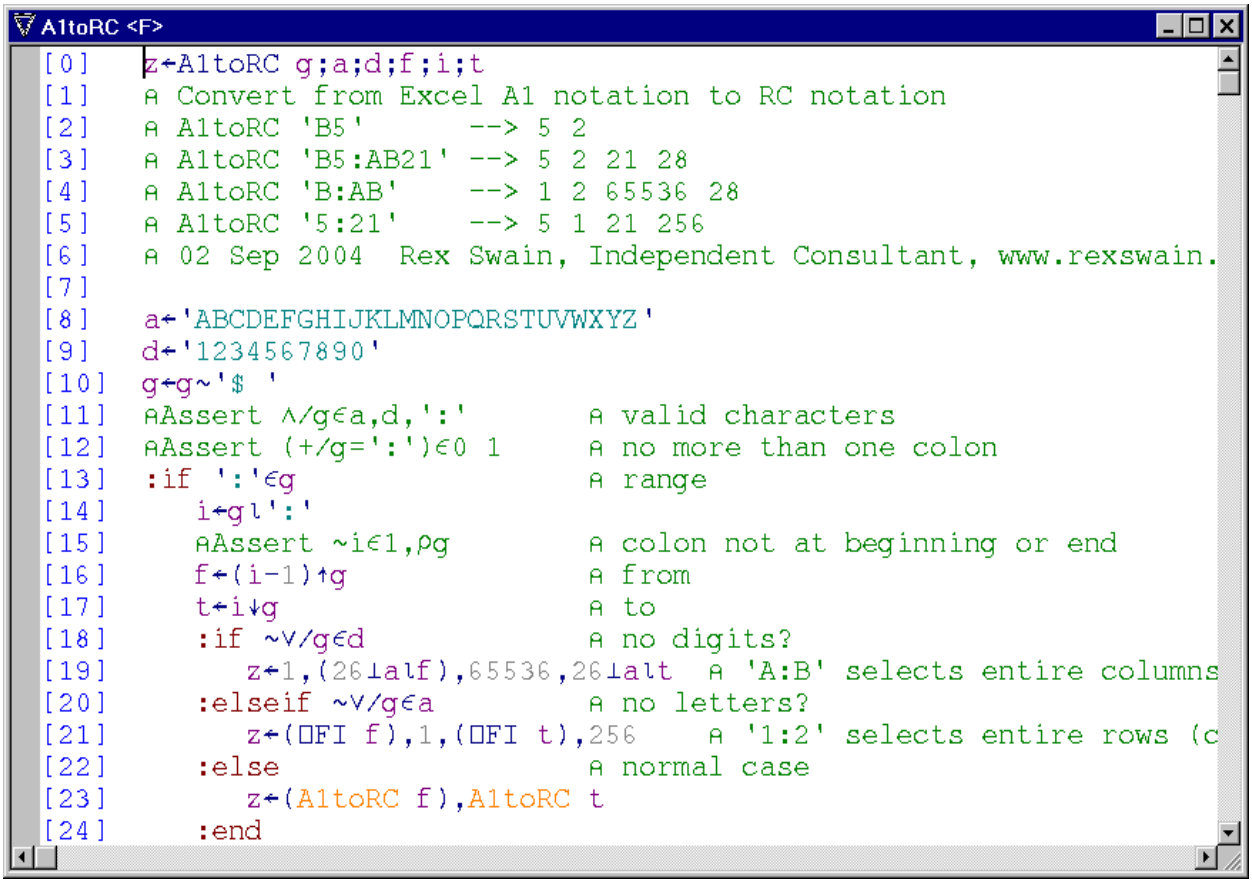
```
     Class: Button
Properties: border caption children class color data def deferexit
            edge enabled events extent font help helpcontext hwnd
            imageindex imagelist imagespace instance keys links
            methods mode modified modifystop name noredraw opened
            order place pointer prompt properties scale scrollaccel
            scrollmargin self size state style suppress tabgroup
            tabparent tabstop targetformats tooltip tooltipwidth
            translate value visible where ++ Hwnd Self _data dragimage
            limitwhere nograychildren onAction siblings sourceformats
            transparent wedProps xdef
```

This tool uses the 'Info' method to find hidden properties, events, and methods; these are shown after "++" in the display.

PEEK: Peek at an Object in a Workspace

I want to look at the function `A1toRC` in workspace `5 XL`, but I don't want to copy it into my active workspace.

```
]peek 5 xl A1toRC
```

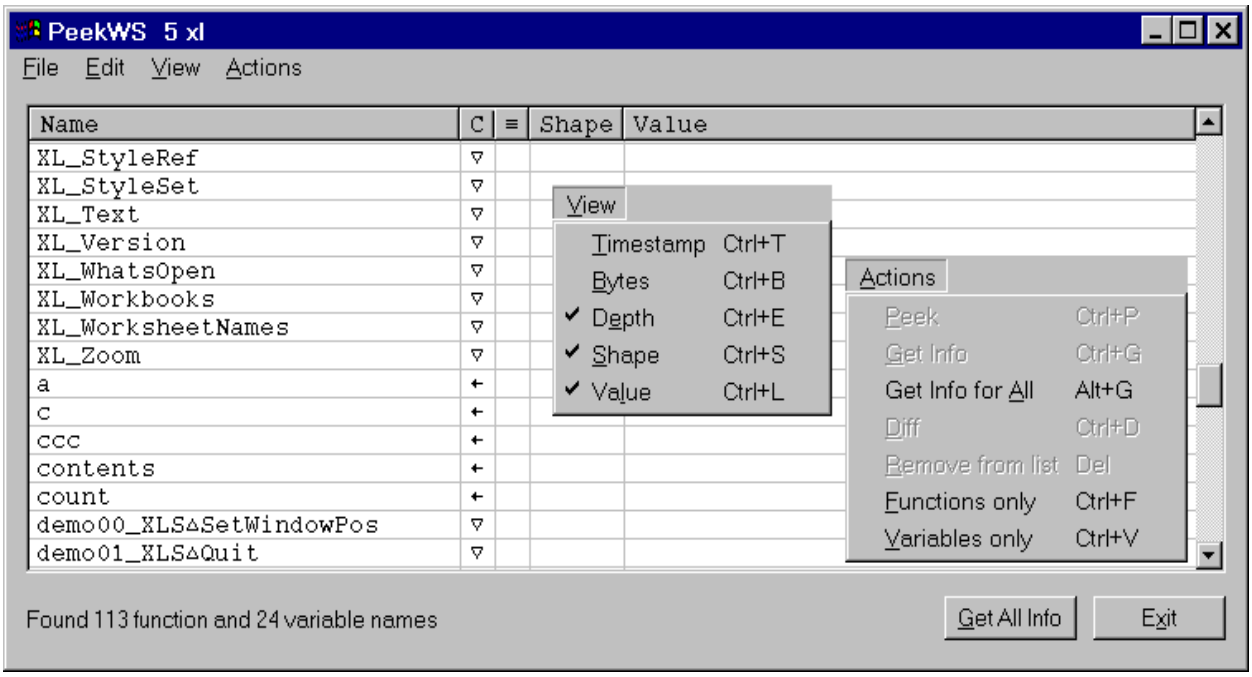


Ctrl+Q to quit -- function was localized and will disappear from active ws
Ctrl+E to define in active ws

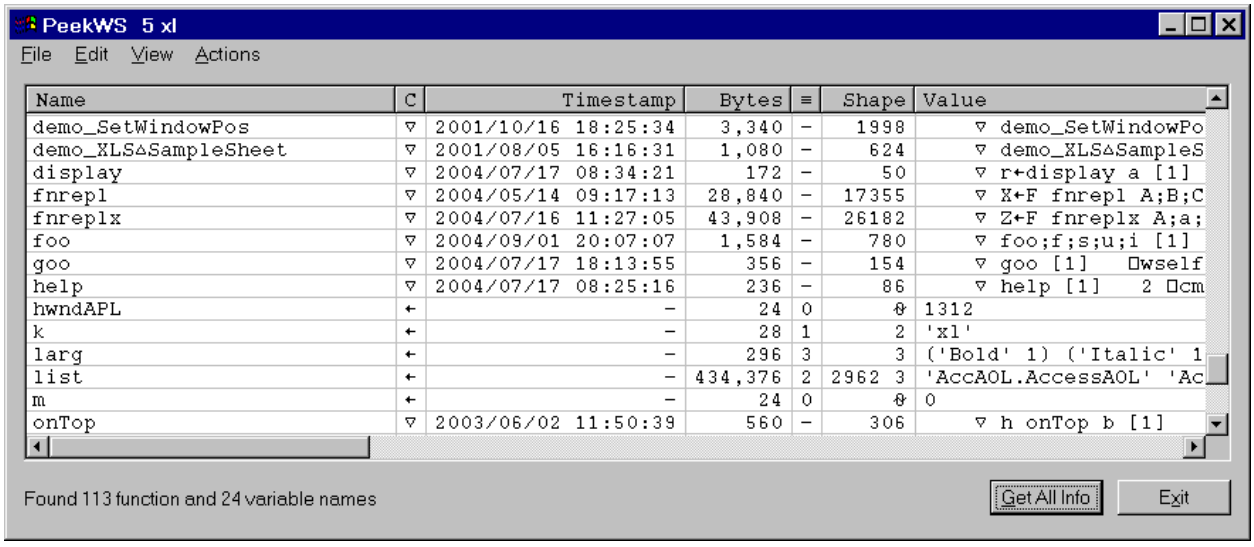
PEEKWS: Peek at a Workspace

Suppose I want to look at the entire workspace `5 XL`...

```
]peekws 5 xl
```



Click Get All Info button; view Timestamps and Bytes



CLIPDOC: What's In My Clipboard?

Copy 8 characters 'cliptest' from APL session into clipboard

```
      ]clipdoc
```

```
Format Hex Name              Bytes Value
------ --- ---------------- ----- ---------------------------------------
     1   1 ⍟ CF_TEXT           12 'cliptest',□TCNUL,□TCNUL,□TCNUL,□TCNUL
    16  10                      4 □TCHT,'◇',□TCNUL,□TCNUL
     7   7 ⍟ CF_OEMTEXT        12 'cliptest',□TCNUL,□TCNUL,□TCNUL,□TCNUL
    13   D ⍟ CF_UNICODETEXT    24 'c',□TCNUL,'l',□TCNUL,'i',□TCNUL,'p',□T
```

Copy 8 characters 'cliptest' from Microsoft Word into clipboard

```
      ]clipdoc
```

```
Format  Hex Name                    Bytes Value
------ ---- ---------------------- ------ --------------------------------
 49161 C009 DataObject                  4 'ó€',□TCLF,□TCNUL
 49166 C00E Object Descriptor         120 'x',□TCNUL,□TCNUL,□TCNUL,'←',□T
 49303 C097 Rich Text Format        3,434 '{\rtf1\ansi\ansicpg1252\uc1\de
 49279 C07F HTML Format             2,379 'Version:1.0',□TCNL,□TCLF,'Star
     1    1 ⍟ CF_TEXT                   9 'cliptest',□TCNUL
    13    D ⍟ CF_UNICODETEXT           18 'c',□TCNUL,'l',□TCNUL,'i',□TCNU
    14    E ⍟ CF_ENHMETAFILE            0 (Could not get GlobalLock)
     3    3 ⍟ CF_METAFILEPICT          16 □TCBS,□TCNUL,□TCNUL,□TCNUL,'`O'
 49163 C00B Embed Source           19,968 '-Ïæαí□→ß',16ρ□TCNUL,'>',□TCNUL
 49156 C004 Native                 19,968 '-Ïæαí□→ß',16ρ□TCNUL,'>',□TCNUL
 49155 C003 OwnerLink                  28 'Word.Document.8',□TCNUL,'Docum
 49165 C00D Link Source               132 □TCHT,'≤',6ρ□TCNUL,'À',6ρ□TCNUL
 49167 C00F Link Source Descriptor    140 'î',□TCNUL,□TCNUL,□TCNUL,'←',□T
 49154 C002 ObjectLink                 37 'Word.Document.8',□TCNUL,'Docum
 49171 C013 Ole Private Data          408 □TCNUL,□TCNUL,□TCNUL,□TCNUL,'⊞§
    16   10                            4 □TCHT,'◇',□TCNUL,□TCNUL
     7    7 ⍟ CF_OEMTEXT                9 'cliptest',□TCNUL
```

Press the PrintScreen key

```
      ]clipdoc
```

```
Format Hex Name             Bytes Value
------ --- ---------- --------- ---------------------------------------
     2   2 ⍟ CF_BITMAP         0 (Could not get GlobalLock)
     8   8 ⍟ CF_DIB    9,437,236 '(',□TCNUL,□TCNUL,□TCNUL,□TCNUL,□TCHT,□T
    17  11             9,437,320 '¦',□TCNUL,□TCNUL,□TCNUL,□TCNUL,□TCHT,□T
```

MF: Monitor Functions

```
]mf /on
...
]mf
```

MF summary for wsid: 5 REXPLAY

File View Resolution (Millisecs)

Function	Total	Here	Pct	⌈/It	Calls	Header
wavefile	1604.550	1603.667	34.03	42	42	r←m wavefile f;⎕ELX
wvh	12082.189	1154.167	24.49	1733	1733	r←opt wvh event;a;b;c;d;
FFAW	1106.679	820.397	17.41	5284	5284	R←W FFAW A;D;E;F;K;N;P;S
TraceMe	1917.115	679.415	14.42	1742	1742	TraceMe;d;e;f;h;r;s;t;v;
wv	339.079	184.786	3.92	8	2	wv;b;d;g;h;i;lw;n;p;q;t;
wvfmtclock	124.662	83.412	1.77	1393	1393	r←wvfmtclock m;s;f;p
wvo	81.176	54.307	1.15	4	4	opt wvo event;f;fh;fw;f1
fexist	52.478	51.617	1.10	42	42	r←fexist f;t;⎕ELX
Ini	48.269	45.802	0.97	126	126	r←v Ini a;d;e;f;s
fread	16.841	16.788	0.36	2	2	r←fread f;t;⎕ELX
Parse	10.214	4.204	0.09	76	76	r←Parse a;q;i;w
workarea	4.125	2.962	0.06	7	7	a←workarea p;e;h;i;n;r;s
mrufiles	15.289	2.346	0.05	36	4	w mrufiles a;f;s;i;m;c;n
dlb	3.555	2.170	0.05	152	152	r←dlb v
TextWidth	1.552	1.362	0.03	16	16	w←o TextWidth t
AlwaysOnTop	1.397	1.312	0.03	2	2	⎕wself AlwaysOnTop b;h;v
trypath	1.281	1.204	0.03	4	4	r←trypath p;⎕ELX
TextHeight	0.677	0.617	0.01	4	4	h←o TextHeight t
KeepOnScreen	15.796	0.538	0.01	5	5	KeepOnScreen;a;c;g;p;u;w
Assert	0.753	0.512	0.01	21	21	w Assert a
upper	0.820	0.450	0.01	38	38	r←upper a
Sink	1.739	0.430	0.01	172	172	Sink a

Double-click on the 4th row:

MF detail for function: TraceMe

File View Resolution (Millisecs)

Line	Total	Here	Iter	Instruction
TraceMe	1917.115	679.415	1742	∇ TraceMe;d;e;f;h;r;s;t;v;w;x
TraceMe[1]	0.000	0.000	0	[1] ⍝ Trace callback function
TraceMe[2]	0.000	0.000	0	[2] ⍝ 11 Feb 2003 Rex Swain, Independ
TraceMe[3]	0.000	0.000	0	[3]
TraceMe[4]	11.213	11.213	1742	[4] →⍙RunTime⍴0 ⍝ only
TraceMe[5]	8.894	8.894	1742	[5] w←⎕PW
TraceMe[6]	797.545	47.452	1742	[6] r←'[',⎕wself,'] ',(w FFAW ⎕wevent)
TraceMe[7]	31.421	31.421	1742	[7] s←⎕SI ⍝ si s
TraceMe[8]	9.892	9.892	1742	[8] →(1=↑⍴s)⍴0 ⍝ done
TraceMe[9]	5.008	5.008	1742	[9] t←''
TraceMe[10]	6.810	6.810	1742	[10] :if 1<↑⍴s ⍝ if a
TraceMe[11]	27.970	27.970	1742	[11] f←s[2;] ⋄ f←(∧\f≠'[')/f ⍝ call
TraceMe[12]	20.176	20.176	1742	[12] d←+/s[;⍳1+⍴f]∧.=f,'[' ⍝ recu
TraceMe[13]	4.723	4.723	1742	[13] e←f
TraceMe[14]	15.174	15.174	1742	[14] e←e,(d>1)/' [depth ',(⍕d),']'
TraceMe[15]	5.690	5.690	1742	[15] e←e,':'
TraceMe[16]	14.506	14.506	1742	[16] (x v)←2⍴1 ⎕AT f ⍝ expl
TraceMe[17]	5.968	5.968	1742	[17] :if v>0 ⍝ if a
TraceMe[18]	12.576	12.576	1737	[18] h←⎕CRL f,'[0]' ⍝ head
TraceMe[19]	15.661	15.661	1737	[19] h←(¯1+h⍳';')↑h ⍝ igno

FNREPLX: Workspace Search/Replace with *Regular Expressions*

- Before using `fnreplx` , copy file APL_PCRE.DLL into your APLW directory.
- `fnreplx` is not a user command; load it into your active ws:

```
]uload fnreplx /f=ucmdsrex
```

```
... fnreplx ...
```

Right argument:

```
... fnreplx 'target'                       ⍝ Search (if target begins with a letter)
... fnreplx 'target/options'               ⍝ Ditto, with options
Any non-alpha delimiter character may be used in place of the slashes shown below
... fnreplx '/target'                      ⍝ Search (exactly 1 delimiter)
... fnreplx '/target/'                     ⍝ Search (exactly 2 delimiters)
... fnreplx '/target/options'              ⍝ Ditto, with options
... fnreplx '/target/replacement/'         ⍝ Replace (exactly 3 delimiters)
... fnreplx '/target/replacement/options'  ⍝ Ditto, with options
```

Regex options:

```
... fnreplx '.../i'  ⍝ Case-insensitive matching
... fnreplx '.../m'  ⍝ Multi-line
... fnreplx '.../x'  ⍝ Extended syntax: un-escaped white space is ignored
... fnreplx '.../s'  ⍝ Dot meta-character matches all characters including new-lines
```

Left argument:

```
        fnreplx ...  ⍝ Look in all functions
'F1 F2' fnreplx ...  ⍝ Only look in named function(s)
   'F⋆' fnreplx ...  ⍝ Function names with wildcards (foo⋆ ⋆old fm⋆def etc.)
  '~F1' fnreplx ...  ⍝ Look in all but named function(s)
    '?' fnreplx ...  ⍝ Prompt for confirmation before each replacement
    '⁻' fnreplx ...  ⍝ Underline the hits as they are displayed ('-' okay too)
    '←' fnreplx ...  ⍝ Suppress display; return fn names
    '⍝' fnreplx ...  ⍝ Match only in comments
    "'" fnreplx ...  ⍝ Match only in character constants ('"' okay too)
    'α' fnreplx ...  ⍝ Match only in APL (not comments/quotes)


Function line numbers in brackets are not considered part of the function
(I display the line numbers so you can double-click on "foo[6]" to start editor)


Function lines may have leading blanks, but never have trailing blanks
Use  /^FOO/m     to find FOO at the (very) beginning of a line
Use  /^\ ⋆FOO/m  to find FOO at the beginning of a possibly-indented line
Use  /FOO$/m     to find FOO at the end of a line


Special regex characters that must be escaped in target: \ | ^ $ . ? ⋆ + [] () {}


PCRE comes from a Perl/Unix heritage, where linefeeds are used between lines of text
So when they say "newline" (\n) they are talking about ⎕TCLF
So when they say "return"  (\r) they are talking about ⎕TCNL
To make the regex engine work, I put LF's (not NL's) between function lines
```

Regex cheat sheet

```
Special characters, outside of square brackets:
  \   general escape character with several uses
  ^   assert start of string (or line, in multiline mode)
  $   assert end of string (or line, in multiline mode)
  .   match any character except newline (by default)
  [   start character class definition
  |   start of alternative branch
  (   start subpattern
  )   end subpattern
  ?   extends the meaning of (
         also 0 or 1 quantifier
         also quantifier minimizer
  *   0 or more quantifier
  +   1 or more quantifier
         also "possessive quantifier"
  {   start min/max quantifier

Special characters, inside square brackets:
  \   general escape character
  ^   negate the class, but only if the first character
  -   indicates character range
  ]   terminates the character class

Non-printing characters:
  \a     alarm (dec 7, hex 07) □TCBEL
  \cx    "control-x", where x is any character
  \e     escape (dec 27, hex 1B) □TCESC
  \f     formfeed (dec 12, hex 0C) □TCFF
  \n     newline (dec 10, hex 0A) □TCLF  ₳ note Unix nomenclature!
  \r     return (dec 13, hex 0D) □TCNL
  \t     tab (dec 9, hex 09) □TCHT
  \ddd   character with octal code ddd (or backreference)
  \xhh   character with hex code hh

Generic character classes (upper-case for "not")
  \d   digit [0123456789]
  \s   whitespace character (blank, tab, LF, CR, formfeed)
       (caution: this also matches LFs between lines!  use \x20 for blank)
  \w   word character [1-9a-zA-Z_]

Assertions:
  ^    matches beginning of line (with /m option)
  $    matches end of line       (with /m option)
  \b   matches at word boundary  (\B for not)

Quantifiers (greedy by default; add ? suffix to make non-greedy):
  ?          0 or 1    (same as {0,1})
  *          0 or more (same as (0,} )
  +          1 or more (same as {1,} )
  {n}        exactly n
  {min,}     at least min
  {min,max}  between min and max inclusive

Back references:
  \2  refers to second parenthesized subexpression
```

An excellent book on regular expressions:

Mastering Regular Expressions (2nd edition) by Jeffrey Friedl (O'Reilly, 2002)

Examples

Find fns that contain aa followed by bb on the same line
```
'←' fnreplx '/aa.⋆bb/'
```

Use the /s regex option to have "." match even newline characters; e.g.,
Find fns that contain aa followed by bb, not necessarily on the same line
```
'←' fnreplx '/aa.⋆bb/s'
```

Find fns that contain aa and bb, in either order, anywhere in the function
```
'←' fnreplx '/aa.⋆bb|bb.⋆aa/s'
```

Use the /i regex option for case-insensitive matching; e.g.,
```
fnreplx '/⎕io/i'
```

For "syntactic" search, use the regex "word boundary" assertion \b
```
'-foo' fnreplx '/\bfoo\b/'
```
This works pretty well for APL, but... Note that the regex engine does not consider ∆ and ⍙ to be letters. So that would match the "foo" in "∆foo".

Match all APL identifier names:
```
'-foo' fnreplx '/[⎕A-Za-z∆⍙][A-Za-z∆⍙0-9_]⋆/'
```

Match integers: basically it's
```
fnreplx '/¯?\d+/'
```
but we need to pay attention to what comes before and after; this is close
```
fnreplx '/(?<![\.\w])¯?\d+(?![\.\w])/'
```
But since \w does not include the deltas, we should really use
```
fnreplx '/(?<![\.a-zA-Z0-9_∆⍙])¯?\d+(?![\.a-zA-Z0-9_∆⍙])/'
```
The (?...) things are lookbehind and lookahead assertions

Delete the leading blank from lines that are indented by exactly one space
```
fnreplx '/^ (?! )//m'
```

Parenthesized subpatterns in the target string may be referred to with
$1 $2 $3 in the replacement string; e.g.,
```
fnreplx ';\b(able|baker)\b;<I>$1</I>;'
```
wraps <I> and </I> html tags around words able and baker

Change a ⎕ss b to b⍷a where a and b are variable names
```
 fnreplx '/([⎕A-Za-z∆⍙][A-Za-z∆⍙0-9_]⋆) ⋆⎕ss +([⎕A-Za-z∆⍙][A-Za-z∆⍙0-9_]⋆)/$2⍷$1/i'
```

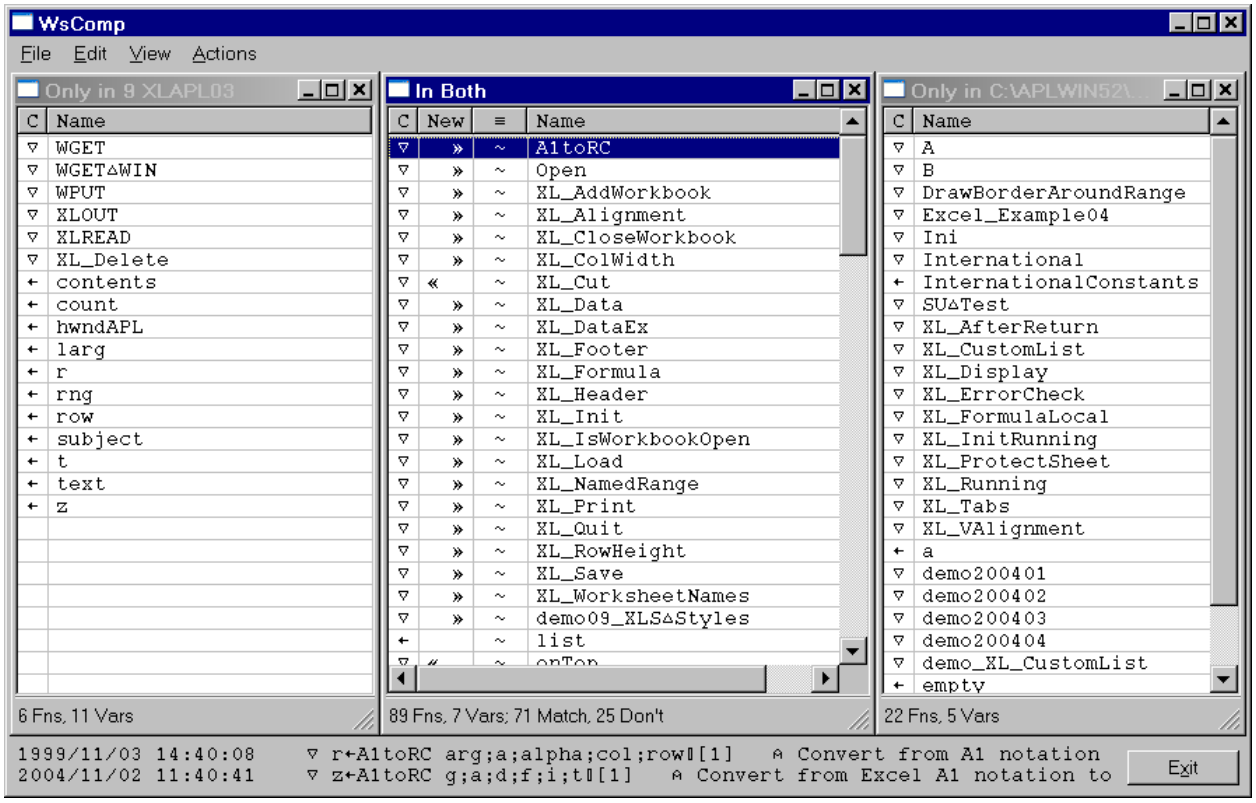
Break lines at diamonds (remember, newline to the Unix crowd is ⎕TCLF)
```
'foo' fnreplx '/\s◇\s/',⎕tclf,'/'
```

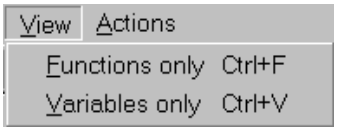
Find bare branches
```
fnreplx '/→[ ]⋆(⍝|◇|$)/m'
```

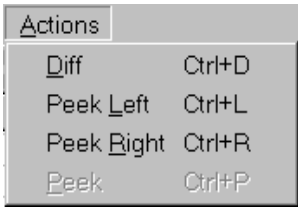
\1 \2 \3 ... \9 may be used for "back references" --
to refer to strings matched in parenthesized subexpressions earlier in the target
Find repeated words
```
'-' fnreplx '/\b([A-Za-z]+) +(\1)\b/'
```

WSCOMP: Workspace Compare

```
]wscomp 9 xlapl03 ; C:\APLWin52\2004 Conf Disk\xlapl04
```







The "C" columns indicate the class of object. The "New" column points to the side with the newer function timestamp. The "≡" column indicates whether the objects are identical: "~" means different and "≡" means the same.

You may double-click and object in the center pane to start a]DIFF comparison.

Try `]wscomp /regserver` if you have not already registered aplw as a server.

Shadowing can be avoided by using a saved ws rather than the active ws.

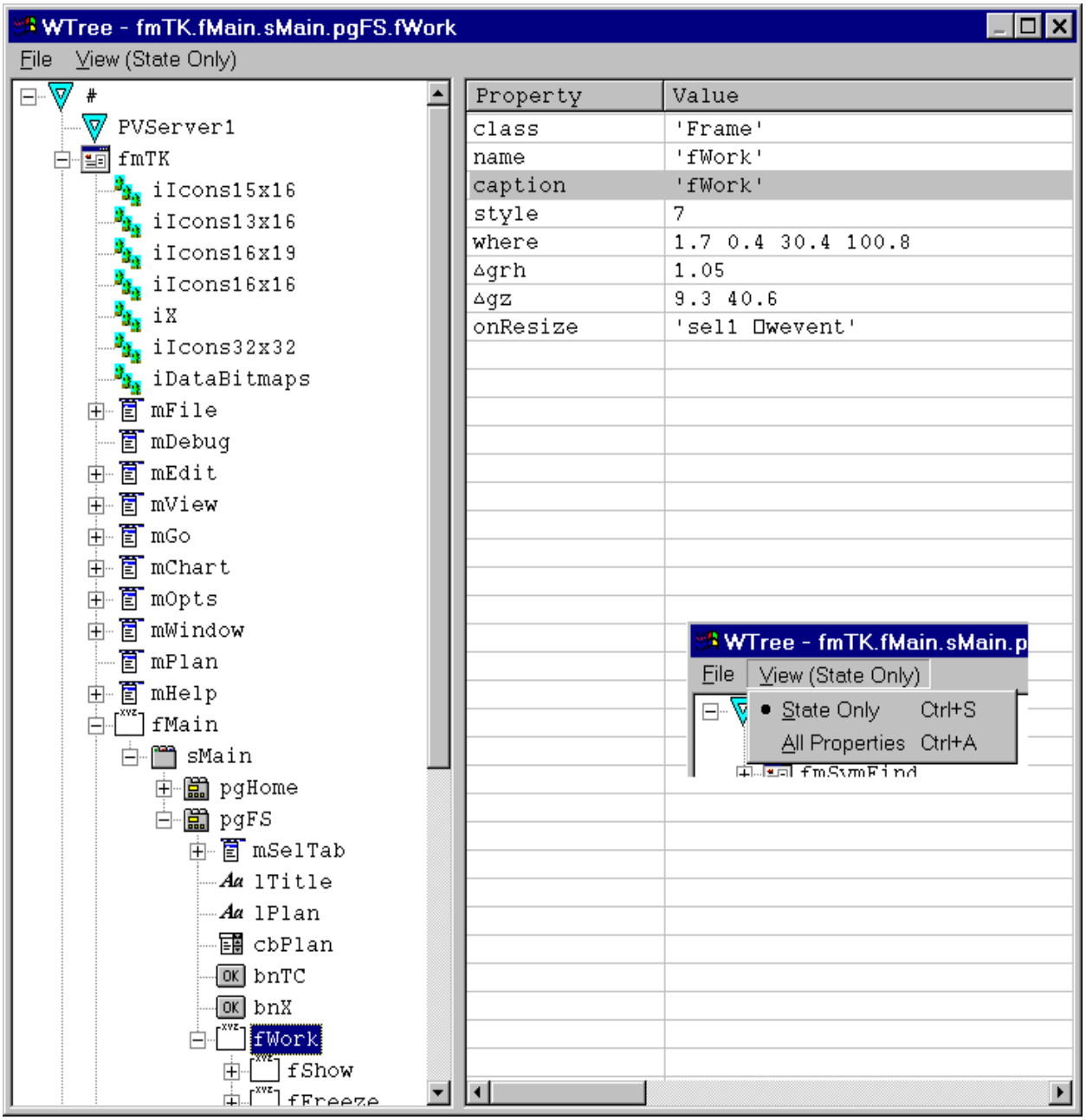
WTREE with argument

An APL2000 tool, enhanced to allow an argument to specify the object to start with.

```
      ⍝ ]WTREE fmFoo.fData.bnCancel  ⍝ start with specified object
      ⍝ ]WTREE .                     ⍝ start with ⎕wself
```

Also added View menu, and remembers window size and position.

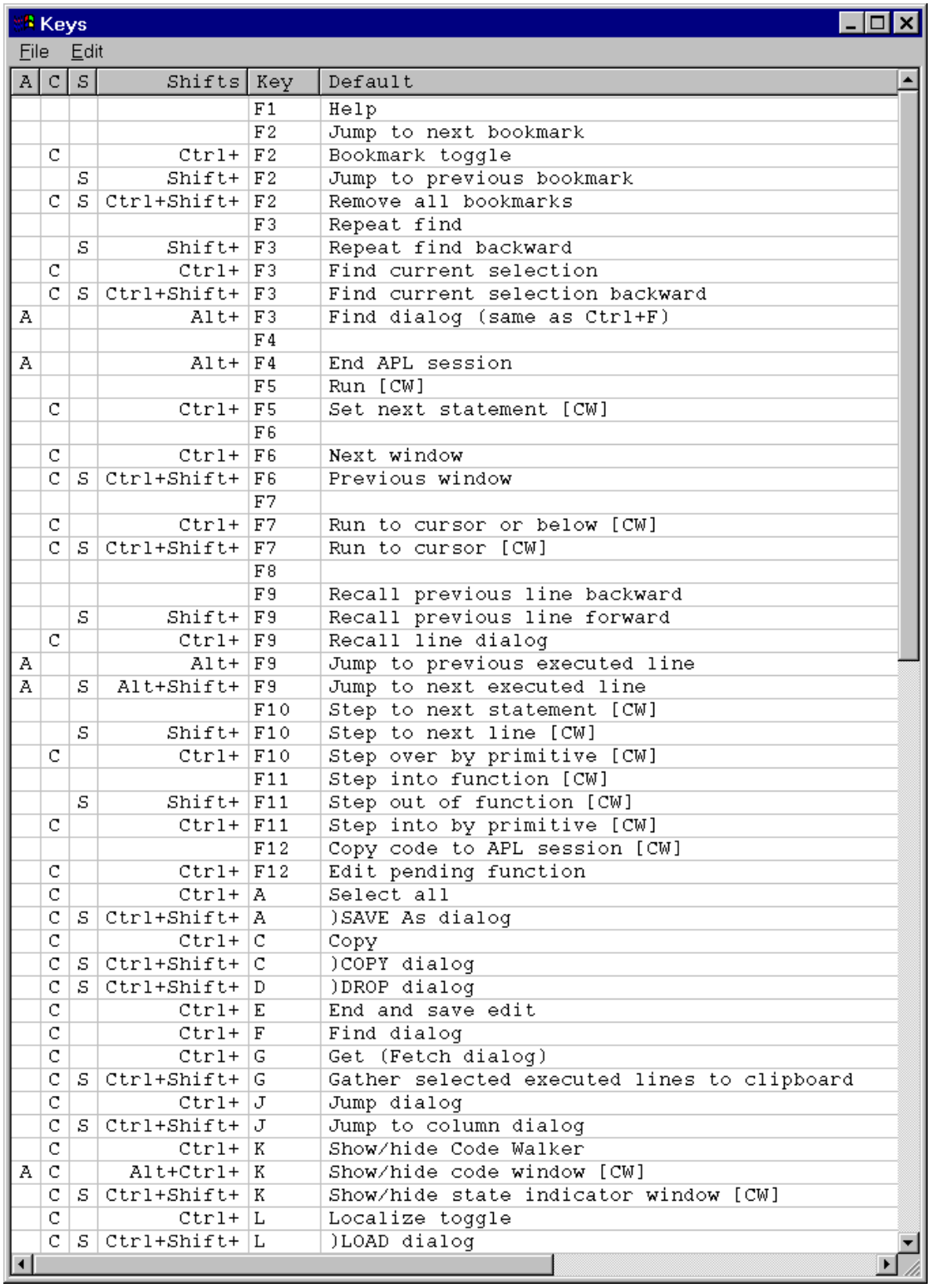
```
      ]wtree fmTK.fMain.sMain.pgFS.fWork
```



KEYS: APL+Win Shortcut Keys

Can't remember an APL+Win shortcut key...?

```
]keys
```



Keys

File Edit

A	C	S	Shifts	Key	Default
				F1	Help
				F2	Jump to next bookmark
	C		Ctrl+	F2	Bookmark toggle
		S	Shift+	F2	Jump to previous bookmark
	C	S	Ctrl+Shift+	F2	Remove all bookmarks
				F3	Repeat find
		S	Shift+	F3	Repeat find backward
	C		Ctrl+	F3	Find current selection
	C	S	Ctrl+Shift+	F3	Find current selection backward
A			Alt+	F3	Find dialog (same as Ctrl+F)
				F4	
A			Alt+	F4	End APL session
				F5	Run [CW]
	C		Ctrl+	F5	Set next statement [CW]
				F6	
	C		Ctrl+	F6	Next window
	C	S	Ctrl+Shift+	F6	Previous window
				F7	
	C		Ctrl+	F7	Run to cursor or below [CW]
	C	S	Ctrl+Shift+	F7	Run to cursor [CW]
				F8	
				F9	Recall previous line backward
		S	Shift+	F9	Recall previous line forward
	C		Ctrl+	F9	Recall line dialog
A			Alt+	F9	Jump to previous executed line
A		S	Alt+Shift+	F9	Jump to next executed line
				F10	Step to next statement [CW]
		S	Shift+	F10	Step to next line [CW]
	C		Ctrl+	F10	Step over by primitive [CW]
				F11	Step into function [CW]
		S	Shift+	F11	Step out of function [CW]
	C		Ctrl+	F11	Step into by primitive [CW]
				F12	Copy code to APL session [CW]
	C		Ctrl+	F12	Edit pending function
	C		Ctrl+	A	Select all
	C	S	Ctrl+Shift+	A	)SAVE As dialog
	C		Ctrl+	C	Copy
	C	S	Ctrl+Shift+	C	)COPY dialog
	C	S	Ctrl+Shift+	D	)DROP dialog
	C		Ctrl+	E	End and save edit
	C		Ctrl+	F	Find dialog
	C		Ctrl+	G	Get (Fetch dialog)
	C	S	Ctrl+Shift+	G	Gather selected executed lines to clipboard
	C		Ctrl+	J	Jump dialog
	C	S	Ctrl+Shift+	J	Jump to column dialog
	C		Ctrl+	K	Show/hide Code Walker
A	C		Alt+Ctrl+	K	Show/hide code window [CW]
	C	S	Ctrl+Shift+	K	Show/hide state indicator window [CW]
	C		Ctrl+	L	Localize toggle
	C	S	Ctrl+Shift+	L	)LOAD dialog

Keys

File Edit

A	C	S	Shifts	Key	Default
	C		Ctrl+	M	Match paren/quote/bracket
	C	S	Ctrl+Shift+	M	Match paren/quote/bracket and select
	C		Ctrl+	N	New edit
	C		Ctrl+	O	Open edit dialog
	C	S	Ctrl+Shift+	O	Open edit session on object at caret
	C		Ctrl+	P	Print dialog
	C	S	Ctrl+Shift+	P	)PCOPY dialog
	C		Ctrl+	Q	Quit edit
	C		Ctrl+	R	Replace dialog
	C	S	Ctrl+Shift+	R	)CLEAR dialog
	C		Ctrl+	S	Save edit
	C		Ctrl+	T	Toggle edit and session
	C		Ctrl+	V	Paste
	C		Ctrl+	W	Watchpoints dialog
	C		Ctrl+	X	Cut
	C		Ctrl+	Z	Undo
	C		Ctrl+	↓	Scroll down without moving caret
	C	S	Ctrl+Shift+	↓	Move the cursor line to the bottom of the page
	C		Ctrl+	↑	Scroll up without moving caret
	C	S	Ctrl+Shift+	↑	Move the cursor line to the top of the page
	C	S	Ctrl+Shift+	End	Select from caret to end
A		S	Alt+Shift+	End	Erase from caret to end of session log
	C	S	Ctrl+Shift+	Home	Select from caret to beginning
	C		Ctrl+	PgDn	Scroll left
	C		Ctrl+	PgUp	Scroll right
				Tab	Insert tab indent
		S	Shift+	Tab	Remove tab indent
	C		Ctrl+	Tab	Next window
	C	S	Ctrl+Shift+	Tab	Previous window
A			Alt+	Tab	Next application [XX]
	C		Ctrl+	Space	Insert space indent
	C	S	Ctrl+Shift+	Space	Remove space indent
A			Alt+	Bksp	Undo (same as Ctrl+Z)
		S	Shift+	Del	Cut (same as Ctrl+X)
		S	Shift+	Ins	Paste (same as Ctrl+V)
	C		Ctrl+	Ins	Copy (same as Ctrl+C)
	C		Ctrl+	Enter	Insert line in session (without execution)
	C	S	Ctrl+Shift+	Enter	Move the cursor line to the middle of the page
	C		Ctrl+	/	Insert lamp
	C	S	Ctrl+Shift+	/	Remove lamp
	C		Ctrl+	+	Increase font size
	C	S	Ctrl+Shift+	+	Decrease font size
	C		Ctrl+	.	Stop toggle
	C		Ctrl+	,	Trace toggle
	C	S	Ctrl+Shift+	.	Stop remove
	C	S	Ctrl+Shift+	,	Trace remove
	C		Ctrl+	Num *	Top of stack [CW]
	C		Ctrl+	Num -	Up one stack level [CW]
	C		Ctrl+	Num +	Down one stack level [CW]
	C		Ctrl+	Num /	Bottom of stack [CW]

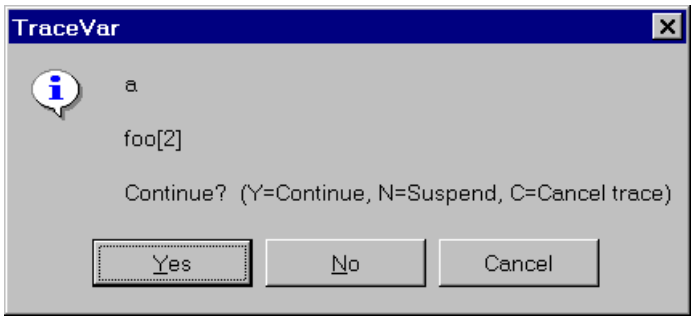
TRACEVAR: Trace Variable Changes

Each time the variable is changed, two diagnostic lines are displayed. The first shows the SI stack, and the second shows the variable's new value.

```
    ∇ foo;a;i
[1]   :for i :in ⍳3
[2]      a←i
[3]   :end
    ∇
```

```
      ]tracevar a
      foo
→→→ TraceVar: foo[2]
←←← TraceVar: a←1
→→→ TraceVar: foo[2]
←←← TraceVar: a←2
→→→ TraceVar: foo[2]
←←← TraceVar: a←3
```

```
      ]tracevar a /alert
      foo
→→→ TraceVar: foo[2]
←←← TraceVar: a←1
```

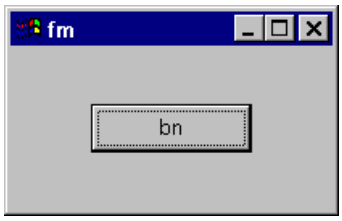



```
      ]tracevar /off
```

```
      ]tracevar ?
]TraceVar foo ...          ⍝ trace changes to the named variable(s)
]TraceVar foo:2⊃foo        ⍝ display 2⊃foo when foo changes; just one var
]TraceVar ... /STOP        ⍝ sets stop on first change; suspends before next
]TraceVar ... /ALERT       ⍝ display MessageBox after change report

]TraceVar foo ... /OFF     ⍝ turn off tracing of the named variable(s)
]TraceVar /OFF             ⍝ turn off all tracing
```

TRACEOBJ: Trace Object Events and Actions

```
      ⎕wself←'fm' ⎕wi 'Create' 'Form' ('size' 5 20)
      ⎕wself←'.bn' ⎕wi 'Create' 'Button'
```




```
      ]traceobj
TraceObj now tracing 31 events on fm.bn
TraceObj now tracing actions on fm.bn
      ⍝ cause an Action
      ⎕wi 'where' 1 1
∘∘∘ TraceObj: fm.bn Action ('where' 1 1) <[fm.bn;where]
      ⍝ Alt+Tab to form
∘∘∘ TraceObj: fm.bn Focus (0)
∘∘∘ TraceObj: fm.bn Paint (0)
      ⍝ use mouse to click button and then click X to close form
      ⍝ trace display is:  ∘∘∘ TraceObj: ⎕wself ⎕wevent (⎕warg) ⎕si
∘∘∘ TraceObj: fm.bn MouseEnter (1.1 9 0 0 0)
∘∘∘ TraceObj: fm.bn MouseMove (1.1 9 0 0 0)
∘∘∘ TraceObj: fm.bn MouseMove (0.8 8.8 0 0 0)
∘∘∘ TraceObj: fm.bn MouseDown (0.4 8.4 1 1 0)
∘∘∘ TraceObj: fm.bn MouseUp (0.4 8.4 1 0 0)
∘∘∘ TraceObj: fm.bn Click (0)
∘∘∘ TraceObj: fm.bn MouseMove (0.4 8.4 0 0 0)
∘∘∘ TraceObj: fm.bn MouseMove (0.4 8.2 0 0 0)
∘∘∘ TraceObj: fm.bn MouseMove (0.4 9 0 0 0)
∘∘∘ TraceObj: fm.bn MouseLeave (0)
∘∘∘ TraceObj: fm.bn Unfocus (0)
∘∘∘ TraceObj: fm.bn Destroy (0)
      ⍝ delete form (which causes button to be deleted)
      'fm' ⎕wi 'Delete'
∘∘∘ TraceObj: fm.bn Delete (0) <[fm;Delete]

      ]traceobj ?

Trace activity on an APL GUI object
Syntax:  ]TraceObj [obj] [/A] [/E] [/OFF] [/X=...] [/I=...]
If object not specified, ⎕wself is used
Use /A to trace just actions or /E to trace just events; default is both
/X excludes (all but) actions/events matching substring -- e.g., /X=mouse
/I includes (only)    actions and events matching substring
Any trace on an object replaces any previous trace
Trace display is:  ∘∘∘ TraceObj: ⎕wself ⎕wevent (⎕warg) ⎕si

For example, after:          'fm' ⎕wi 'Create' 'Form' ('onResize' 'foo')

/A traces property changes:  'fm' ⎕wi 'size' 10 20
               and methods:  'fm' ⎕wi 'Show'
but it does NOT trace events (Resize is not traced if user resizes fm)

/E traces events such as Resize triggered if user resizes form
```

FNREPL: Function Search and Replace

```
      ]fnrepl ?

Find (and optionally replace) strings in functions
```

Syntax:

```
  ]fnrepl target                 ⍝ Search (if target begins with a letter)
  ]fnrepl target/options         ⍝ Ditto, with options

Any non-alpha delimiter character may be used in place of the slashes
shown below

  ]fnrepl /target                ⍝ Search (exactly 1 delimiter)
  ]fnrepl /target/               ⍝ Search (exactly 2 delimiters)
  ]fnrepl /target/options        ⍝ Ditto, with options
  ]fnrepl /target/replacement/   ⍝ Replace (exactly 3 delimiters)
  ]fnrepl /target/rep/options    ⍝ Ditto, with options

Multiple targets (not allowed in replace mode):

  ]fnrepl /t1/∧/t2/options   ⍝ Find only lines with both matches
  ]fnrepl /t1/∨/t2/          ⍝ Find lines with either match
  ]fnrepl /t1/∨\t2\          ⍝ Delimiters may be different
  ]fnrepl /t1/&~/t2/         ⍝ Logical not; & and | are also permitted
  ]fnrepl /t1/>/t2/          ⍝ Can use any of <≤=≥>≠∨∧⍲⍱&| and ~
  ]fnrepl /t1/∧~/t2/∨/t3/    ⍝ Evaluated as APL would ("right to left")

Default is to perform case-sensitive, syntactic (name) search,
on all global functions
```

Options:

```
To loosen match criteria:

  /⌈       ⍝ Non-syntactic search (Alt+S for non-Syntactic)
  /∩       ⍝ Ignore case (upper/lower/underscore) (Alt+C for Case)

To limit functions searched:

  /f1 f2   ⍝ Only look in named functions
  /f⋆      ⍝ Fn names with wildcards (foo⋆ ⋆old fm⋆def etc.)
  /~f1 f2  ⍝ Look in all but named functions

To limit search scope (using all three is equivalent to using none):

  /⍝       ⍝ Find only in comments
  /'       ⍝ Find only in character constants (or /")
  /⍺       ⍝ Find only in APL (not comments/quotes)

To expand search scope:

  /▽       ⍝ Find strings anywhere in a function -- not necessarily on
           ⍝ the same line.  Only useful with multiple targets.  Doesn't
           ⍝ change behavior of /t1/∨/t2/.  Display from /t1/∧/t2/▽ is
           ⍝ similar to /t1/∨/t2/ but only shows functions that contain
           ⍝ at least one hit on both strings.
```

```
Other:

  /⍴       ⍝ Auto reply Yes to the replacement confirmation prompt
  /?       ⍝ Prompt before each replacement
  /←       ⍝ Suppress display; return fn names

Nice features:

   Both single- and double-quotes are handled (thanks, Zark!)
   Searches for system names (like ⎕io) are automatically case-insensitive

Outstanding issues:

   Search for '¯2' should not match '1E¯2'
   Search for 'if' should not match ':if'
   Search for ':if' should be automatically case-insensitive
```

I recently implemented the multiple target feature, and I like it. My `fnreplx` which uses regular expressions is much more powerful, but I have trouble remembering the regex syntax. This `/t1/∧/t2/` syntax solves about 95% of my problems -- and I can remember it!

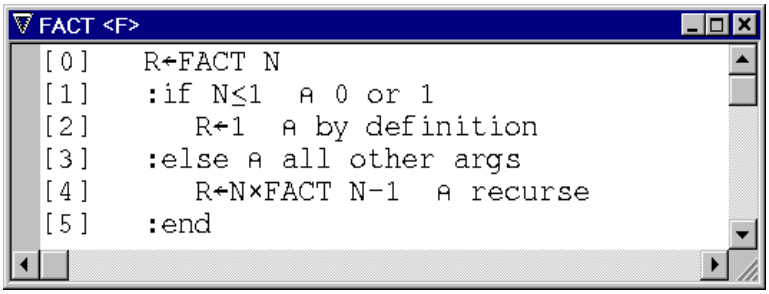
I also like the `/t1/∧/t2/∇` feature. I often look for a string and get a flood of hits. Generally I only care about the hits in functions where a second string is present -- though not necessarily on the same line.

There is also `]scan` which is the same as `]fnrepl` except:

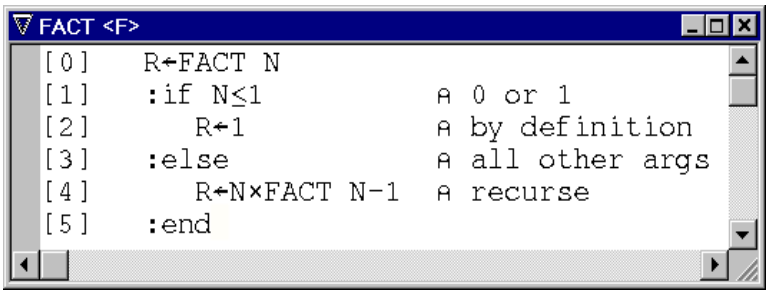
- Search only (no replace)
- Case-insensitive
- Non-syntactic

Align and WordWrap Comments

Would you like to go from this...



... to this:



Or, from this...

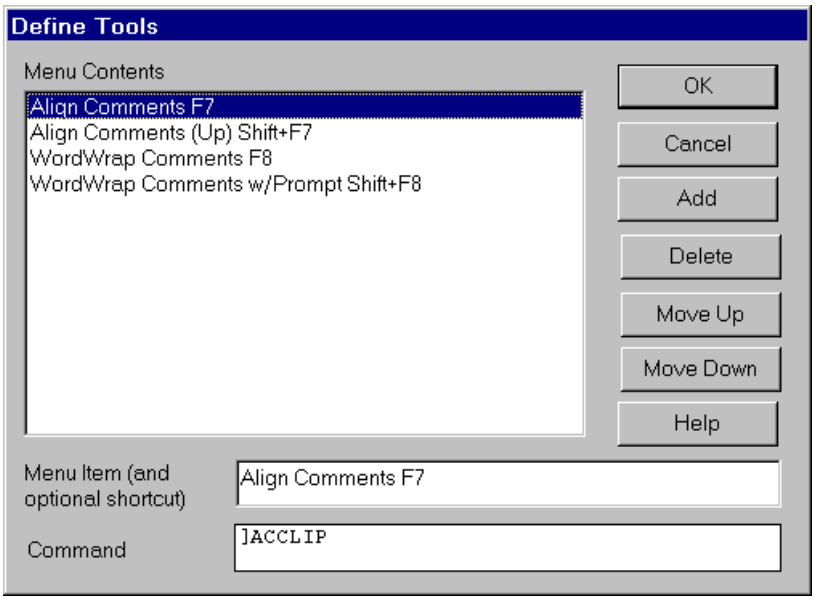
```
[154]
[155] ⍝ 403 Forbidden: The server understood the request,
[156] ⍝ but is refusing to fulfill it.
[157] ⍝ Authorization will not help and the request should not be repeated.
[158] ⍝ If the request method
[159] ⍝ was not HEAD and the server wishes to make public why the request
[160] ⍝ has not been fulfilled,
[161] ⍝ it should describe the reason for the refusal in the entity body.
[162] ⍝ This status code is
[163] ⍝ commonly used when the server does not wish to reveal exactly why
[164] ⍝ the request has been
[165] ⍝ refused, or when no other response is applicable.
[166]
```

... to this:

```
[154]
[155] ⍝ 403 Forbidden: The server understood the request, but is refusing to fulfill
[156] ⍝ it.  Authorization will not help and the request should not be repeated.  If
[157] ⍝ the request method was not HEAD and the server wishes to make public why the
[158] ⍝ request has not been fulfilled, it should describe the reason for the refusal
[159] ⍝ in the entity body.  This status code is commonly used when the server does
[160] ⍝ not wish to reveal exactly why the request has been refused, or when no other
[161] ⍝ response is applicable.
[162]
```

Define Tools

- Select Options|Tools from the APL+Win session menu...



- ...and define four tools:

Menu Item	Command
Align Comments F7	`]ACCLIP`
Align Comments (Up) Shift+F7	`]ACCLIP /UP`
WordWrap Comments F8	`]WWCLIP`
WordWrap Comments w/Prompt Shift+F8	`]WWCLIP /W=?`

- Select Options|Save Settings Now from the APL+Win session menu.

Digression

Another way to assign these commands to function keys is:

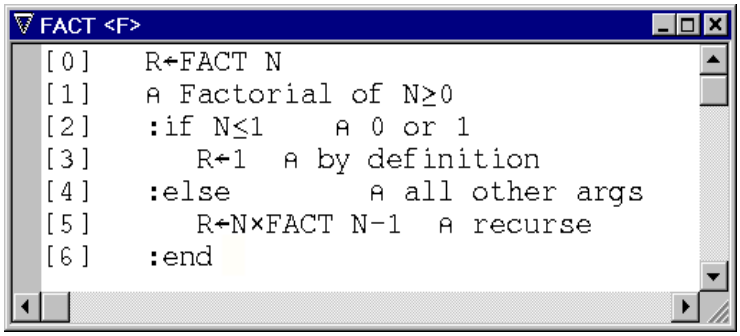
```
'#' ⎕WI 'PF'     118 ']ACCLIP'        ⍝ F7
'#' ⎕WI 'PF' 100118 ']ACCLIP /UP'     ⍝ Shift+F7
'#' ⎕WI 'PF'     119 ']WWCLIP'        ⍝ F8
'#' ⎕WI 'PF' 100119 ']WWCLIP /W=?'    ⍝ Shift+F8
```

See APL+Win Windows Interface help for "Keyboard Events" for Virtual Key Codes.
Function keys F1-F24 are 112-135; add Shift=100000, Ctrl=200000, Alt=400000.

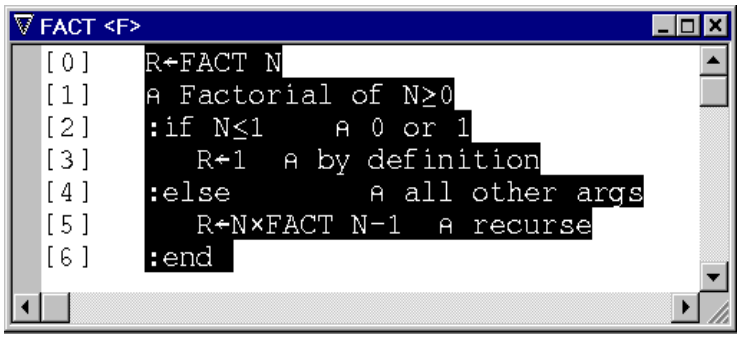
But these are not saved in your APLW.INI file.

Align Comments

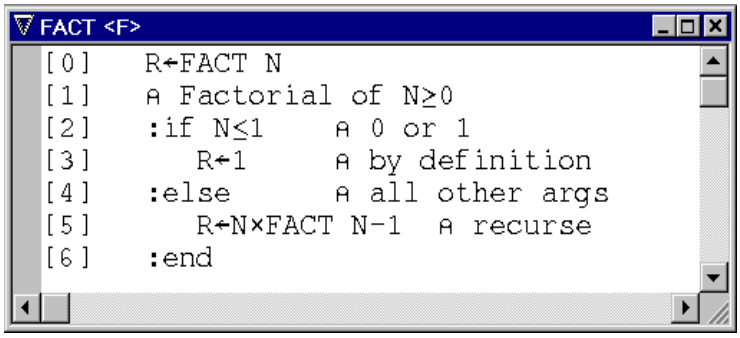
- `)edit FACT`



- Ctrl+A



- F7



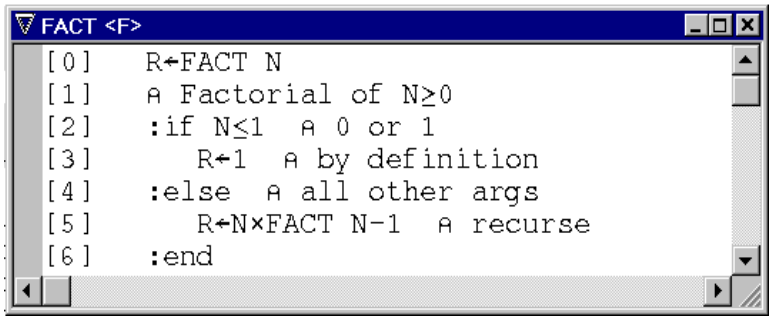
Whole-line comments are ignored.

Working from the top down, the first partial-line comment establishes the desired lamp position.

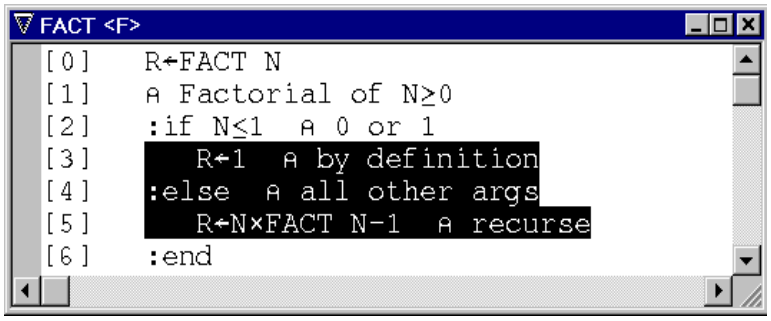
If you don't like what happened, just use Edit|Undo (Ctrl+Z).

Align Comments (Up)

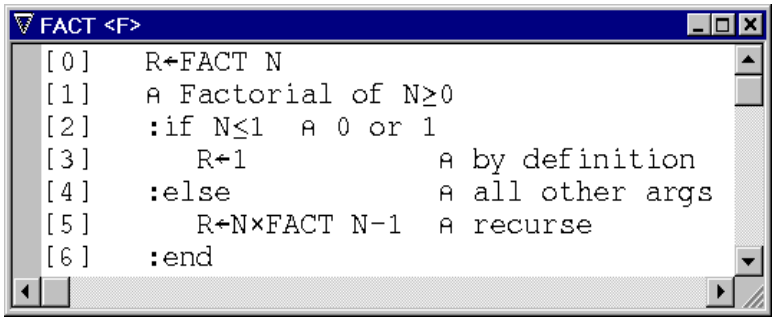
- `)edit FACT`



- Select just three lines



- Shift+F7



Only the selected lines are considered.

Working from the bottom up, the first partial-line comment establishes the desired lamp position.

If you don't like what happened, just use Edit|Undo (Ctrl+Z).

How Does This Work?

The way that APL+Win "Tools" work is perfect for this. When you press F7, the user command `]ACCLIP` is executed. Even though your focus is on the function editor window, the user command is executed in the session window -- not typed into the function as a keyboard macro program (such as Macro Express) would do. It's also nice that what's executed does not *appear* in the session (you may have seen `⎕INBUF` used when you *want* something to appear in the session).

Once we get our hands on the selected text, the rest is easy. The trick is getting (and replacing) the selected text.

When the user command executes, it copies the selected text into the Windows clipboard. It does this by using the Windows keybd_event function to simulate typing Ctrl+C. This has to be done at a low level:

1. press Ctrl
2. press C
3. release C
4. release Ctrl

`]ACCLIP` then retrieves the clipboard text using a series of Windows functions such as GetClipboardData, processes the text, and writes it back into the clipboard with SetClipboardData. (See utility functions `ClipGet` and `ClipCopy`.)

Finally we paste the modified text from the clipboard back into the function with more simulated keystrokes that perform Ctrl+V.

A nice aspect of using Paste from the clipboard: it's undoable.

Digression

There is an alternative to using the Windows keybd_event function to trigger things like Copy and Paste: you can send WM_COMMAND messages to the APL session. For example:

```
h←'#' ⎕wi 'hwndmain'  ⍝ session handle
c←57634  ⍝ ID_EDIT_COPY = 0xE122 = 57634
⎕wcall 'SendMessage' h 'WM_COMMAND' c 0
```

This uses a standard Windows menu item which would work for many Windows applications. If you search MSDN for ID_EDIT_COPY you will find lots of others like File New/Open/Close/Save and Edit Clear/Cut/Copy/Paste.

WordWrap Comments

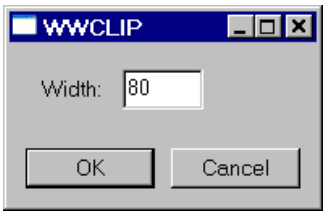
- `)edit foo`

```
[154]
[155] ⍝ 403 Forbidden: The server understood the request,
[156] ⍝ but is refusing to fulfill it.
[157] ⍝ Authorization will not help and the request should not be repeated.
[158] ⍝ If the request method
[159] ⍝ was not HEAD and the server wishes to make public why the request
[160] ⍝ has not been fulfilled,
[161] ⍝ it should describe the reason for the refusal in the entity body.
[162] ⍝ This status code is
[163] ⍝ commonly used when the server does not wish to reveal exactly why
[164] ⍝ the request has been
[165] ⍝ refused, or when no other response is applicable.
[166]
```

- Select lines [154] or [155] through [165] or [166]
- Press F8

```
[154]
[155] ⍝ 403 Forbidden: The server understood the request, but is refusing to fulfill
[156] ⍝ it.  Authorization will not help and the request should not be repeated.  If
[157] ⍝ the request method was not HEAD and the server wishes to make public why the
[158] ⍝ request has not been fulfilled, it should describe the reason for the refusal
[159] ⍝ in the entity body.  This status code is commonly used when the server does
[160] ⍝ not wish to reveal exactly why the request has been refused, or when no other
[161] ⍝ response is applicable.
[162]
```

- Press Shift+F8 if you want to specify a different wordwrap right margin.



This value is remembered in your APLW.INI file.

Works With and Without Lamps

If you're editing a character vector or matrix of text, you can use `]WWCLIP` for a simple paragraph. (Actually, this works in functions too -- if the first non-blank character in the selection is a lamp, then lamps are used as a prefix.)

Preserves First-Line and Hanging Indents

Indentation of the first line (that is, blanks between the lamp and the text) is preserved.

Indentation on the second line is used to set a hanging indent, which will be used on all subsequent lines.

Preserves Indented Lamps

If the selected block of comments is consistently indented (that is, if the lamps are indented), WWCLIP will preserve that indentation.

How Does This Work?

`]WWCLIP` uses the same clipboard techniques as `]ACCLIP`.

There is nothing particularly fancy about the paragraph flow logic. This tool does just one paragraph at a time, and it will preserve indents. That is adequate for my purposes. I'm sure you have some more elaborate code that you could incorporate.

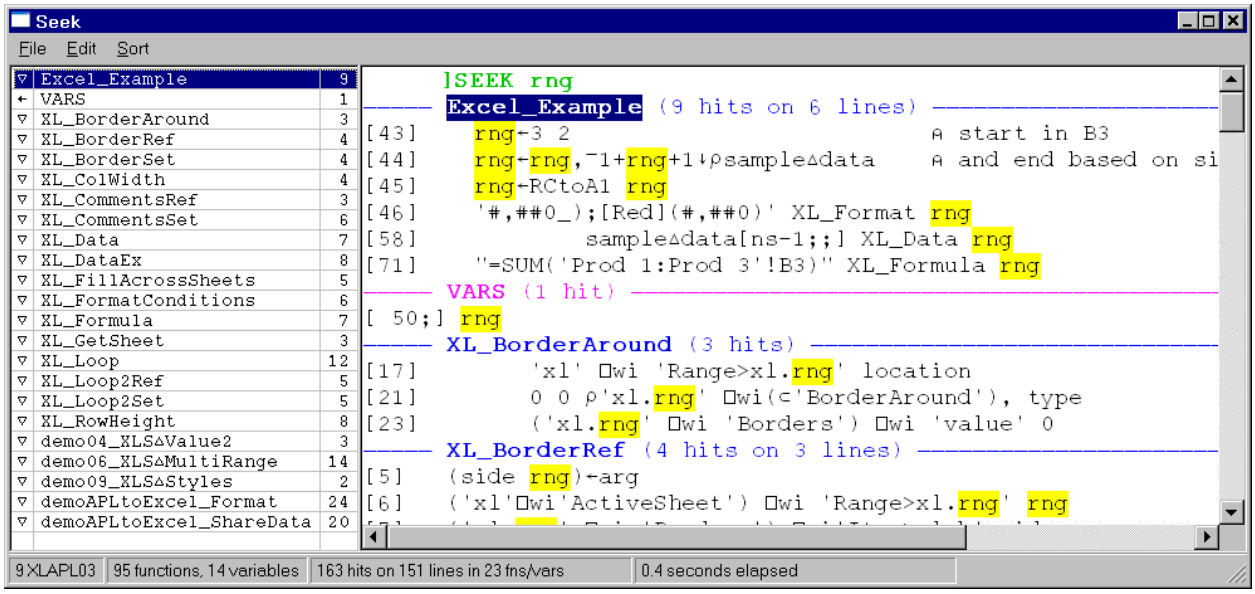
The idea of this presentation is to give you something that works, and more importantly, to spark ideas for more elaborate or completely different tools of your own.

SEEK: Search active workspace with GUI display

Color highlighting makes it easier to see hits -- particularly multiple occurrences on one line (e.g., `Excel_Example[44]`).

Note that global variables are searched too (e.g. `VARS`).

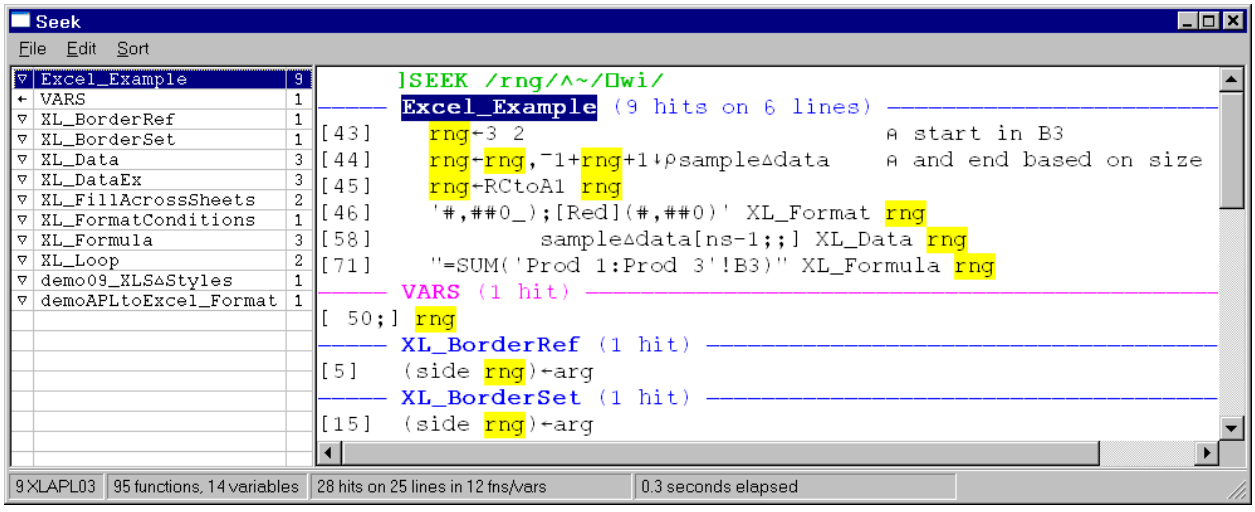
```
]seek rng
```



You can click on the left pane to scroll the right pane. You can double-click on the right pane to start)EDIT on the object.

See `]seek ?` for help on complex arguments. For instance, this means to search for `rng` *and not* `⎕wi`:

```
]seek /rng/^~/⎕wi/
```

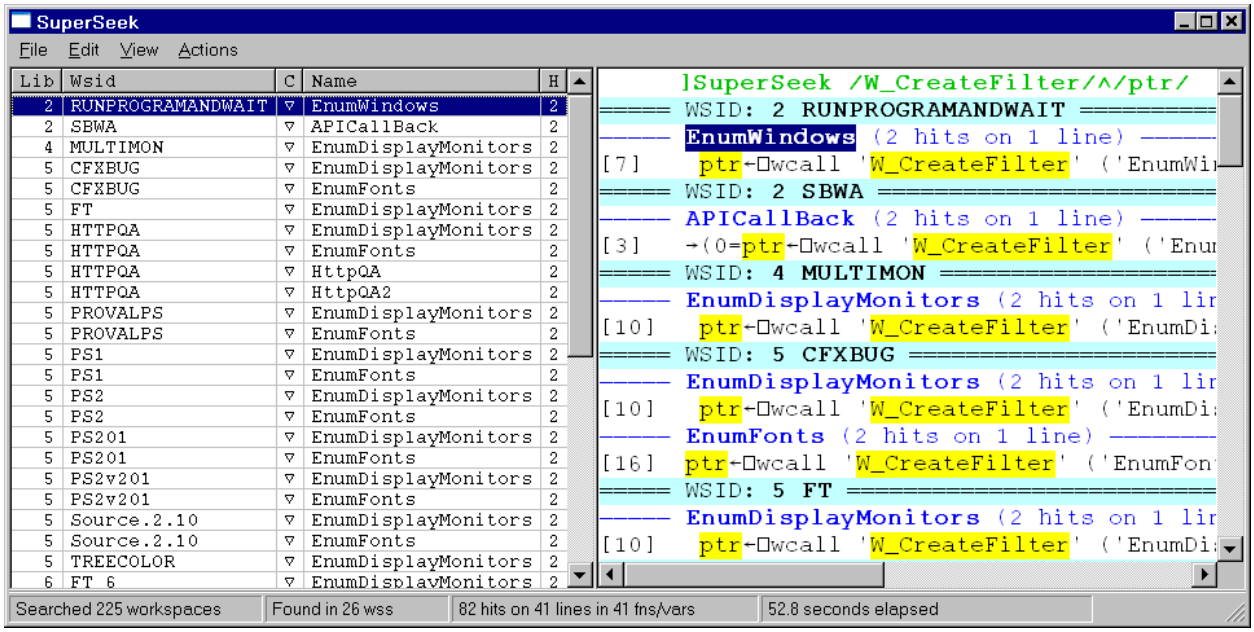


SUPERSEEK: Search across workspaces

Sometimes you need to search the attic and the basement to find what you want.

This tool searches though all the workspaces in all your)LIBS.

```
]superseek /W_CreateFilter/^/ptr/
```



Options allow you to restrict the libraries searched -- or to scan your entire drive for workspaces!

You may double-click on the right pane to]PEEK at an object.

SEEK and SUPERSEEK both use the “APLNext” Unicode font, which may be downloaded from www.rexswain.com/tooltalk.html.