

# Tool Talk 4

Rex Swain  
Independent Consultant  
8 South Street  
Washington, CT 06793  
Tel 860-868-0131  
Fax 860-868-9970  
rex@rexswain.com

<http://www.rexswain.com>

APL2000 User Conference

November 2006

## Notes

- To add UCMSDREX.SF to your user command files path:

```
]UFILE [1.5] UCMSDREX
```

If you already have UCMSDREX.SF from a previous conference, just replace it with this one; the tools from 2003, 2004, and 2005 are all here too.

- To learn about tools from previous conferences, see:

```
UCMSDREX03.PDF (2003)
```

```
UCMSDREX04.PDF (2004)
```

```
UCMSDREX05.PDF (2005)
```

- These tools work with both )EVLEVEL 1 and )EVLEVEL 2
- Help:

```
]foo ?
```

## Enhancements to Previous Tools

- ]DIFF -- Automatically prompts to save changes made in Araxis Merge
- ]PEM Edit?style -- Displays help for property

## 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.

## Function Search and Replace

lfnrepl ?

Find (and optionally replace) strings in functions

### Syntax:

```
lfnrepl target           A Search (if target begins with a letter)
lfnrepl target/options  A Ditto, with options
```

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

```
lfnrepl /target         A Search (exactly 1 delimiter)
lfnrepl /target/       A Search (exactly 2 delimiters)
lfnrepl /target/options A Ditto, with options
lfnrepl /target/replacement/ A Replace (exactly 3 delimiters)
lfnrepl /target/rep/options A Ditto, with options
```

Multiple targets (not allowed in replace mode):

```
lfnrepl /t1/^/t2/options A Find only lines with both matches
lfnrepl /t1/v/t2/       A Find lines with either match
lfnrepl /t1/v\t2\      A Delimiters may be different
lfnrepl /t1/&~/t2/     A Logical not; & and | are also permitted
lfnrepl /t1/>/t2/     A Can use any of <=>≠v^v*^&! and ~
lfnrepl /t1/^~/t2/v/t3/ A 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:

```
/Γ      A Non-syntactic search (Alt+S for non-Syntactic)
/∅      A Ignore case (upper/lower/underscore) (Alt+C for Case)
```

To limit functions searched:

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

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

```
/A      A Find only in comments
/'      A Find only in character constants (or /")
/α      A Find only in APL (not comments/quotes)
```

To expand search scope:

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

Other:

```
/p      A Auto reply Yes to the replacement confirmation prompt
/?      A Prompt before each replacement
/←      A 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 (see UCMSDREX04.PDF from the 2004 conference) 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...

```

FACT <F>
[0] R←FACT N
[1] :if N≤1 A 0 or 1
[2]     R←1 A by definition
[3] :else A all other args
[4]     R←N×FACT N-1 A recurse
[5] :end

```

... to this:

```

FACT <F>
[0] R←FACT N
[1] :if N≤1           A 0 or 1
[2]     R←1           A by definition
[3] :else             A all other args
[4]     R←N×FACT N-1 A recurse
[5] :end

```

Or, from this...

```

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

```

... to this:

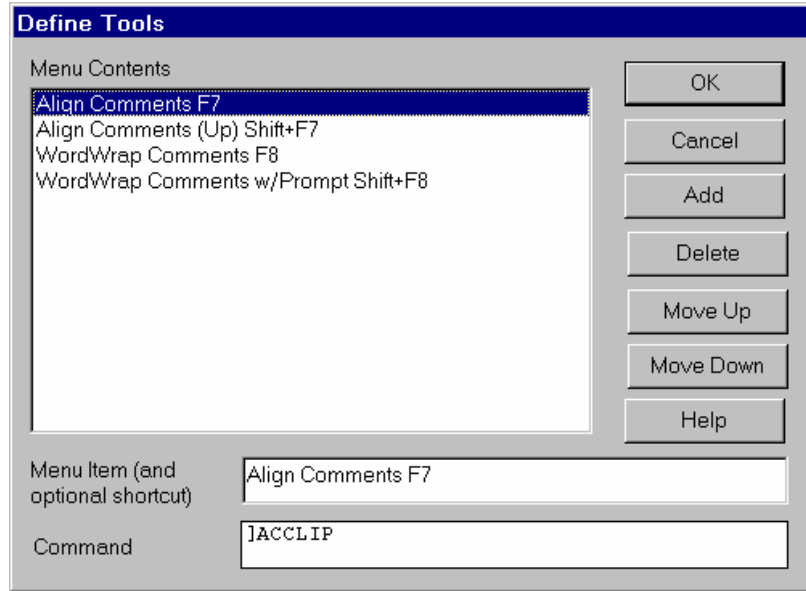
```

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

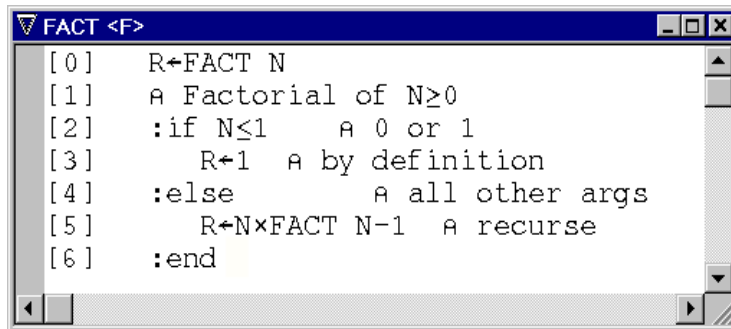
```
'#' □WI 'PF' 118 ']ACCLIP' A F7
'#' □WI 'PF' 100118 ']ACCLIP /UP' A Shift+F7
'#' □WI 'PF' 119 ']WWCLIP' A F8
'#' □WI 'PF' 100119 ']WWCLIP /W=?' A Shift+F8
```

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

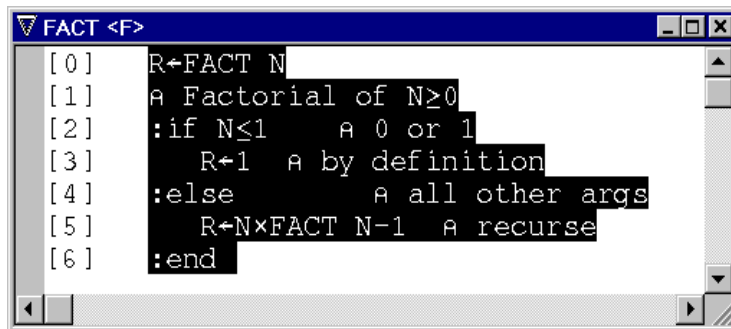


```

[0]  R←FACT N
[1]  A Factorial of N≥0
[2]  :if N≤1    A 0 or 1
[3]      R←1    A by definition
[4]  :else      A all other args
[5]      R←N×FACT N-1  A recurse
[6]  :end

```

- Ctrl+A

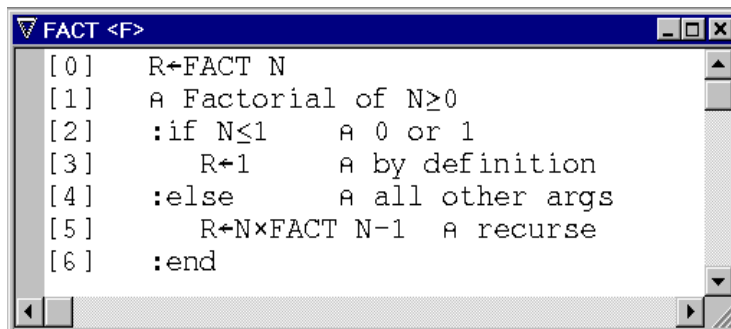


```

[0]  R←FACT N
[1]  A Factorial of N≥0
[2]  :if N≤1    A 0 or 1
[3]      R←1    A by definition
[4]  :else      A all other args
[5]      R←N×FACT N-1  A recurse
[6]  :end

```

- F7



```

[0]  R←FACT N
[1]  A Factorial of N≥0
[2]  :if N≤1    A 0 or 1
[3]      R←1    A by definition
[4]  :else      A all other args
[5]      R←N×FACT N-1  A recurse
[6]  :end

```

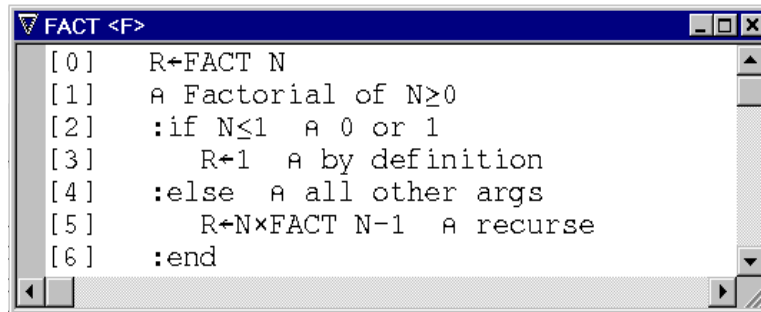
Whole-line comments are ignored.

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

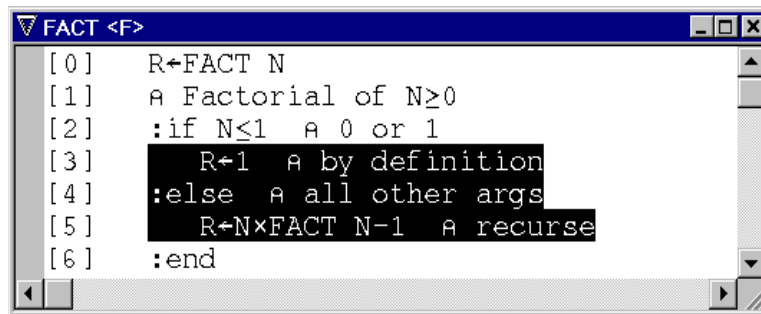


```

[0] R←FACT N
[1] A Factorial of N≥0
[2] :if N≤1 A 0 or 1
[3]   R←1 A by definition
[4] :else A all other args
[5]   R←N×FACT N-1 A recurse
[6] :end

```

- Select just three lines

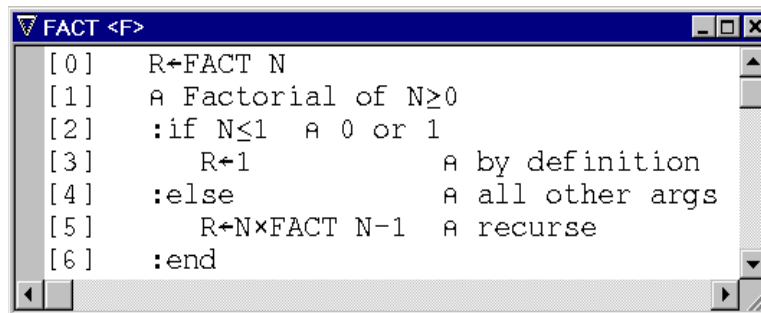


```

[0] R←FACT N
[1] A Factorial of N≥0
[2] :if N≤1 A 0 or 1
[3]   R←1 A by definition
[4] :else A all other args
[5]   R←N×FACT N-1 A recurse
[6] :end

```

- Shift+F7



```

[0] R←FACT N
[1] A Factorial of N≥0
[2] :if N≤1 A 0 or 1
[3]   R←1 A by definition
[4] :else A all other args
[5]   R←N×FACT N-1 A recurse
[6] :end

```

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'  A session handle
c←57634  A 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

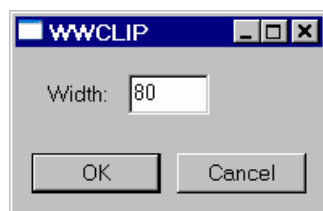
- )edit foo

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

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

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