BASUG-2022-April-Fehd-3-Q-and-A

Ronald J. Fehd
Senior Maverick
Fragile-Free Software Institute

BASUG, 2022-Apr-13

Contents

1 Do Which

1.1 stop statement

Is the 'stop' statement necessary?

Technically: no. The SAS Supervisor will execute the loop shown in the paper only once.

Here's where it is necessary:

From SAS documentation:

Example 2: Avoiding an Infinite Loop This example shows how to use STOP to avoid an infinite loop within a DATA step when you are using random access methods:

```
data sample;
do sampleobs = 1 to totalobs by 10;
   set master.research
       point = sampleobs
       nobs  = totalobs;
```

```
   output;
end;
stop;
run;
```

I always write explicit read loops because that visually separates the data structure from the algorithm.

```
data sample;
     * data structure:;
     * attrib  ...;
     * array   ...;
     * retain  ...;

do until(end_of_file);
   set sashelp.class
       end  = end_of_file;
   *calculation(s);
   output;
   end;
* post processing;
stop;
run;
```

Other writers use an implicit loop:

```
data sample;
if _n_ eq 1 then do;
     * data structure:;
     * attrib  ...;
     * array   ...;
     * retain  ...;
     end;

*do until(end_of_file);
   set sashelp.class
       end  = end_of_file;
   *calculation(s);
*   output;
*   end;

if end_of_file then do;
   * post processing;
   end;
*stop;
run;
```

The results are the same.

2 Do We Need Macros

2.1 %put &=data

Please explain the syntax: %put &=data;

```
%let data = sashelp.class;
%put &data;
*: sashelp.class;

%put data = &data;
*: data = sashelp.class;

%put &=data;

*: DATA=sashelp.class;

* I add the word 'echo'
  so I can easily find all the occurences;
%put echo: &=data;

*: echo DATA=sashelp.class;
```

2.2 macros or proc lua?

What is your take on using macros vs proc lau?

I'm already good at using macros.

I'll have a look at this paper and decide whether I want to learn yet another computer language.

reference: Driving SAS with Lua

`https://support.sas.com/resources/papers/proceedings15/SAS1561-2015.pdf`

`https://www.lua.org/`

`https://en.wikipedia.org/wiki/Lua_(programming_language)`

2.3 autocall and naming collisions

How can we overwrite the macro references from an autocall?

For Ex: I have a macro name 'test' stored in location1 fileref and the next version of 'test' is stored in location2 fileref. I want to use the 'test' macro for some calls and again another version of 'test' for the rest of the calls.

I tried to re-use the sasautos with changing the filerefs order, but I couldn't get the new version reference to be used for rest of the calls

The problem you are having is known as *naming collision*.

Here is what you are doing and how SAS works.

```
filename fileref1 '.';*local: here, this folder;
option sasautos = (fileref1 sasautos);
%test()
*macro definition test.v1 is now in catalog work.sasmacro.test;

filename fileref2 'C:\SAS-projects\SAS-site\sas-macros';
options sasautos = (fileref2 sasautos);
*still using test.v1;
%test()
*sas found work.sasmacro.test and did not use autocall;
%include fileref2(test);
*or options mrecall;
*macro definition test.v2 is now in catalog work.sasmacro.test;
%test()
```

This is not a good idea.

Write one macro definition and provide a parameter which switches between the two actions that you want.

```
%macro test(which);
%if "&which" eq "" %then %do;
    %put echo &=which default;
    %end;
%else %do;
      %put echo &=which other;
      %end;
%mend;
```

Check out this paper:

Bulletproof Macros: Avoiding Macro Name Collisions

`https://analytics.ncsu.edu/sesug/2014/CC-132.pdf`

2.4 replace macros with user-generated functions

> *What do you think of replacing some macros by writing user-generated functions?*

I'm guessing you mean proc fcmp?

Yes, one ought to be spending twenty percent of one's time doing bricolage — creative tinkering — in order to learn other ways of using the language.

Just remember: you're creating a maintenance issue for yourself and others, if you share your programs.

2.5 source or source2?

options mprint source2 — why is it source2?

Option `source2` specifies whether SAS writes secondary source statements from %included files to the SAS log. While option `source` specifies whether SAS writes (echos) source (all) statements to the SAS log.

Note that neither of these options silences note written by the `%put` statement.