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
Any sufficiently advanced technology is indistinguishable from magic

Arthur C. Clarke

Routine CallXinc replaces macro %do loop
Sound Byte

Any sufficiently advanced technology is indistinguishable from magic

Arthur C. Clarke

Routine CallXinc replaces macro %do loop
Outline

1. Introduction
   - Concepts
   - Demonstration

2. CallXinc
   - Algorithm
   - Data Structure
   - Process
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

<table>
<thead>
<tr>
<th>Types</th>
<th>Module</th>
<th>Routine</th>
<th>Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>calls routines, subroutines</td>
<td>calls subroutines</td>
<td>no calls</td>
<td></td>
</tr>
</tbody>
</table>

### Ideas
- Height: fan-in, fan out
- Reusable routines and subroutines
### Program Types

<table>
<thead>
<tr>
<th>Types</th>
<th>Module</th>
<th>calls routines, subroutines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Routine</td>
<td>calls subroutines</td>
</tr>
<tr>
<td></td>
<td>Subroutine</td>
<td>no calls</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ideas</th>
<th>Height</th>
<th>fan-in, fan out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reusable</td>
<td>routines and subroutines</td>
</tr>
</tbody>
</table>
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";
PROC Print data = &Data.
  (where = (&Where.));
  title2  "&Data..&Where. ";
run;

Is a subroutine: no calls
%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
```sas
PROC Freq data = SAShelp.Class;
    tables Sex
    / out = Work.Class_Sexs;

<table>
<thead>
<tr>
<th>Sex</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>9</td>
<td>47.37</td>
</tr>
<tr>
<td>M</td>
<td>10</td>
<td>52.63</td>
</tr>
</tbody>
</table>
```
%Let CxData = Work.Class_Sexs;
%Let CxInclude = SiteIncl(print-subset);
%Include SiteIncl(CallXinc);

see Ex-sashelp-class.sas
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 _Stmtnt 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 _Stmtnt 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  _Stmtnt  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 _Stmtnt 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
do until (EndoFile);
set &CxData.
  end = EndoFile ;
...
  end;
stop;
do _I = 1 to dim(Mvar) -2;

_Name = vname(Mvar(_I));

** make statement: %let Name = value;
_Stmnt = catx(' ','%let',_Name,'='
, Mvar(_I) ,
            ';');

link CxStmtnt;
end;

link == goto
do _I = 1 to dim(Mvar) -2;

_Name = vname(Mvar(_I));

** make statement: %let Name = value;
_Stmtnt = catx(' ','%let',_Name,'=',
    ,Mvar(_I) ,
    ' ;');

link CxStmtnt;
end;
do _I = 1 to dim(Mvar) -2;

_Name = vname(Mvar(_I));

** make statement: %let Name = value;
_Stmnt = catx(' ', '%let', _Name, '=' ,
Mvar(_I) , ' ';);

link CxStmtnt;
end;

link == goto
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

CxStmtnt:
call execute(cats('%nrstr(,,_Stmnt,')'));
return;
run;
_Stmtnt = "%Include &CxInclude.;;"
link CxStmtnt;

end;* do until(EndoFile) == read rows;
stop;
_Stmtnt = "%Include &CxInclude.;";
link CxStmtnt;

end;* do until(EndOfFile) == read rows;
stop;
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
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)')
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
Ronald J. Fehd  
**RJF2@cdc.gov**
Stat Software HelpDesk  
SAS Site Rep  
Centers for Disease Control  
Atlanta, GA, USA

Presentation pdf: **\LaTeX** Beamer class