List Processing Routine CallXinc

Using Data Sets as Lists of Parameters

Ronald J. Fehd
Centers for Disease Control

SCSUG 2009

Outline

1. Introduction
   - Concepts
   - Demonstration
2. CallXinc
   - Algorithm
   - Data Structure
   - Process

Sound Byte

Any sufficiently advanced technology
is indistinguisible
from magic

Arthur C. Clarke

Routine CallXinc replaces macro %do loop

Sound Byte

Any sufficiently advanced technology
is indistinguisible
from magic

Arthur C. Clarke

Routine CallXinc replaces macro %do loop

Outline

1. Introduction
   - Concepts
   - Demonstration
2. CallXinc
   - Algorithm
   - Data Structure
   - Process

Names of Organization
Concepts

Array of elements

List of items

Expectations
Concepts

Forms	fill in the blank
Jigs	hold things in place
Masks	holes for eyes, nose, mouth, ears
Patterns	replication
Template	trim to fit

HIPO: identifying parameters

Concepts

Hierarchical

Input data set

Process (sub)routine

Output data or file

Program Types
Concepts

Types	Module	calls routines, subroutines
	Routine	calls subroutines
	Subroutine	no calls
Ideas	Height	fan-in, fan out
	Reusable	routines and subroutines

Program Types
Concepts

Types	Module	calls routines, subroutines
	Routine	calls subroutines
	Subroutine	no calls
Ideas	Height	fan-in, fan out
	Reusable	routines and subroutines

Two Ad Hoc Programs

```
* adhoc-1;
PROC Print data  = SAShelp.Class
          (where =                (Sex = 'F'));
           title2 "SAShelp.Class.Sex :  F";

* adhoc-2;
PROC Print data  = SAShelp.Class
          (where =                (Sex = 'M'));
           title2 "SAShelp.Class.Sex :  M";
```

Two Ad Hoc Programs

```
* adhoc-1;
PROC Print data  = SAShelp.Class
          (where =                (Sex = 'F'));
           title2 "SAShelp.Class.Sex :  F";

* adhoc-2;
PROC Print data  = SAShelp.Class
          (where =                (Sex = 'M'));
           title2 "SAShelp.Class.Sex :  M";
```

Parameterized Include Subroutine

Parameters are Options

```
PROC Print data    =  &Data.
          (where   =         (&Where.));
           title2   "&Data..&Where. ";
run;
```

is.a subroutine: no calls

Parameterized Include Calling Program

Module or Routine

```
%Let Data  = SAShelp.Class;

%let Where = Sex eq 'F';
%Include SiteIncl(print-data-where);

%let Where = Sex eq 'M';
%Include SiteIncl(print-data-where);
```

idea: routine calls subroutine
see print-data-where-demo.sas

Outline

1. Introduction
   - Concepts
   - Demonstration
2. CallXinc
   - Algorithm
   - Data Structure
   - Process

Information
A List

```
PROC Freq  data    = SAShelp.Class;
           tables                 Sex
           /  out =    Work.Class_Sexs;
```

Sex	Count	Percent
F	9	47.37
M	10	52.63

List Processing Program
is.a Module

```
%Let CxData    = Work.Class_Sexs;
%Let CxInclude = SiteIncl(print-subset);
%Include         SiteIncl(CallXinc);
```

see Ex-sashelp-class.sas

List Processing Program demo-class

Log: output from Routine CallXinc

```
NOTE:   CALL EXECUTE generated line.
1    +   %let Sex  = F ;
2    +   %Include SiteIncl(print-subset);
3    +   %let Sex  = M ;
4    +   %Include SiteIncl(print-subset);
```

for each row
for each column

Subroutine Print-Subset

```
PROC Print data =  &Libname..&Memname.
          (where=(                          &Name= "&Value"));
           title2 "&Libname..&Memname..&Name:  &Value";
run;
```

Outline

1. Introduction
   - Concepts
   - Demonstration
2. CallXinc
   - Algorithm
   - Data Structure
   - Process

Algorithm

1. allocate data structure
2. read rows
3. read columns
4. make assignment statement
5. call subroutine

Outline

1. Introduction
   - Concepts
   - Demonstration
2. CallXinc
   - Algorithm
   - Data Structure
   - Process

Allocate Data Structure

for use by upper bound

```
DATA    _Null_;

if 0    then set &CxData.;

attrib _Stmnt length = $128
       _Name  length = $ 32;

array  Mvar(*) _character_;

loop:  do I = 1 to dim(Mvar) -2;
```

Allocate Data Structure

for use by upper bound

```
DATA    _Null_;

if 0    then set &CxData.;

attrib _Stmnt length = $128
       _Name  length = $ 32;

array  Mvar(*) _character_;

loop:  do I = 1 to dim(Mvar) -2;
```

Allocate Data Structure

for use by upper bound

```
DATA    _Null_;

if 0    then set &CxData.;

attrib _Stmnt length = $128
       _Name  length = $ 32;

array  Mvar(*) _character_;

loop:  do I = 1 to dim(Mvar) -2;
```

Allocate Data Structure

for use by upper bound

```
DATA    _Null_;

if 0    then set &CxData.;

attrib _Stmnt length = $128
       _Name  length = $ 32;

array  Mvar(*) _character_;

loop:  do I = 1 to dim(Mvar) -2;
```

Allocate Data Structure

for use by upper bound

```
DATA    _Null_;

if 0    then set &CxData.;

attrib _Stmnt length = $128
       _Name  length = $ 32;

array  Mvar(*) _character_;

loop:  do I = 1 to dim(Mvar) -2;
```

Outline

1 Introduction
- Concepts
- Demonstration

2 CallXinc
- Algorithm
- Data Structure
- Process

Algorithm
Read Rows

```
do until     (EndoFile);
   set &CxData.
       end = EndoFile ;
   ...
   end;
stop;
```

Read Columns

```
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;
```

link == goto

Read Columns

```
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;
```

link == goto

Read Columns

```
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;
```

link == goto

Read Columns

```
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;
```

link == goto

Data Step Subroutine

Encapsulates Call Execute and No Rescan String

```
CxStmnt:
    call execute(cats('%nrstr(',_Stmnt,')'));
    return;
run;
```

Call Subroutine

```
_Stmnt = "%Include &CxInclude.;";
link CxStmnt;

end;* do until(EndoFile) == read rows;
stop;
```

Call Subroutine

```
_Stmnt = "%Include &CxInclude.;";
link CxStmnt;

end;* do until(EndoFile) == read rows;
stop;
```

List Processing Program demo-class

Log: output from Routine CallXinc

```
NOTE:   CALL EXECUTE generated line.
1    +   %let Sex  = F ;
2    +   %Include SiteIncl(print-subset);
3    +   %let Sex  = M ;
4    +   %Include SiteIncl(print-subset);
```

for each row
for each column

Summary
Magic

Data Struc.

- `if 0 then set &CxData.`
- `array Mvar(*) character`

Loop

- `do I = 1 to dim(Mvar) -2`

Process

- `Name = vname(Mvar(I))`
- `Stmnt = catx(' '`
  `,'%let',Name,'=',Mvar(I)`

Conclusion

Parameterized Includes or . . .

Macros?	• Routine CallXinc replace macro %do loop • Includes have line numbers in log
Develop	• HIPO: identify input, process, output • review library of routines and subroutines
Testing	• Unit: Subroutines are already tested • Integration: Routines and Subroutines

Author Information

Ronald J. Fehd **RJF2@cdc.gov**
Stat Software HelpDesk
SAS Site Rep
Centers for Disease Control
Atlanta, GA, USA

Presentation pdf: LaTeX Beamer class