Abstract Description: SAS® software has sections in its global symbol table for options and macro variables. Sysparm is both an option and a macro variable. As an option, it can be assigned during startup on the command line; in programs, values can be assigned with the options statement; values are stored and referenced as a macro variable.

Purpose: The purpose of this paper is to provide a general-purpose program, parse-sysparm.sas, which provides a method of deconstructing a list of comma-separated values (csv) into separate macro variables. This is a useful method of passing a set of parameter values from one program to another.

Audience: programmers

Keywords: macro variables, scan function, startup options, sysparm

In this paper

Introduction 2

Setup for Batch Processing 3

Demonstration of Concepts 4

Program Parse-Sysparm 5

References 8
Introduction

Overview

This section covers the basics of macro variable assignment and referencing syntax and the options assignment statement. These are the topics in this section.

- macro variable assignment
- echo syntax
- options syntax

macro variable assignment

A macro variable assignment statement has five elements.
1. the verb %let
2. the name of the macro variable
3. an equals sign (=)
4. the value, which is all text between the equals sign and
5. the ending semicolon. The %put statement can be used to echo the value to the log.

```
1 %let mvar = text;
2 %put mvar = &mvar;
log: mvar=text;
```

Since v9.3 this echo syntax is available.

```
1 %let mvar = text;
2 %put echo &=mvar;
log: echo MVAR=text;
```

options syntax

Sysparm can be assigned a value with either the macro variable assignment statement or the options statement.

```
1 %let sysparm = some text;
2 %put echo mvar: &=sysparm;
3 options sysparm = 'more text';
4 %put echo options: &=sysparm;
log: echo mvar: SYSPARM=some text
    echo options: SYSPARM=more text
```
Setup for Batch Processing

Overview

This section shows the files used in batch processing. These are the topics in this section.

- `sas.bat`
- `autoexec.sas`
- `my-program`: `my-program.bat`, `my-program.sas`
- `sysparm` on command line

`sas.bat`

The batch files for each program depend on this `sas.bat` file in the same folder.

```rem name: sas.bat```
``"C:\program-files\SASHome\SASFoundation\9.4\sas.exe" %*``

Notes: `%*` means pass all command-line parameters to SAS

`autoexec.sas`

The batch files for each program shown here depend on a `fileref` `site_inc` with a path to the folder which contains the `site-include` program `parse-sysparm` (`p-sysprm.sas`).

```* name: autoexec.sas;```
```filename site_inc '<...>\SAS-site\includes\';```

`my-program`

Two files are required to submit a program in batch.

1. `my-program.bat` and 2. `my-program.sas`.

```rem name: my-program.bat```
```sas my-program```
```* name: my-program.sas;```
```*...;```

`sysparm` on command line

This pair of batch and SAS programs show the `sysparm` command-line option used in the batch file and two SAS statements in the program used to echo the value in the log.

```rem name: demo-sysparm-command-line.bat```
```sas demo-sysparm-command-line -sysparm 'cmd-line'```

```*name: demo-sysparm-command-line.sas;```
```%put sysparm:%sysfunc(getoption(sysparm));```
```%put &sysparm;```

Log:

```%put sysparm:%sysfunc(getoption(sysparm));```
```sysparm:cmd-line```
```%put &sysparm;```
```SYSPARM=cmd-line```
Demonstration of Concepts

Overview
This is the overview, which consists of a list of topics in this section.

- macro variable contains assignments
- scanning macro variable with delimiters

macro variable contains assignments
The value of a macro variable is always and only text and the text may contain special characters. This program shows that the text may contain a phrase of name=value. This value may be used in the macro variable assignment statement.

```sas
1 *name: demo-mvar-eq-var-eq-text;
2 %let mvar=.;
3 %put echo &=mvar;
4 %let var_eq_value = mvar = value;
5 %put echo &=var_eq_value;
6 %let &var_eq_value;
7 %put echo &=mvar;
```

log:
```
2 %let mvar=.;
3 %put echo &=mvar;
   echo MVAR=. 
4 %let var_eq_value = mvar = value;
5 %put echo &=var_eq_value;
   echo VAR_EQ_VALUE=mvar = value
6 %let &var_eq_value;
7 %put echo &=mvar;
   echo MVAR=value
```

scanning macro variable with delimiters
This program shows the use of the macro `%scan` function to fetch two macro assignment statements from a delimited list.

```sas
1 %let sysparm = b=2+c=3;
2 %put &sysparm;
3 %let var_eq_text=%scan(&sysparm,1,+);
4 %put &=var_eq_text;
5 %let &var_eq_text;
6 %put &b;
7 %let %scan(&sysparm,2,+);
8 %put &c;
```

log:
```
1 %let sysparm = b=2+c=3;
2 %put &sysparm;
   SYSPARM=b=2+c=3
3 %let var_eq_text=%scan(&sysparm,1,+);
4 %put &=var_eq_text;
   VAR_EQ_TEXT=b=2
5 %let &var_eq_text;
6 %put &b;
   B=2
7 %let %scan(&sysparm,2,+);
8 %put &c;
   C=3
```
Program Parse-Sysparm

Overview

This is the overview, which consists of a list of topics in this section.

- parse-sysparm.sas
- parse-sysparm-test

parse-sysparm

This program brings together the concepts shown previously.

1. macro variable contains a delimited list
2. using scan function to separate set of tokens
3. assemble macro variable assignment statement

```sas
%put trace: p-sysprm=parse-sysparm beginning;
%put echo parameter: &=sysparm;
/* name: <UNC>\SAS-site\includes\p-sysprm.sas
parse-sysparm
description: convert sysparm to set of macro variables
purpose : subroutine for parameterized includes
note : multiple values are named parameters
and delimited by comma: a=1,b=2
usage:
autoexec: filename site_inc '...\SAS-site\includes\';
options sysparm = 'a=1,b=2,d=4';
%include site_inc(p-sysprm);
**** ******/
DATA _null_;
  attrib stmt length = $%length(*let_&sysparm!);
if length(sysparm()) then do;
  ** upper bound is n(equal signs);
do i = 1 to countc(sysparm(),'=');
  ** assemble macro variable assignment statement;
  stmt= catx( ' ','%let ',scan(sysparm(),i,','),';');
  putlog 'echo: ' stmt=;
call execute(cat('%nrstr(',stmt,')'));
end;
end;
stop;
run;
%put trace: p-sysprm=parse-sysparm ending;
```
parse-sysparm test

This program is used to test parse-sysparm.sas.

```sas
* name: <UNC>\SAS-site\sas-include-tests\parse-sysparm-test.sas;
options source2;
options sysparm = 'a1=1,b2=22,text=unquoted';
%include site_inc(p-sysprm);
options nonotes nosource nosource2;
options sysparm = 'data=sashelp.class,var=height,by=sex';
%include site_inc(p-sysprm);
options notes source;
%put _global_;
```

parse-sysparm test log

This is the log of the test program.

```sas
* name: <UNC>\SAS-site\sas-include-tests\parse-sysparm-test.sas;
options source2;
options sysparm = 'a1=1,b2=22,text=unquoted';
%include site_inc(p-sysprm);
NOTE: %INCLUDE (level 1) file SITE_INC(p-sysprm) is file C:\...\SAS-site\includes\p-sysprm.sas.
trace: p-sysprm=parse-sysparm beginning;
echo parameter: SYSPARM=a1=1,b2=22,text=unquoted
[included statements not shown]
echo: stmt=%let a1=1 ;
echo: stmt=%let b2=22 ;
echo: stmt=%let text=unquoted ;
[some notes not shown]
NOTE: CALL EXECUTE generated line.
+ %let a1=1 ;
+ %let b2=22 ;
+ %let text=unquoted ;
trace: p-sysprm=parse-sysparm ending;
NOTE: %INCLUDE (level 1) ending.
trace: p-sysprm=parse-sysparm beginning
echo parameter: SYSPARM=data=sashelp.class,var=height,by=sex
echo: stmt=%let data=sashelp.class ;
echo: stmt=%let var=height ;
echo: stmt=%let by=sex ;
trace: p-sysprm=parse-sysparm ending
65 %put _global_;
GLOBAL A1 1
GLOBAL B2 22
GLOBAL TEXT unquoted
GLOBAL DATA sashelp.class
GLOBAL VAR height
GLOBAL BY sex
```
Suggested Reading

Many authors place several values in `sysparm` and use the `scan` function to separate them. Janka [3], shows colon-delimited values in `sysparm`; Yue [7], shows colon-delimited values in `sysparm`; Coar [1], shows `sysparm` with values delimited by vbar parsed by autoexec in batch programming; Langston [5], shows parsing of `sysparm` with comma-separated values; Salter and Cumming [6], shows how to assign default values when `sysparm` is empty; Jackson [2], shows `sysparm()` in data step; Johnson [4], discusses issues in writing list-processing programs that pass values from one step to later programs.

Conclusion

`Sysparm` can be assigned by either a macro variable assignment statement or an options statement. The option `sysparm` can be used on the command line. In either case the value can contain one or more sets of assignment tokens which can be parsed and copied into the global symbol table. This method can be used to write a unit test of a program as well as run the program with parameters set on the command line.

Contact Information

Ronald J. Fehd
Ron.Fehd.macro.maven@gmail.com
sco.wiki
http://www.sascommunity.org/wiki/Ronald_J._Fehd
LinkedIn
www.linkedin.com/Ronald.Fehd
Stakana Analytics, Senior Maverick
also known as macro maven on SAS-L, Theoretical Programmer
Programs: sas.community.organization wiki Parse_sysparm

Trademarks

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries @ indicates USA registration. Other brand and product names are trademarks of their respective companies.
References


