List Processing with Call Execute: Routine CallXinc for Calling Parameterized Include Programs Using a Data Set as List of Parameters

Ronald J. Fehd, Centers for Disease Control, and Prevention, Atlanta, GA, USA

Abstract

Description: This paper reviews routine CallXinc, a parameterized include program, which reads a data set, converts each character variable in each row into a global macro variable assignment statement, and calls another parameterized include program. An example of list processing using this routine is shown.

Purpose: This is another in the Journeymen's Tools series.

Audience: intermediate to advanced users and macro programmers

Copyright Notice: This is a Derivative Work of the book, A SAS(R) Companion, a work in progress by R. J. Fehd, copyright 2008, 2009.

Information: This article shows how to convert unknown variable names and values to macro variable assignment statements.

Keywords: call execute, catx, list processing, nrstr, vname,

Contents

Introduction 2

Routine CallXinc 2

Algorithm 3

Full Listing 7

Usage Example 8

CallXinc-Test.sas 8

Conclusion 9

Bibliography 9

Download Programs 10

Author Information 10
Introduction

Overview

This paper examines the list processing issues of calling one routine or subroutine many times, using each row of data set variable values as a list of parameters.

History

SmryEachVar CiteT used these routines CallXAtt CallXcsv CallXProc CallXRpt CallXVal

Book work on CxInclude consolidated these five routines into a single routine CxInclude.

SmryEachVar, version 2, uses this single routine CallXinc.

Routine CallXinc

Overview

The program CallXinc is a parameterized include program.

It has two parameters which support these processes:

1. CxData: a data set name; reads the data set; for each row, converts each character variable into a global macro variable

2. CxInclude: a file-specification of one or more parameterized routines or subroutines; calls these parameterized include programs

Example Output

The routine CallXinc produces the following output.

```plaintext
program : example
1  %Let CxData = Work.ListMvars;
2  %Let CxInclude = Project(MySubRoutine);
3  %Include SiteIncl(CallXinc);

log : example
1  NOTE: CALL EXECUTE generated line.
2  1 + %Let Col1 = Value-Row1-Col1;
3  2 + %Let Col2 = Value-Row1-Col2;
4  3 + %Include Project(MySubRoutine);
5  4 + %Let Col1 = Value-Row2-Col1;
6  5 + %Let Col2 = Value-Row2-Col2;
7  6 + %Include Project(MySubRoutine);
```

Documentation

```plaintext
/* name: ?:\SAS-site\includes\CallXinc.sas
   is.a: parameterized include routine
   description: using data as parameters, call a parameterized include
   purpose : list processing: calling routine or subroutine
```
Algorithm

This program uses this algorithm.

1. allocate data structure
   (a) read input data structure
   (b) add local variables
   (c) provide index
   (d) enable testing
2. read all rows
   (a) read all columns, convert each to a macro variable
   (b) call subroutine
3. housecleaning
4. data step subroutine
   (a) report when testing
   (b) call routine

Allocate Data Structure

This routine reads the data structure of the list processing data set before allocating its own character variables, CxStmtnt and CxVname. This data structure supports the do I loop upper bound of dim(Mvar) -2.

Read input data structure

The ability of this routine to process any data set depends on the input data structure being allocated before the local processing variables.

```
23 DATA Work._Null_;
24 ** read input data structure into PDV;
25 if 0 then set &CxData.;
```
Add local variables

** Principle of Data Step Allocation of Local Variable Names: **
Identify temporary variables with a common prefix, such as underline.

The variable Stmnt (Statement) is used to hold the macro variable assignment statement and the name of the program to be included.

```plaintext
26 ** allocate local processing variables;
27 attrib _Name length = $ 32
28 _Stmt length = $128 label = 'Statement';
```

** Caveat:** The length of Stmnt is fragile. If the length of either, 1, the macro variable assignment statement or 2, the program to be included, is greater than this length then the statement sent to call execute will be truncated and have no terminating semicolon. The absence of the semicolon will generate an error message:

```plaintext
1 %Let Test1 = missing semicolon
2 %Let Test2 = has semicolon;
3 ERROR: Open code statement recursion detected.
```

Solution: increase the length of Stmnt, which increases the memory needed for this step.

** Provide index **

All character variables are assigned to the array.

```plaintext
29 array Mvar(*) _character_;
```

** Testing **

The variable Testing is used in the ExecStmt data step subroutine. The option source2 has two values, true: source2, and false: nosource2. The value of Testing is set to the boolean value of the option source2.

```plaintext
31 retain _Testing %eval(0 or %sysfunc(getoption(Source2)) eq SOURCE2);
```

** CiteT Writing Testing-Aware Programs **

** For All Rows **

This loop reads the input data set.

```plaintext
1 do until(EndoFile);
2   set &CxData. end = EndoFile;
3   *...;
4   end;
```

** CiteTFehd Do Which **
For All Columns

This loop through the array creates a macro variable assignment statement for each character variable in the list processing data set.

```plaintext
36  do _I = 1 to dim(Mvar) -2;
37    _Name = vname(Mvar(_I));
38    ** make statement: *let Name = value ;
39    _Stmnt = catx(' ','%let',_Name,'=',Mvar(_I),';');
40    link CxStmnt;
41  end;
42  _Stmnt = "%Include &CxInclude.;";
```

Notes:

Upper bound: dim(Mvar) -2

The loop upper bound excludes the CallXinc local variables Stmnt and Name.

Functions: vname(Mvar(I))

An array reference returns the value of the variable; the vname function returns the name of the variable.

catx(’ ’, ...)

The function catx concatenates its arguments and inserts the separator named in the first argument.

Link and Return: link ExecStmnt

The link statement is a go-to and return-from a data step subroutine.

Call Parameterized Include

The parameter CxInclude contains one or more file-specifications of parameterized include routines or subroutines. These statements call the named programs.

```plaintext
43  link CxStmnt;
44  end;
```

Housecleaning

**Principle of List Processing:** A program which allocates a global macro variable is responsible for its removal.

This routine allocates an unknown number of global macro variables for use by the called program. When processing is complete, these are no longer needed. They are removed from the global macro variable symbol table with the Symbol-Delete statement.

```plaintext
47  do _I = 1 to dim(Mvar) -2;
48    _Name = vname(Mvar(_I));
49    _Stmnt = catx(' ','%symdel',_Name,';');
50    link CxStmnt;
51  end;
```
Note: the `stop` statement ends the processing. This is the boundary before the data step subroutine definition.

Subroutine ExecStmt

The data step subroutine `ExecStmt` encapsulates two processes.

1. when testing, writes the value of `Stmtnt` to the log
2. call execute plus `nrstr` of the variable `Stmtnt`

```plaintext
if _Testing then putlog _Stmtnt=
call execute(cats('%nrstr(',_Stmtnt,')'));
return;
run;
```

Note that the variable Testing writes messages to the log when included statements are echoed to log with the option `source2`.

The variable Testing is assigned a value in ¶ p. 4.
Full Listing

/* name: \SAS-site\includes\CallXinc.sas
   is.a: parameterized include routine
   description: using data as parameters, call a parameterized include
   purpose : list processing: calling routine or subroutine
   parameters : data set name:
     CxData = libref.data(drop | keep | rename | where)
   file-specification:
     CxInclude = Project(SubroutineA)
   input : &CxData
   process : for each character variable make macro variable assignment
     call named Include(s): &CxInclude.
   output : from (sub)routine(s) : &CxInclude.
   Notes : RJF2 4/23/2008 made from routine CxInclude
   which is.a: Derivative Work of the book: A SAS(R) Companion
   usage :
     %let CxData = Libref.Data; *(drop | keep | rename =() | where = (1));
     %let CxInclude = Project(SubRoutine);*SiteIncl(SubRoutineA SubRoutineB);
     %Include SiteIncl(CallXinc);
   *********************************/

DATA Work._Null_; /* read input data structure into PDV;
   if 0 then set &CxData.;
   allocate local processing variables;
   retain _Testing %eval(0 or %sysfunc(getoption(Source2)) eq SOURCE2);
   do until(EndoFile);
     set &CxData. end = EndoFile;
     do _I = 1 to dim(Mvar) -2;
       _Name = vname(Mvar(_I));
       ** make statement: *let Name = value ;
       _Stmnt = catx(' ','%let',_Name,'=',Mvar(_I),';');
       link CxStmnt;
     end;
     _Stmnt = "%Include &CxInclude.;";
     link CxStmnt;
   end;
   ** Symbol-Delete global macro variables allocated by this routine;
   do _I = 1 to dim(Mvar) -2;
     _Name = vname(Mvar(_I));
     _Stmnt = catx(' ','%symdel',_Name,';');
     link CxStmnt;
   end;
   stop;

CxStmnt:
   if _Testing then putlog _Stmnt=;
   call execute(cats('%nrstr(',_Stmnt,')'));
   return;
run;
%Put Note: CallXinc of &CxData. &CxInclude. ending;
Usage Example

CallXinc-Test.sas

1 * name: CallXinc-Test;
2 options source2 obs = 2;
3
4 %Let CxData = sashelp.class;
5 %Let CxInclude = SiteIncl(PutGlobal);
6 %Include SiteIncl(CallXInc);
7 %symdel CxData CxInclude;
8 %put _global_;

Log when Testing

1 Stmnt=%let Name = Alfred ;
2 Stmnt=%let Sex = M ;
3 Stmnt=%Include SiteIncl(PutGlobal);
4 Stmnt=%let Name = Alice ;
5 Stmnt=%let Sex = F ;
6 Stmnt=%Include SiteIncl(PutGlobal);
7 Stmnt=%symdel Name ;
8 Stmnt=%symdel Sex ;
9 NOTE: There were 2 observations read from the data set SASHELP.CLASS.

Log of Call Execute

1 NOTE: CALL EXECUTE generated line.
2 1 + %let Name = Alfred ;
3 2 + %let Sex = M ;
4 3 + %Include SiteIncl(PutGlobal);
5 NOTE: %INCLUDE (level 2) file SITEINCL(PutGlobal)
6 is file C:\SAS-site\includes\putglobal.sas.
7 65 +%Put _global_;
8 GLOBAL CXDATA sashelp.class
9 GLOBAL CXINCLUDE SiteIncl(PutGlobal)
10 GLOBAL DOTSAS
11 GLOBAL NAME Alfred
12 GLOBAL SEX M
13 NOTE: %INCLUDE (level 2) ending.
14 NOTE: %INCLUDE (level 1) resuming.
15 4 + %let Name = Alice ;
16 5 + %let Sex = F ;
17 6 + %Include SiteIncl(PutGlobal);

End of Job

1 7 + %symdel Name ;
2 8 + %symdel Sex ;
3 67 +%Put Note: CallXinc of &CxData. &CxInclude. ending;
4 Note: CallXinc of sashelp.class SiteIncl(PutGlobal) ending
5 NOTE: %INCLUDE (level 1) ending.
6 68 %symdel CxData CxInclude;
Conclusion

FreqAll  The data review utility program FreqAll provides a short
data set summary using Proc Freq.
FreqLibname  The data review utility suite FreqLibname provides more
information, especially for numerics.
Call Execute  This paper shows that call execute is a powerful method
for list processing.
%Includes  Doing list processing with call execute of %Includes can
eliminate the use of macros. This yields clearer code.

Do until(EndoFile):  Do Which Fehd [3], sgf2007.067
Call Execute and %nrstr:  List Processing Basics Fehd and Carpenter [5], sgf2007.113 demonstrate the timing of the error of using call execute of macros without the macro function %nrstr.
SmryEachVar:  A Data Review Suite for Each Data Set in a Libref Fehd [4], sgf2008.003
Writing Testing-Aware Programs:  That Self-Report when Testing Options are True Fehd [1], pharmasug2008.P025
PharmaSUG 2008 set NewName=PO25

Bibliography


Download Programs

To get the code examples in this paper search www.sascommunity.org for Call Execute Parameterized Include.

Author Information

Author: Ronald J. Fehd mailto:RJF2@cdc.gov
Centers for Disease Control
4770 Buford Hwy NE
Atlanta GA 30341-3724

about the author:

---
education: B.S. Computer Science, U/Hawaii, 1986
SAS User Group conference attendee since 1989
SAS-L reader since 1994

experience: programmer: 20+ years
data manager at CDC, using SAS: 18+ years
author: 12+ SUG papers

SAS-L: author: 4,000+ messages to SAS-L since 1997