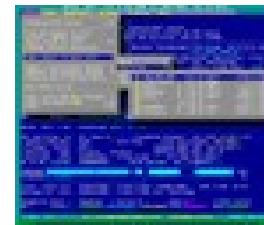# *TxWin enhanced scripting*

Jan van Wijk

TX enhanced scripting language
(as used with DFSee 9.03 and up)

FSYS - *Software*   **TxWin**

Presentation contents

- TxWin and DFSee scripting history

- Design goals for scripting, alternatives

- High level layout of TxScript programs

- Script parameters and variables

- Expressions, available operators

- Built-in functions

- Control structures, program flow

- Expression substitution in commands

FSYS - *Software*   TxWin   DFSee

Who am I ?

Jan van Wijk

- Software Engineer, C, Rexx, Assembly, PHP
- Founded FSYS Software in 2001, developing and supporting DFSee from version 4 to 9.xx
- First OS/2 experience in 1987, developing parts of OS/2 1.0 EE (Query Manager, later DB2)
- Used to be a systems-integration architect at a large bank, 500 servers and 7500 workstations

Home page: http://www.dfsee.com

FSYS - *Software*   TxWin   DFSee

Dfsee scripting history

- Over time, to automate repeating and more complex tasks, several scripting methods have been (and still are!) used with DFSee:

  - BAT/CMD/SHELL scripts, calling DFSee

  - Rexx subcmd environment for OS/2 version

  - Native scripting, being a simple list of DFSee commands, executed sequentially, with simple error handling and parameter substitution

TxScript design goals

- Backwards compatible with existing .DFS scripts as far as possible, allowing re-use

- Direct access to much DFSee internal info, including disk sectors from a script

- Powerful expressions, variables and functions
  To be used from and in the DFSee commandline too

- Conditional and looping control to allow more intelligent and powerful scripts

  Note: For 'DFSee' you can read any hosting program
           that uses the TxScript engine from the TxLib library

Do we need another language ?

- Trying to avoid re-inventing yet another wheel, some alternatives have been considered:

  - Rexx, as used in OS/2 version already
  - Python, clean OO type language
  - Perl, very powerful, hackers heaven :-)
  - PHP, Ruby etc as used in WEB environments

- All had problems with integration in the hosting program (DFSee), availability on all required platforms, or added complexity for install etc.

- Developing a new language is fun, so YES :-)

High level layout of TxScript

- LINE-oriented, but ignores whitespace usage with the lines. Each line is either:

    - A comment line (ignored mostly :-)

    - An interpreter 'pragma' altering its behaviour

    - Program flow statements like IF or WHILE

    - An assignment to one or more script variables

    - A command to be passed to the host (DFSee) to be executed, including substitution of expressions

Example for script layout

;script example

- A comment line

;;defaultparam 1 5

- A pragma

IF $1 < $_parts

- Control statement with an expression

  Say $1 is OK!

- A command to be executed by DFSee

ENDIF

- End of the Control statement

Script parameters and variables

- Parameters to the script are positional, and named $1 through $9, $0 is the scriptname

- Variables follow the 'Perl' syntax where possible, with a subset of the functionality

  - $variable   a scalar variable
  - $array[index]   scalar taken from an array
  - %array   whole array
  - $hash{key}   scalar taken from a hash
  - #hash   whole hash variable

System variables

- Variablenames starting with '$_' are system variables (DFSee) and are read-only

  - They come as scalar and scalar-from-array variants

- Some examples (there are dozens :-)

  - $_parts   total number of partitions, 1..n
  - $_disk   current opened disk number
  - $_this   sector number for current sector
  - $_d_size[X]   size in sectors for disk nr X
  - $_p_fsform[Y]   FS-format for partition nr Y
  - $_b_sector[Z]   Contents of sector nr Z, in a (512 byte) binary string

Expression and variable values

- Variable and expression values are either:

    - A character string of arbitrary length, may contain any value from 0..255, allowing binary data manipulation

    - A 64-bit signed integer value, allowing huge numbers while maintaining the exact integer value

- Expression operators and built-in functions automatically convert between these

    - Other types like floating-point may be added later

FSYS - Software   TxWin   DFSee

Expressions, operators, functions

- Expression syntax and semantics are pretty close to those defined in 'Perl' and 'C' but are not exactly identical

- Operators work on 1, 2 or 3 operands:

    - Unary, like + - ! NOT   1 operand
    - Binary, like + * < =   2 operands
    - Ternary, (cond) ? exp1 : exp2   3 operands

- Functions take zero or more arguments and return a value (in an expression)

Operator precedence, high to low

$name[]++ --

- Variable, indexed and auto incr/decr

Atom, Term

- String, number, function nested-expr or ternary

- + ! ~

- Unary operators

* / %

- Binary multiply/division

+ -

- Binary plus/minus

x

- String replication

.

- String concatenation

<< >>

- Numeric bit-shift

== != < > <= >=

- Numeric compare

=== !==

- Same value AND type

eq ne lt gt le ge

- String compare

Operator precedence, part 2

&	▪	Bitwise AND
^	▪	Bitwise XOR
|	▪	Bitwise OR
&&	▪	Logical AND (C-style)
||	▪	Logical OR (C-style)
=	▪	Assignment
,	▪	Comma, multi-expression
NOT	▪	Logical NOT (Perl style)
AND	▪	Logical AND (Perl style)
OR	▪	Logical OR (Perl style)

Built-in functions, A-F

abs	▪ Absolute value, numeric
b32	▪ Clip to 32-bit unsigned
b2asc	▪ Binary string to ASCII
b2int	▪ Binary string to reversed int
chr	▪ ASCII value for number
canceled	▪ Test for canceled last operation
confirmed	▪ Confirmation Yes/No/Cancel
defined	▪ Is variable defined
drivefs	▪ FS-name for drive letter
drivelabel	▪ Label string for drive letter
drives	▪ All drive letters in string
drivespace	▪ Freespace in KiB for drive
exists	▪ File exists
filext	▪ Set default file extension
fnbase	▪ Extract filename without ext

Built-in functions, G-M

fnfile	▪ Extract filename without path
fnpath	▪ Extract path only, no filename
getcwd	▪ Get current working directory
h2asc	▪ Get string from hex-ascii str
h2int	▪ Get integer from hex-ascii str
i2dec	▪ Convert int to decimal str
i2hex	▪ Convert int to hexadecimal str
index	▪ Find substring in string
lc	▪ Return lowercased string
left	▪ Left adjust string, pad/clip
length	▪ Get length of string
makedir	▪ Create full directory path
max	▪ Ret maximum of values
min	▪ Ret minimum of values
message	▪ Message popup, until [OK]

Built-in functions, O-Z

ord	• Numeric value 1st char in str
prompt	• Popup question, return string
replace	• Replace characters in string
sec2gib	• Get GiB value for #sectors
sec2kib	• Get KiB value for #sectors
sec2mib	• Get MiB value for #sectors
reverse	• Reverse characters in string
right	• Right adjust string pad/clip
rindex	• Reverse find substring in str
strip	• Strip leading/trailing chars from a string (default spaces)
substr	• Extract substring from string
uc	• Return uppercased string
undef	• Undefine (free) a variable releasing any used storage

Control structures, branching

IF (condition)

   statement-list

ELSEIF (condition)

   statement-list

ELSE

   statement-list

ENDIF

- Like the Perl IF, not using a {} block but an ENDIF keyword

- () parenthesis on conditions optional

- Any number of the ELSEIF clause

- ELIF, ELSIF and ELSEIF accepted

Control structures, looping

WHILE (condition)
    statement-list
ENDWHILE

FOR init;condition;iterator
    statement-list
ENDFOR

DO
    statement-list
UNTIL (condition)

- Like Perl or 'C' but not using a {} block but an explicit end

- () parenthesis on conditions optional

- Loop can be exited using 'break'

  - 'continue' or the perl 'last' are considered

Command expression substitution

- Transparent, replacing expressions by the result of the expression, when starting with a variable:

  - $_this + 100
  - Wipe z $start $_d_cylsize * 25
  - Say You have $_parts partitions on $_disks disks

- Explicit, enclose in curly brackets if NOT starting with a variable, or any conflicting syntax:

  - Restore {$imgfile} -P:$partition   ; -P would conflict
  - Say we are in: {getcwd()}   ; not a variable

Miscelaneous comments

- Keywords are case-insensitive (IF, WHILE)

- Parenthesis on conditions are optional

- Lines are 'continued' using '\' as last char allowing long expressions to be spread over more than one physical line

- Script syntax is checked BEFORE running any statement, except expressions to be substituted in commands (to be refined :-)

Considered improvements

- User defined functions

- More/better array and hash variable handling and manipulation (perl like)

- A FOREACH loop/iterator control

- Floating point variables

TxWin enhanced scripting

Questions ?