

New Data Step Functions in SAS® 6.12 - Part III Special Functions

Caroline Bahler, BIOPOP, Radford, VA

Abstract

The Data step in SAS® 6.12 is the recipient of 50+ new functions. These new functions are broken into the following groupings data set functions, external file functions, library and catalog functions, special functions, variable functions, and statistical functions. The objective of this paper is to discuss the special functions and to illustrate their potential uses.

Introduction

The specialty functions new with SAS 6.12 include the SYSGET, SYMSG, SYSRC and SYSTEM functions that occurred originally within SCL and two (2) totally new functions GETOPTION and SPEDIS. These functions can be utilized within a macro or Data step and provide the programmer with SAS and host system information, and a fuzzy spelling descriptor. Within a macro, a programmer will reference these functions by using the %sysfunc and/or %qsysfunc macros. Whereas, within a Data step the functions are called in a similar manner used within SCL code¹.

Documentation for each of the special functions can be found within the HELP area of SAS 6.12, including syntax specifications. To access this information a HELP user needs to click on the HELP icon within the toolbar and select one of the following areas from the main HELP menu - What's New and Base SAS. A listing of all of the new functions occurs within the What's New section under Base SAS - Functions. Or the information about the functions can be accessed within the complete listing of all Base SAS functions under Base SAS Documentation - Language Reference, SAS Functions³.

Host Information and Commands

SYSGET - This function provides the value of a host system environmental variable. For example, from the DOS prompt the environmental variables and their values are available using the DOS SET command (Fig. 1). The SYSGET functions copies the values of these variables into SAS macro or Data step variables for use within programs. One use, by a SAS programmer, could be security since the user id used to log into a network is accessible from these variables. The type and number of environmental variables available is completely host operating system dependent.

An example of code utilizing this function is:

```
data test;
  length result $80;
  result = sysget('windir');
run;
```

and

```
%let windir=%sysfunc(sysget(windir));
```

```
data test;
  length result $80;
  result="&windir";
run;
```

In both, cases the value copied into the variable windir is the Windows® directory.

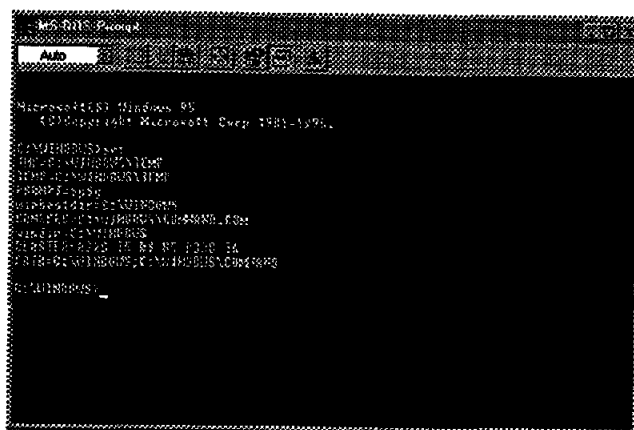


Figure 1. MS DOS environmental variables.

SYSTEM - This function is similar to the X statement⁴ (a global statement) and the Call System routine⁵ in that it allows an operating system command to be issued within a SAS program. All three methods for issuing a system command can occur within or outside of a Data step. An advantage of using the SYSTEM function is that it can be issued conditionally in a similar fashion the Call System routine within a Data step. However, utilizing the SYSTEM function has less overhead than the Call System routine. The X statement is a global statement and can not be issued conditionally except within a macro.

An example utilizing all three (3) methods of issuing a host system command is:

```
options noxwait;
data a;
x 'dir c:\sas\sasuser';
r=system('dir c:\sas\sasuser');
command='dir c:\sas\sasuser';
call system(command);
run;
```

In this example, all three commands will be issued one right after the other. The NOXWAIT system option indicates that an EXIT command is not necessary to exit the X command window (Fig. 3 and 4).

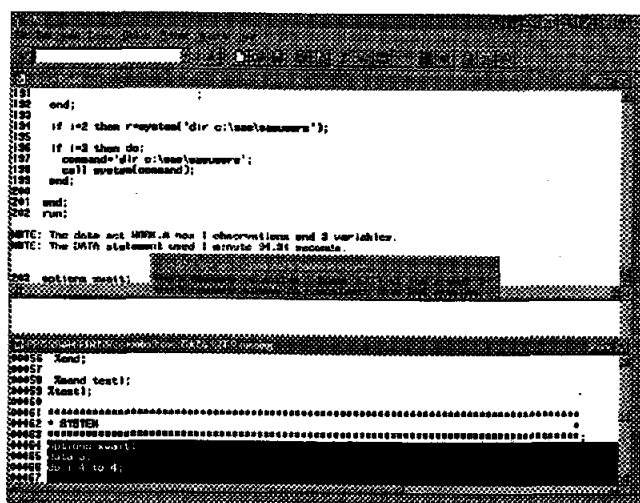


Figure 2. Message issued by SAS after system command issued through X statement, SYSTEM function or Call System routine.

An example of conditionally issuing a system command:

```
data a;
do i=2 to 3;
  if i=1 then do;
    x 'dir c:\sas\sasuser';
  end;
  if i=2 then r=system('dir c:\sas\sasusers');
  if i=3 then do;
    command='dir c:\sas\sasusers';
    call system(command);
  end;
end;
run;
```

In the example above the X statement will be issued even though the do loop specifies i values of 2 and 3. However, the SYSTEM function and Call System routine will only be issued when the value of i is 2 or 3 respectively.

Error Information

The SYMSG and SYSRC provide the SAS developer or programmer with information about errors that have occurred within a program when one of the new data set or external file functions are used. For example -

```
%macro test1;
%let test=sashelp.company;
```

```
%if %sysfunc(exist(&test)) %then %do;
%let opened=%sysfunc(open(&test,i));
```

etc.

```
%let closed=%sysfunc(close(&opened));
%end;
```

```
%else %do;
%let rc=%sysfunc(sysrc());
%let msg=%sysfunc(sysmsg());
%put &rc &msg;
```

```
%end;
```

```
%mend test1;
%test1;
```

In the example above, SAS statements will not be performed unless the data set exists³. If the data set does not exist then the error code and text message will be written for trouble shooting purposes.

Spelling/Misspellings

The SPEDIS function is a function that is described within the HELP facility as returning the asymmetric

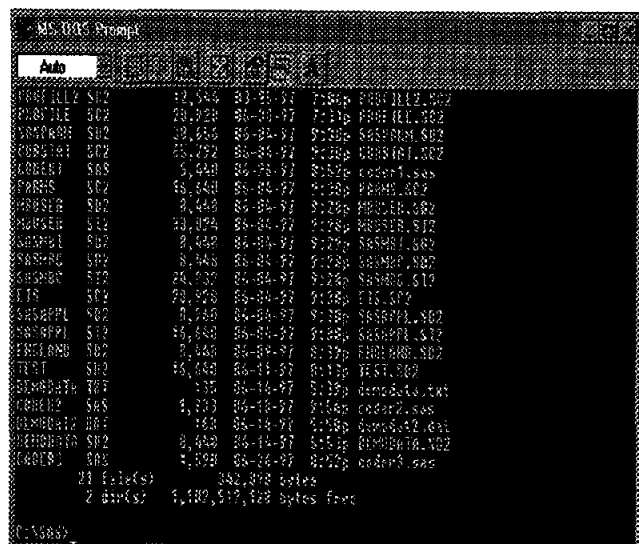


Figure 3. MS DOS prompt after system command issued. Exit command not necessary if NOXWAIT system option used.

distance between the word of interest and a keyword. The SPEDIS function distance when multiplied with the length of the query word gives an indication of the type of misspelling that could of occurred (defined as the cost within the HELP documentation). The default cost values are detailed within the HELP documentation for each type of misspelling that could occur. The type of misspelling that can be potentially detected are:

1. Complete match;
2. One letter of a double letter combination is missing;
3. One letter with the word is doubled;
4. A letter combination is reversed, ie. e before l when it should be l before e;
5. A letter is missing from the end of a word;
6. A letter is added to the end of a word;
7. A letter is missing from the middle of a word;
8. A letter is inserted into the middle of a word;
9. One letter within the middle of a word is incorrect;
10. First letter of a word is missing;
11. An additional letter occurs at the beginning of a word; and
12. First letter of a word is incorrect.

For example, the following code:

```
data special;
set sasuser.demodata;
match = ' Jacksonville';
dis = spedis(city,match);
cost = dis * length(city);
run;
```

Returns the following information which corresponds to the twelve conditions (12) described above:

TYPE	CITY	MATCH	COST		
			DIS	CALC	DEFLT
1	Jacksonville	Jacksonville	0	0	0
2	Jacksonville	Jacksonville	2	22	25
3	Jacksonville	Jacksonville	3	39	50
4	Jacksonville	Jacksonville	4	48	50
5	Jacksonville	Jacksonville	4	44	50
6	Jacksonville	Jacksonville	2	26	35
7	Jacksonville	Jacksonville	4	44	50
8	Jacksonville	Jacksonville	7	91	100
9	Jacksonville	Jacksonville	8	96	100
10	Jacksonville	Jacksonville	22	242	100
11	Jacksonville	Jacksonville	15	156	200
12	Jacksonville	Jacksonville	16	192	200

In this scenario, the keyword is the Match variable and the query variable is City. The spelling distance and cost in this example do not all correspond to the default costs for each condition. The defaults are a guide to allow the programmer to start defining

within a given data set data value what type of misspelling could have occurred. In the example above, the calculated cost in some cases is close to the default cost for a given type and for others the calculated cost is significantly different. The reason for this is that the first letter is capitalized and all other letters are lower case.

Changing the keyword and query word to the same case causes the calculated cost values come closer to the default values. In the example below, the only case where the calculated and default are equal occurs when the words match exactly.

TYPE	CITY	MATCH	COST		
			DIS	CALC	DEFLT
1	JACKSONVILLE	JACKSONVILLE	0	0	0
2	JACKSONVILLE	JACKSONVILLE	2	22	25
3	JACKSOONVILLE	JACKSONVILLE	3	39	50
4	JAKCSONVILLE	JACKSONVILLE	4	48	50
5	JACKSONVILL	JACKSONVILLE	4	44	50
6	JACKSONVILLEE	JACKSONVILLE	2	26	35
7	JAKSONVILLE	JACKSONVILLE	4	44	50
8	JACKSONSVILLE	JACKSONVILLE	7	91	100
9	JACKSOMVILLE	JACKSONVILLE	8	96	100
10	ACKSONVILLE	JACKSONVILLE	9	99	100
11	KJECKSONVILLE	JACKSONVILLE	15	195	200
12	HACKSONVILLE	JACKSONVILLE	16	192	200

As the number of misspelling types within a word increases so does the cost. For example, attempting to match JACSENVILLE to JACKSONVILLE results in a distance of 33 and a cost of 297. So care in interpreting the results from the SPEDIS function is necessary.

One great benefit that this function could provide is within a matching routine for mail lists. Utilizing the guideline described by Charles Patridge within his paper⁶, the SPEDIS function can be used as an additional check for potential matches. Especially, where breaking down individual components of a person's name or street address is feasible.

Another use of the SPEDIS function is in developing a specialized spell checker application. This could be accomplished by utilizing the misspelling types to correct misspelling within a word.

In conclusion, the new special functions within SAS 6.12 give programmers an assortment of tools to access host system information and determine if a misspelling may have occurred. These functions, in addition to the other new functions within SAS 6.12, allow a programmer flexibility in solving application development problems.

References

1. SAS Institute Inc., SAS Screen Control Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 91-374 pp.
2. SAS Institute Inc., SAS Screen Control Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 47-50 & 91pp.
3. Bahler, C. (1997) New Data Step Functions in SAS 6.12 - Part I Data Set Functions. Proceedings of the Fifth SouthEast SAS Users Group Conference.
4. SAS Institute Inc., SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 516 pp.
5. SAS Institute Inc., SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 633 pp.
6. Patridge, C. (1997) The Fuzzy Feeling SAS Software Provides to the Electronic Matching of Records without Common Keys Proceedings of the 22nd Annual SUGI Conference. 142-147 pp.

SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

Windows is a registered trademark or trademark of Microsoft, Inc. in the USA and other countries. ® indicates USA registration.

Caroline Bahler
GIV Inc.
7335 Lee Highway
Radford, VA 24141
540-633-3500
cbahler@giv.com