Paper 70-2018

Ron Fehd, SAS-L's Macro Maven, Answers Your Questions on Macros and Reusable Program Development

Ronald J. Fehd, Stakana Analytics

Abstract

Description: SAS ® software consists of two languages, SAS and its macro language.

Purpose: The purpose of this question-and-answer session is to provide overview and perspective of how SAS works and how the macro language can work both within and before SAS program statements and steps.

Audience: all levels

Keywords: macros, autoexec, conditionals, configuration, compile, execution, loops

In this paper

Introduction

Overview

These slides discuss my overview of programming and software design.

What is Information?

Information is **the** difference
that makes **a** difference
Gregory Bateson, 1904–1980
Steps to an Ecology of Mind, 1972

Vilfredo Pareto, 1896
Pareto principle 80/20 rule
80% of effects come from 20% of the causes
vital few and useful many
popularized by Joseph Juran, 1941

slide 2

aspects of learning computer language

- variables — constants — scope: global or local symbol table
- conditions — `if... then... else...` — used to: stop
  - comparisons: lt, le, eq, ge, gt — additional code
  - logical operators: and, or, not — branching
- loops — conditional exit — `while leave until`
  - enumerate: iterative — `start, stop, step=n`
    - sequence: step eq 1
    - series: step ne 1
  - itemize: list of items in associative array
- functions — macro, method, process, or procedure
- syntax — how to write that idea in this language

slide 3

HIPO, three aspects of every program

Hierarchical	location	type	calls
	top	module	routines
	middle	routine	subroutines
	bottom	subroutine	
Input	*ingredients*	data structure	
Process	*algorithm*	process	procedure
		N steps	Q: if . . . else
	executed:	always	conditionally
Output	*name of recipe*	data structure	

IBM Corporation, HIPO — A Design Aid and Documentation Technique, [25]

slide 4

data structure or algorithm?

20%	80%
compiled	executed
data structure	algorithm
define object	method
`attribute` statement(s):	
less than 1% of program	
$(20\%)^3 \cong 0.8\%$	

slide 5

How SAS works

processing a job

OpSys	icon	`.../sas.exe -config .../sasv9.cfg`
	sas.cmd	`.../sas.exe %*`
	my-program.bat	`sas my-program` *<command-line options>*

startup	configuration files	`<path>/nls/en/sasv9.cfg`
		`<project>/sasv9.cfg`
		add e-vars
		add/change startup-only options:
		`initcmd, initstmt, termstmt`

	command-line	startup-only options: `echoauto, oplist/verbose, sysparm`
setup	`autoexec`	add/change entries in global symbol table for access to libraries
	`initcmd`	initial command, an application window
	`initstmt`	initial statements
job	`my-program.sas`	—> my-program.log, .lst
shutdown	`termstmt`	terminating statements, `endsas`

notes

- `initcmd, initstmt` and `termstmt` are startup-only options, which can be set in a configuration file, e.g.: `sasv9.cfg` or on command line

citations

- `sas.exe` in `sas.cmd, sasv9.cfg, autoexec:` [2], [1]
- `echoauto:` [15], [8],
- sysparm companion: [3], [21]
- autoexec companion: [19]
- `initstmt:` [4]
- `termstmt:` [5], [6]

slide 6

autoexec process

adding entries to global symbol table for access to libraries [19]

catnames		formats, macros, compiled programs
environment variable(s)		site_root
filenames		project, site_includes, site_macros
libname		library
macro		variables definitions
options	autocall:	`mautosource` `sasautos=(project site_macros sasautos)`

	compiled+stored:	`libname mac_lib '...';`
		`mstored sasmstore = mac_lib;`
		sasautos=(project site_macros sasautos)

running text:		titles footnotes

slide 7

Conditionals

Conditionally execute global statements using `sysfunc` and `ifc`, [9]

True is not false, [23]

Truth table [7]

Writing testing-aware programs, [8]

---

if fail then stop

```
%let data = sashelp.class;
%sysfunc(ifc(%sysfunc(exist(&data))
            ,%nrstr(%put info:     exist &=data;)
            ,%nrstr(%put fail: not exist &=data;
                    endsas;
        )   )      )
```

slide 8

additional statements

```
DATA process_or_procedure;
     attrib    ...;       * <---<<< data structure;
*...;                     * <---<<< algorithm     ;
run;* step: update syslast;
*** echo data structure to log?;
%sysfunc(ifc(%sysfunc(getoption(source2)) eq SOURCE2
            ,%nrstr(proc sql;
                    describe table &syslast;
                    quit;) , ))
```

slide 9

conditionals in macro

```
%macro demo(data    = sashelp.class
           ,testing = 0 );
%let testing=%eval(   not(0 eq &testing)
                   or %sysfunc(getoption(mprint)) eq MPRINT);
*...;
%if &testing %then %do;
    proc sql; describe table &syslast;
              quit;
    %end;
```

Note: the comparison `not(0 equal &testing)` changes any value to *true*; this can also be written `&testing ne 0`

slide 10

Loops: List Processing

loops: list processing

list contains items — sets $n = cardinality$; arrays contain elements $n = dimension$

an item can be — attribute, value, or a list

a list does not contain data, — i.e. summable numeric facts

each row contains a set of parameters

other terms: — data- or table-driven; associative arrays — dynamic programs; self-modifying

slide 11

list processing tools

making lists, of files, data sets, variables [10]

cx-include: call execute a parameterized include [16]

macro callmacro [18]

macro calltext [22]

macro dateloop [17]

slide 12

using macro loops in functions

```
%let macro_name = proc_freq;
%let list       = a bb ccc;
                  *count word(s) delimited by space;
%let n_items    = %sysfunc(countw(&list,%str( )));

%do i = 1 %to &n_items;
    %let item = %scan(&list,&i);
    %put echo: &=item;
    %&macro_name(item=&item)    %* know semicolon!;
    %end;
```

Note: *count delimiters and add one;

```
%let n_items = %eval(%sysfunc(countc(&list,%str( )))+1);
```

Fehd and Carpenter, List Processing Basics, 2007 [24]

slide 13

issues when using macros

- autocall — autoexec: `filename macros '...';` `options sasautos = (macros sasautos);`
- macro functions — return tokens less than statement
- quoting

```
%let x = %nrstr(...);
%put %unquote(&x);
```

- suffix of reference is dot: `&mvar.` `($)format&length..` `&libname..memname` `&filename..txt`

slide 14

reasons to use macros

- variables: programs have access to global symbol table
- control: add statements or branching
- loops: write macro definitions as functions

citations: Do we need macros [20]

Macro design ideas [14]

slide 15

Summary

Conclusion

Using SAS software effectively requires gaining an understanding of software design issues.

Author Information

Ronald J. Fehd mailto:Ron.Fehd.macro.maven@gmail.com

sco.wiki http://www.sascommunity.org/wiki/Ronald_J._Fehd
LinkedIn www.linkedin.com/Ronald.Fehd
affiliation Stakana Analytics, Senior Maverick
also known as `macro maven` on SAS-L, Theoretical Programmer

Programs: http://www.sascommunity.org/wiki/

Macro_CallMacro [12]

Macro_CallText [11]

Macro_DateLoop [13]

Routine Call-Exec-Include [16]

Conditionally Execute Global Statements [9]

List Processing Basics, Fehd and Carpenter [24]

References

[1] Editor R.J. Fehd. Setting up project config and autoexec. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Setting_Up_Project_Config_and_AutoExec.

[2] Editor R.J. Fehd. Batch processing under windows. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Batch_processing_under_Windows.

[3] Editor R.J. Fehd. Parse sysparm. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Parse_sysparm.

[4] Editor R.J. Fehd. Writing info to log using option initstmt. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/PassInfX_Writing_Info_to_Log_with_InitStmt.

[5] Editor R.J. Fehd. Option termination statement. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Option_TermStmt.

[6] Editor R.J. Fehd. Saving job time. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Saving_Job_Time.

[7] Editor R.J. Fehd. Truth table of logical operators *and*, *or*, *not*, *xor* and De Morgan's laws of *nand, nor*. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Truth_Table.

[8] Editor R.J. Fehd. Writing testing aware programs. In *sasCommunity.org*, 2007. URL http://www.sascommunity.org/wiki/Writing_Testing_Aware_Programs.

[9] Editor R.J. Fehd. Conditionally executing global statements. In *sasCommunity.org*, 2008. URL http://www.sascommunity.org/wiki/Conditionally_Executing_Global_Statements. topics: combining functions sysfunc and ifc; info: example assertions, caveats on logical comparisons.

[10] Editor R.J. Fehd. Making lists. In *sasCommunity.org*, 2008. URL http://www.sascommunity.org/wiki/Making_Lists. how to make lists of data sets, and variable names using proc contents and sql; using data steps to get lists of files and folders.

[11] Editor R.J. Fehd. Macro Call-Macro. In *sasCommunity.org*, 2012. URL http://www.sascommunity.org/wiki/Macro_CallMacr. using SCL functions to read a data set and call macros.

[12] Editor R.J. Fehd. Macro Call-Text. In *sasCommunity.org*, 2012. URL http://www.sascommunity.org/wiki/Macro_CallText. using SCL functions to read a data set and return tokens within a statement.

[13] Editor R.J. Fehd. Macro loops with dates. In *sasCommunity.org*, 2013. URL http://www.sascommunity.org/wiki/Macro_Loops_with_Dates. example macros, programs and updates.

[14] Editor R.J. Fehd. Macro design ideas. In *sasCommunity.org*, 2014. URL http://www.sascommunity.org/wiki/Macro-Design_Ideas. history, theory, templates for documentation and macros, best practices.

[15] Ronald J. Fehd. Writing testing-aware programs that self-report when testing options are true. In *NorthEast SAS Users Group Conference Proceedings*, 2007. URL http://www.nesug.org/Proceedings/nesug07/cc/cc12.pdf. Coders' Corner, 20 pp.; topics: options used while testing: echoauto, mprint, source2, verbose; variable testing in data step or macros; call execute; references.

[16] Ronald J. Fehd. List processing routine CallXinc: Calling parameterized include programs using a data set as list of parameters. In *Western Users of SAS Software Annual Conference Proceedings*, 2009. URL www.lexjansen.com/wuss/2009/app/APP-Fehd2.pdf. Applications Development, 20 pp.; call execute, data review, data structure, dynamic programming, list processing, parameterized includes, examples.

[17] Ronald J. Fehd. Writing macro do loops with dates from then to when. In *MidWest SAS Users Group Annual Conference Proceedings*, 2013. URL http://analytics.ncsu.edu/sesug/2013/CC-03.pdf. 20 pp.; topics: dates are integers, formats and functions to convert date references to integers, calculations, bibliography.

[18] Ronald J. Fehd. List processing macro call-macro. In *MidWest SAS Users Group Annual Conference Proceedings*, 2014. URL www.mwsug.org/proceedings/2014/BB/MWSUG-2014-BB04.pdf. Coders Corner, 19 pp.; using %sysfunc with SCL functions to read a list, a control data set, and for each observation, call a macro with variable names and values as named parameters.

[19] Ronald J. Fehd. An autoexec companion, allocating location names during startup. In *MidWest SAS Users Group Annual Conference Proceedings*, 2015. URL http://www.lexjansen.com/mwsug/2015/BB/MWSUG-2015-BB-10.pdf. Beyond Basics, 15 pp.; autocall macros, global symbol table, filerefs, librefs, cexist catalogs, exist data set, sasautos.

[20] Ronald J. Fehd. Do we need macros? an essay on the theory of application development. In *MidWest SAS Users Group Annual Conference Proceedings*, 2015. URL http://www.lexjansen.com/mwsug/2015/BB/MWSUG-2015-BB-11.pdf. Beyond Basics, 10 pp.

[21] Ronald J. Fehd. A sysparm companion, passing values to a program from the command line. In *MidWest SAS Users Group Annual Conference Proceedings*, 2016. URL http://www.mwsug.org/proceedings/2016/TT/MWSUG-2016-TT04.pdf. Tools of Trade, 8 pp.; shows use of sysparm as macro variable and option which can be assigned value on command line in batch programs; program parse-sysparm parses a list of comma-separated values (csv) of form var1=value1,var2=value2,...,varN=valueN into macro variables.

[22] Ronald J. Fehd. List processing macro call-text. In *MidWest SAS Users Group Annual Conference Proceedings*, 2016. Tools of Trade, 10 pp.; using %sysfunc with SCL functions to read a list, a control data set, and for each observation, return %unquoted text.

[23] Ronald J. Fehd. True is not false: Evaluating logical expressions. In *SouthEast SAS Users Group Conference Proceedings*, 2016. URL https://analytics.ncsu.edu/sesug/2016/BB-186_Final_PDF.pdf. Beyond the Basics, 9 pp.; Boolean algegra, Boolean logic, De Morgan's law, evaluation, logical operators, sql joins.

[24] Ronald J. Fehd and Art Carpenter. List processing basics: Creating and using lists of macro variables. In *SAS Global Forum Annual Conference Proceedings*, 2007. URL http://www2.sas.com/proceedings/forum2007/113-2007.pdf. Hands On Workshop, 20 pp.; comparison of methods: making and iterating macro arrays, scanning macro variable, writing calls to macro variable, write to file then include, call execute; using macro function nrstr in call execute argument; 11 examples, bibliography.

[25] IBM, editor. *HIPO — A Design Aid and Documentation Technique*. IBM Corporation, White Plains, NY, 1974. URL http://en.wikipedia.org/wiki/HIPO. Publication Number GC20-1851.