An Introduction to SAS® Macros

SAS Program (Input Stack)

SAS Wordscanner (Tokenization)

Non-Macro (Tokens)

SAS Compiler

Expanded token

% and & Triggers

Macro Facility

Steven First, President
2997 Yarmouth Greenway Drive, Madison, WI  53711
Phone: (608) 278-9964, Web: www.sys-seminar.com

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.
An Introduction to SAS® Macros

This course was written by Systems Seminar Consultants, Inc. SSC specializes SAS software and offers SAS:

• Training Services
• Consulting Services
• Help Desk Plans
• Newsletter subscriptions to The Missing Semicolon™.

COPYRIGHT© 1999, 2005 STEVEN FIRST

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher. SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. The Missing Semicolon is a trademark of Systems Seminar Consultants, Inc.
Objectives

After completing this class students will:
• understand how the macro system fits with the rest of SAS software
• use system (automatic) macro variables
• create and use user defined macro variables
• define simple macros
• pass data from the data step to the macro system.
SAS Macro Overview

Macros construct input for the SAS compiler.

Functions of the SAS macro processor:
- pass symbolic values between SAS statements and steps
- establish default symbolic values
- conditionally execute SAS steps
- invoke very long, complex code in a quick, short way.

Notes:
- The MACRO PROCESSOR is the SAS system module that processes macros.
- The MACRO LANGUAGE is how you communicate with the processor.
Traditional SAS Programming

Without macros what are some things you cannot easily do?

• substitute text in statements like TITLEs
• communicate across SAS steps
• establish default values
• conditionally execute SAS steps
• hide complex code that can be invoked easily.
Flow Without Macros

Without macros, SAS programs are DATA and PROC steps.
- The program is scanned one statement at a time looking for the beginning of step (step boundary).
- When the beginning of step is found, all statements in the step are compiled.
- When the end of step is found (the next step boundary), the previous step executes.
SAS Step Boundaries

Step boundaries are the SAS keywords:

- DATA ENDSAS
- PROC LINES
- CARDS LINES4
- CARDS4 PARMCARDS
- DATALINES QUIT
- DATALINES4 RUN
Step Boundaries

RUN acts as an explicit step boundary in most PROCs.

data saleexps; <--Step, start compile
    infile rawin;
    input name $1-10 division $12
    years 15-16 sales 19-25
    expense 27-34;
    run; <--Step end, exec previous

proc print data=saleexps; <--Step start, start compile
    run; <--Step end, exec previous
proc means data=saleexps;
    var sales expense;
    run; <--Step end, exec previous

Notes:
• The use of RUN after each step is highly recommended.
Practice Exercises

Exercise 1:
• A program that will build several datasets used later in this course is provided on your system.
• Start SAS for this workshop WS148.
• Check the SAS log to insure that the program ran correctly, use the LIBNAME command, file icon, or submit:

```sas
proc contents data=work._all_;
   run;
```

to verify that the datasets were built correctly.
• Explore SAS if you would like to.
The SAS Macro Language

A second SAS programming language for string manipulation.

Characteristics:

- strings are sequences of characters
- all input to the macro language is a string
- usually strings are SAS code, but don't need to be
- the macro processor manipulates strings and may send them back for scanning.
Macro Language Components

The macro language has several kinds of components.

**Macro variables:**
- are used to store and manipulate character strings
- follow SAS naming rules
- are NOT the same as DATA step variables
- are stored in memory in a macro symbol table.

**Macro statements:**
- begin with a % and a macro keyword and end with semicolon (;)
- assign values, substitute values, and change macro variables
- can branch or generate SAS statements conditionally.
Macro Processor Flow

Macro statements are given to the macro processor BEFORE the compiler.

- Macro statements start with %.
- Macro variables are referred to with &.
Macro Processor Flow Example

---

**Your SAS program**

```
%let name=METRIC;
data &name;
infile rawin;
input name $ lbs;
kilos=lbs*.45;
run;
proc print;
run;
```

**SAS compiler**

1. Check syntax
2. If % or &
3. Set up datasets
4. Compile program

**Executable program**

```
Data METRIC;
infile rawin;
input name $ lbs;
kilos=lbs*.45;
run;
```

**Symbol Value**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>METRIC</td>
</tr>
</tbody>
</table>

**MACRO PROCESSOR**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>METRIC</td>
</tr>
</tbody>
</table>

**SAS dataset METRIC**

```
Descriptor
NAME   LBS   KILOS
-------------------
Tom    150   67.5
Julie  93    44.1
Lois   88    39.6
```

---

An Introduction to SAS Macros
A Macro Problem

Problem:
• You would like to have the Day of Week and current date appear in a title, but SAS titles are text, not variables.

Solution:
• use some system macro variables.
Solution to Macro Problem

PROC PRINT DATA=DEPTSALE;
  TITLE "Department Sales as of &SYSDAY &SYSDATE";
  TITLE2 "Deliver to Michael O'Malley";
RUN;

Notes:
- Macro variables are NOT resolved within single quotes.
Automatic Macro Variables

A partial list of automatic macro variables and their usage:

- **SYSBUFR**: text entered in response to %INPUT
- **SYSCMD**: last non-SAS command entered
- **SYSDATE**: current date in DATE6. or DATE7. format
- **SYSDAY**: current day of the week
- **SYSDEVIC**: current graphics device
- **SYSDSN**: last SAS dataset built (i.e., WORK SOFTSALE)
- **SYSENV**: SAS environment (FORE or BACK)
- **SYSERR**: return code set by SAS procedures
- **SYSFILRC**: whether last FILENAME executed correctly
- **SYSINDEX**: number of macros started in job
- **SYSINFO**: system information given by some PROCS
- **SYSJOBID**: name of executing job or user
- **SYSLAST**: last SAS dataset built (i.e., WORK.SOFTSALE)
- **SYSLIBRC**: return code from last LIBNAME statement
- **SYSLCKRC**: whether most recent lock was successful
### Automatic Macro Variables (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSMENV</td>
<td>macro execution environment</td>
</tr>
<tr>
<td>SYSMSG</td>
<td>message displayed with <code>%DISPLAY</code></td>
</tr>
<tr>
<td>SYSPARM</td>
<td>value passed from SYSPARM in JCL</td>
</tr>
<tr>
<td>SYSPROD</td>
<td>indicates whether a SAS product is licensed</td>
</tr>
<tr>
<td>SYSPBUFF</td>
<td>all macro parameters passed</td>
</tr>
<tr>
<td>SYSRC</td>
<td>return code from macro processor</td>
</tr>
<tr>
<td>SYSSCP</td>
<td>operating system where SAS is running</td>
</tr>
<tr>
<td>SYSTIME</td>
<td>starting time of job</td>
</tr>
<tr>
<td>SYSVER</td>
<td>SAS version</td>
</tr>
</tbody>
</table>

**Example:**

FOOTNOTE "THIS REPORT WAS RUN ON &SYSDAY, &SYSDATE";

**Resolves to:**

FOOTNOTE "THIS REPORT WAS RUN ON FRIDAY, 26MAR99";
Displaying Macro Variables

%PUT displays macro variables to the log at compile time.

Syntax:

%PUT text macrovariables ;
%PUT _all_;

Example:

DATA NEWPAY;
  INFILE DD1;
  INPUT EMP$ RATE;
RUN;
%PUT ***** &SYSDATE *****;

Partial SAS Log:

***** 26MAR99 *****
Displaying All Macro Variables

%PUT can display all current macro variables.

%PUT _ALL_;

Partial SAS Log:

GLOBAL MBILLYR 99
GLOBAL SSCDEV C
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCMD
AUTOMATIC SYSDATE 26MAR99
AUTOMATIC SYSDAY Tuesday
Partial SAS LOG Continued

AUTOMATIC SYSDSN   _NULL_
AUTOMATIC SYSENV FORE
AUTOMATIC SYSERR 0
AUTOMATIC SYSFILRC 0
AUTOMATIC SYSINDEX 1
AUTOMATIC SYSINFO 0
AUTOMATIC SYSJOBID 0000016959
AUTOMATIC SYSLAST _NULL_
AUTOMATIC SYSMSG
AUTOMATIC SYSPARM
AUTOMATIC SYSRC 0
AUTOMATIC SYSSCP WIN
AUTOMATIC SYSSCPL WIN_32S
AUTOMATIC SYSSITE 0011485002
AUTOMATIC SYSTIME 10:35
AUTOMATIC SYSVER 6.11
AUTOMATIC SYSVLONG 6.11.0040P030596
Practice Exercises

Exercise 2:
• Determine the values of the following system macro variables using your current computer system.

  &SYSDAY           Current day of the week
  &SYSTIME          Current time
  &SYSSCP           Operating system being used
  &SYSVER           Current SAS version number
  &SYSDATE          Current date, in DATE7. Format

Exercise 3:
• Using system macro variables run a PROC CONTENTS and a PROC PRINT on the LAST SAS dataset that was created. Include its name in a title.
Exercise 2 Solution

%PUT **** SYSDAY = &SYSDAY;
%PUT **** SYSTIME = &SYSTIME;
%PUT **** SYSSCP = &SYSSCP;
%PUT **** SYSVER = &SYSVER;
%PUT **** SYSDATE = &SYSDATE;
Exercise 2 Output

```sas
%PUT **** SYSDAY = &SYSDAY;
    **** SYSDAY = Friday
2  %PUT **** SYSTIME = &SYSTIME;
    **** SYSTIME = 13:42
3  %PUT **** SYSSCP = &SYSSCP;
    **** SYSSCP = WIN
4  %PUT **** SYSVER = &SYSVER;
    **** SYSVER = 6.12
5  %PUT **** SYSDATE = &SYSDATE;
    **** SYSDATE = 26MAR99
```
Exercise 3 Solutions

proc contents data=&syslast;
  title "contents of &syslast";
run;

The Generated Code:

proc contents data=WORK.COUNTYDT;
  title "contents of WORK.COUNTYDT";
run;
## Exercise 3 Partial Output

```plaintext
contents of WORK.COUNTYDT
CONTENTS PROCEDURE

Data Set Name: WORK.COUNTYDT  Observations: 6
Member Type: DATA  Variables: 2
Engine: V612  Indexes: 0
Created: 13:51 Friday, March 26, 1999  Observation Length: 23
Last Modified: 13:51 Friday, March 26, 1999  Deleted Observations: 0
Protection: Compressed: NO
Data Set Type: Sorted: NO
Label:
```
A Macro Problem

Problem: You reference a SAS datasetname several times in a SAS job.

DATA PAYROLL;
  INPUT EMP$ RATE;
  DATALINES;
  TOM 10
  JIM 10
;
PROC PRINT DATA=PAYROLL;
  TITLE "PRINT OF DATASET PAYROLL";
RUN;

Question: How can you change the name quickly in one place only AND have the datasetname appear in a title?

Solution:
• Use a macro variable.
Macro Variables

- You can define macro variables with %LET.
- You refer to the variables later with &variable.
- Macro will substitute value for all occurrences of &variable.

Syntax:

```sas
%LET variable=value;

%LET NAME=PAYROLL;
DATA &NAME;
  INPUT EMP$ RATE;
  DATALINES;
  TOM 10
  JIM 10
;
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;
```
The Generated SAS Code

DATA PAYROLL;
   INPUT EMP$ RATE;
   DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=PAYROLL;
   TITLE "PRINT OF DATASET PAYROLL";
RUN;

Notes:
• Macro variables are not resolved within single quotes.
• Leading and trail spaces are discarded.
Assigning a New Value

Use another %LET to assign a different value.

```sas
%LET NAME=NEWPAY;
DATA &NAME;
  INPUT EMP$ RATE;
  DATALINES;
TOM 10
JIM 10
;
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;
```
DATA NEWPAY;
  INPUT EMP$ RATE;
DATALINES;
  TOM 10
  JIM 10
; PROC PRINT DATA=NEWPAY;
TITLE "PRINT OF DATASET NEWPAY";
RUN;
Assigning SAS Statements

%STR allows values with ; etc.

%LET NAME=NEWPAY;
%LET CHART=%STR(PROC CHART;VBAR EMP;RUN;;);
DATA &NAME;
  INPUT EMP$ RATE;
  DATALINES;
  TOM 10
  JIM 10
;
&CHART
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;
The Generated SAS Code

```
DATA NEWPAY;
  INPUT EMP$ RATE;
DATALINES;
  TOM 10
  JIM 10
;
PROC CHART;VBAR EMP;RUN;
PROC PRINT DATA=NEWPAY;
  TITLE "PRINT OF DATASET NEWPAY";
RUN;
```
Nesting of Macro Variables

Macro variables can contain other macro variables.

```
%LET NAME=NEWPAY;
%LET CHART=%STR(PROC CHART DATA=&NAME; VBAR EMP; RUN;);
DATA &NAME;
  INPUT EMP$ RATE;
  DATALINES;
TOM  10
JIM  10
 ;
&CHART
PROC PRINT DATA=&NAME;
  TITLE "PRINT OF DATASET &NAME";
RUN;
```
The Generated SAS Code

DATA NEWPAY;
  INPUT EMP$ RATE;
  DATALINES;
TOM 10
JIM 10
;
PROC CHART DATA=NEWPAY;VBAR EMP;RUN;
PROC PRINT DATA=NEWPAY;
  TITLE "PRINT OF DATASET NEWPAY";
RUN;
Practice Exercises

Directions:
• Work out the problems below on paper.
• If terminals are available, log on, type in the statements, and use %PUT statements to check your answers.

Exercise 4:
• After execution of the following %LET statements

  %LET A=ANDY;
  %LET B=1999;
  %LET C=CANES;
  %LET D=DECEMBER 31,;
  %LET E="TREMENDOUS";
Exercise 4 (continued)

What would be the results of these %PUT statements?

```sas
%PUT &C;
%PUT FISCAL YEAR &B;
%PUT YEAR ENDED &D &B;
%PUT &B C&A&C WERE SOLD IN &B;
%PUT &B WAS A &E SALES YEAR!;
```
Exercise 4 Solution

%LET A=ANDY;
%LET B=1999;
%LET C=CANES;
%LET D=DECEMBER 31,;
%LET E="TREMENDOUS";
%PUT &C;
CANES
%PUT FISCAL YEAR &B;
FISCAL YEAR 1999
%PUT YEAR ENDED &D &B;
YEAR ENDED DECEMBER 31, 1999
%PUT &B C&A&C WERE SOLD IN &B;
1999 CANDY CANES WERE SOLD IN 1999
%PUT &B WAS A &E SALES YEAR!;
1999 WAS A "TREMENDOUS" SALES YEAR!

Symbol Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ANDY</td>
</tr>
<tr>
<td>B</td>
<td>1999</td>
</tr>
<tr>
<td>C</td>
<td>CANES</td>
</tr>
<tr>
<td>D</td>
<td>DECEMBER 31,</td>
</tr>
<tr>
<td>E</td>
<td>&quot;TREMENDOUS&quot;;</td>
</tr>
</tbody>
</table>
Other Macro Debugging Options

SAS gives several options that control log output.

- SYMBOLGEN/NOSYMBOLGEN controls display of variable resolution
- MPRINT/NOMPRINT displays generated statements given to the compiler
- MLOGIC/NOMLOGIC displays tracing information during macro execution.
Other Macro Debugging Options

A diagram of where macro debugging options display values:

- SAS Program (Input Stack)
- SAS Wordscanner
- SAS Compiler
- Macro Facility
- MLOGIC
- SYMBOLGEN
- MPRINT
What is a Macro?

Stored text that can be inserted anywhere in a SAS program and expanded.

Macros can include:

• constants such as literals, variables, names, and statements
• assignments to macro variables
• macro programming statements
• macro language functions
• invocations of other functions
• nested macro definitions
• **LOGIC** to conditionally generate SAS code.
%MACRO and %MEND define macros.

%macroname will invoke it later.

**Example:** Define a macro to run PROC CHART and later invoke

```
%MAMCRO CHART;
  PROC CHART DATA=&NAME;
    VBAR EMP;
RUN;
%MEND;
```
%LET NAME=NEWPAY;
DATA &NAME;
   INPUT EMP$ RATE;
   DATALINES;
TOM 10
JIM 10
;
RUN;
%CHART
PROC PRINT DATA=&NAME;
   TITLE "PRINT OF DATASET &NAME";
RUN;
The Generated Code

DATA NEWPAY;
    INPUT EMP$ RATE;
    DATALINES;
    TOM 10
    JIM 10
    ;
RUN;
PROC CHART DATA=NEWPAY;
    VBAR EMP;
RUN;
PROC PRINT DATA=NEWPAY;
    TITLE "PRINT OF DATASET NEWPAY";
RUN;
Exercise 5:

- If the following macro was defined to the SAS system:

```sas
%MATERIAL FREQ;
   PROC FREQ DATA=&DSN;
       TITLE "EXERCISE 5";
       TABLES &VAR1*&VAR2 / NOPERCENT;
       RUN;
%MEND FREQ;
```

- What code would the SAS compiler see after these statements?

```sas
OPTIONS MPRINT SYMBOLGEN;
%LET DSN=FREQ;
%LET VAR1=DEPT;
%LET VAR2=SALES;

%FREQ
```
PROC FREQ DATA=FREQ;
   TITLE "EXERCISE 5";
   TABLES DEPT*SALES / NOPERCENT;
RUN;
### Exercise 5 Output

#### Exercise 5

**TABLE OF DEPT BY SALES**

<table>
<thead>
<tr>
<th>Frequency</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEPT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SALES</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Row Pct</strong></td>
<td>50.00</td>
<td>50.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td><strong>Col Pct</strong></td>
<td>100.00</td>
<td>50.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DEPT</strong></th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SALES</strong></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

| **Total** | 1   | 2   | 1   | 4   |
Positional Macro Parameters

Macro parameters are defined in order after the macro name.

```
%MACRO CHART(NAME,BARVAR);
   PROC CHART DATA=&NAME;
      VBAR &BARVAR;
   RUN;
%MEND;

%CHART(PAYROLL,EMP)

Resolves to:

PROC CHART DATA=PAYROLL;
   VBAR EMP;
RUN;
```

Notes:

• Keyword parameters are also allowed, and can set default values.
Nested Macros

Macros can call other macros.

`%MACRO CHART(NAME,BARVAR);
    PROC CHART DATA=&NAME;
    VBAR &BARVAR;
    RUN;
%MEND;

%MACRO PTCHART(NAME,BARVAR);
%CHART(PAYROLL,EMP)
    PROC PRINT DATA=&NAME;
    TITLE "PRINT OF DATASET &NAME";
    RUN;
%MEND;

%PTCHART(PAYROLL,EMP)`
The Generated SAS Code

PROC CHART DATA=PAYROLL;
  VBAR EMP;
RUN;
PROC PRINT DATA=PAYROLL;
  TITLE "PRINT OF DATASET PAYROLL";
RUN;
Conditional Macro Compilation

%IF can conditionally pass code to the compiler.

**Example:** Run PROC PRINT only if PRTCH=YES.

```sas
%MACRO PTCHT(PRTCH,NAME,BARVAR);
  %IF &PRTCH=YES %THEN PROC PRINT DATA=&NAME;
    ;
    PROC CHART DATA=&NAME;
    VBAR &BARVAR;
  RUN;
%MEND;

%PTCHT(YES,PAYROLL,EMP)
```
The Generated SAS Code

PROC PRINT DATA=PAYROLL ;

PROC CHART DATA=PAYROLL;
   VBAR EMP;
RUN;
The %DO Statement

%DO allows many statements to be conditionally compiled.

Example: Submit as before, but include titles.

```
%MACRO PTCHT(PRTCH,NAME,BARVAR);
  %IF &PRTCH=YES %THEN
    %DO;
      PROC PRINT DATA=&NAME;TITLE "PRINT OF DATASET &NAME";
      RUN;
    END;
    PROC CHART DATA=&NAME;
    VBAR &BARVAR;
    RUN;
  %MEND;
%PTCHT(YES,PAYROLL,EMP)
```
The Generated SAS Code

PROC PRINT DATA=PAYROLL;
   TITLE "PRINT OF DATASET PAYROLL";
RUN;
PROC CHART DATA=PAYROLL;
   VBAR EMP;
RUN;
Interactive Macro Invocation

%DO can also vary a value.

**Example:** Run PROC PRINT &PRTNUM times.

```sas
%MALLOC PRTMAC(PRTNUM,NAME);
  %DO I= 1 %TO &PRTNUM;
    PROC PRINT DATA=&NAME&I;
    TITLE "PRINT OF DATASET &NAME&I";
  RUN;
  %END;
%MEND;

%PRTMAC(4,PAYROLL)
```
PROC PRINT DATA=PAYROLL1;
  TITLE "PRINT OF DATASET PAYROLL1";
RUN;
PROC PRINT DATA=PAYROLL2;
  TITLE "PRINT OF DATASET PAYROLL2";
RUN;
PROC PRINT DATA=PAYROLL3;
  TITLE "PRINT OF DATASET PAYROLL3";
RUN;
PROC PRINT DATA=PAYROLL4;
  TITLE "PRINT OF DATASET PAYROLL4";
RUN;
Exercise 6:

- If the following macro was defined to the SAS system:

```sas
%MACRO FREQ(DSN, VAR1, VAR2);
   PROC FREQ DATA=&DSN;
   TABLES &VAR1*&VAR2 / NOPERCENT;
   RUN;
%MEND FREQ;
```

- What code would the SAS compiler see after this macro call?

```sas
%FREQ(FREQ,SALARIES,SALES)
```
Exercise 6 Solution

PROC FREQ DATA=FREQ;
   TABLES SALARIES*SALES / NOPERCENT;
RUN;
SYMGET, SYMPUT, and macro variables can transfer values between SAS steps.

**Example:** Display the number of observations in a dataset in a title.

```sas
%MACRO OBSCOUNT(NAME);
   DATA _NULL_;
      SET &NAME NOBS=OBSOUT;
      CALL SYMPUT('MOBSOUT',OBSOUT);
      STOP;
   RUN;
      PROC PRINT DATA=&NAME;
         TITLE "DATASET &NAME CONTAINS &MOBSOUT OBSERVATIONS";
   RUN;
%MEND;

%OBSCOUNT(PAYROLL)
```
The Generated SAS Code

DATA _NULL_;
  SET PAYROLL NOBS=OBSOUT;
  CALL SYMPUT('MOBSOUT',OBSOUT);
  STOP;
RUN;
PROC PRINT DATA=PAYROLL;
  TITLE "DATASET PAYROLL CONTAINS 50 OBSERVATIONS";
RUN;

Notes:
• SYMGET returns macro variable values to the DATA step
• Macro variables created with SYMPUT, can be referenced via & in the NEXT step.
The following problem needs some help.

Data Set COUNTYDT

<table>
<thead>
<tr>
<th>Obs</th>
<th>COUNTYNM</th>
<th>READING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ASHLAND</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>ASHLAND</td>
<td>611</td>
</tr>
<tr>
<td>3</td>
<td>BAYFIELD</td>
<td>101</td>
</tr>
<tr>
<td>4</td>
<td>BAYFIELD</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>BAYFIELD</td>
<td>222</td>
</tr>
<tr>
<td>6</td>
<td>WASHINGTON</td>
<td>143</td>
</tr>
</tbody>
</table>
Problem: Each day you read a SAS dataset containing data from counties in Wisconsin. Anywhere between 1 and 72 counties might report that day. Do the following:

1. Create a separate dataset for each reporting county.
2. Produce a separate PROC PRINT for each reporting county.
3. In the TITLE print the county name.
4. Reset the page number to 1 at the beginning of each report.
5. In a footnote print the number of observations processed for each county.

Question: How do you do it?
Solution: A Data Step and a SAS Macro.

A data step and a macro to generate the PROC PRINTs.

The data step goes through the data and:

• counts counties
• counts observations per county, puts in macro variables
• puts countynms into macro variables
• puts total counties reporting into a macro variable.
The Data Step Code

DATA _NULL_;  
SET COUNTYDT END=EOF;     /* READ SAS DATASET     */
   BY COUNTYNM;           /* SORT SEQ                */
   IF FIRST.COUNTYNM THEN DO;   /* NEW COUNTY ?       */
      NUMCTY+1;          /* ADD 1 TO NUMCTY      */
      CTYOBS=0;         /* OBS PER COUNTY TO 0 */
   END;
   CTYOBS+1;                   /* ADD ONE OBSER FOR CTY */
IF LAST.COUNTYNM THEN DO;  /* EOF CTY, MAKE MAC VARS*/
   CALL SYMPUT('MCTY'||LEFT(PUT(NUMCTY,3.)),COUNTYNM);
   CALL SYMPUT('MOBS'||LEFT(PUT(NUMCTY,3.)),LEFT(CTYOBS));
   END;
IF EOF THEN
   CALL SYMPUT('MTOTCT',NUMCTY); /* MAC VAR NO DIF CTYS */
RUN;
%PUT *** MTOTCT=&MTOTCT;    /* DISPLAY NO OF CTYS */
The Generated Macro Variables

One for each countynm, obs/county, and total num of counties.

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCTY1</td>
<td>ASHLAND</td>
</tr>
<tr>
<td>MOBS1</td>
<td>2</td>
</tr>
<tr>
<td>MCTY2</td>
<td>BAYFIELD</td>
</tr>
<tr>
<td>MOBS2</td>
<td>3</td>
</tr>
<tr>
<td>MCTY3</td>
<td>WASHINGTON</td>
</tr>
<tr>
<td>MBOS3</td>
<td>1</td>
</tr>
<tr>
<td>MTOTCT</td>
<td>3</td>
</tr>
</tbody>
</table>
The Macro to Loop Around PROC PRINT

%MACRO COUNTYMC; /* MACRO START */
%DO I=1 %TO &MTOTCT; /* LOOP THRU ALL CTYS */
%PUT *** LOOP &I OF &MTOTCT; /* DISPLAY PROGRESS */
PROC PRINT DATA=COUNTYDT; /* PROC PRINT */
   WHERE COUNTYNM="&&MCTY&I"; /* GENERATED WHERE */
   OPTIONS PAGENO=1; /* RESET PAGENO */
   TITLE "REPORT FOR COUNTY &&MCTY&I"; /* TITLES AND FOOTNOTES */
   FOOTNOTE "TOTAL OBSERVATION COUNT WAS &&MOBS&I";
RUN;
%END; /* END OF %DO */
%MEND COUNTYMC; /* END OF MACRO */
%COUNTYMC /* INVOKE MACRO */
The Generated Code

*** MTOTCT=3
*** LOOP 1 OF 3
   PROC PRINT DATA=COUNTYDT;
   WHERE COUNTYNM="ASHLAND";   OPTIONS PAGENO=1;
   TITLE "REPORT FOR COUNTY ASHLAND";
   FOOTNOTE "TOTAL OBSERVATION COUNT WAS 2";  RUN;
*** LOOP 2 OF 3
   PROC PRINT DATA=COUNTYDT;
   WHERE COUNTYNM="BAYFIELD";   OPTIONS PAGENO=1;
   TITLE "REPORT FOR COUNTY BAYFIELD";
   FOOTNOTE "TOTAL OBSERVATION COUNT WAS 3";  RUN;
*** LOOP 3 OF 3
   PROC PRINT DATA=COUNTYDT;
   WHERE COUNTYNM="WASHINGTON";   OPTIONS PAGENO=1;
   TITLE "REPORT FOR COUNTY WASHINGTON";
   FOOTNOTE "TOTAL OBSERVATION COUNT WAS 1";  RUN;
### The Generated Output

#### REPORT FOR COUNTY **ASHLAND**

<table>
<thead>
<tr>
<th>OBS</th>
<th>COUNTYNM</th>
<th>READING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ASHLAND</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>ASHLAND</td>
<td>611</td>
</tr>
</tbody>
</table>

**TOTAL OBSERVATION COUNT WAS 2**

#### REPORT FOR COUNTY **BAYFIELD**

<table>
<thead>
<tr>
<th>OBS</th>
<th>COUNTYNM</th>
<th>READING</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>BAYFIELD</td>
<td>101</td>
</tr>
<tr>
<td>4</td>
<td>BAYFIELD</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>BAYFIELD</td>
<td>222</td>
</tr>
</tbody>
</table>

**TOTAL OBSERVATION COUNT WAS 3**
## The Generated Output

<table>
<thead>
<tr>
<th>Obs</th>
<th>County</th>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Washington</td>
<td>143</td>
</tr>
</tbody>
</table>

TOTAL OBSERVATION COUNT WAS 1
A Solution Via PROC SQL

PROC SQL can create macro variables.

PROC SQL;
    SELECT  LEFT(PUT(COUNT(DISTINCT COUNTYNM),3.))
            INTO:MTOTCT FROM COUNTYDT;
    /* NO OF UNIQUE CTYS */
    SELECT  DISTINCT COUNTYNM /* EACH CTY NAME */
            INTO:MCTY1-MCTY&MTOTCT FROM COUNTYDT;
    SELECT  COUNT(*) /* OBS PER */
            INTO:MOBS1-MOBS&MTOTCT FROM COUNTYDT
    GROUP BY COUNTYNM;

%PUT *** MTOTCT=&MTOTCT; /* DISPLAY NO OF CTYS */
%MACRO COUNTYMC; /* MACRO START */
%DO I=1 %TO &MTOTCT; /* LOOP THRU ALL CTYS */
%PUT *** LOOP &I OF &MTOTCT; /* DISPLAY PROGRESS */
PROC PRINT DATA=COUNTYDT; /* PROC PRINT */
  WHERE COUNTYNM="&MCTY&I"; /* GENERATED WHERE */
  OPTIONS PAGENO=1; /* RESET PAGENO */
  TITLE "REPORT FOR COUNTY &MCTY&I";
  FOOTNOTE "TOTAL OBSERVATION COUNT WAS &MOBS&I";
RUN;
%END; /* END OF %DO */
%MEND COUNTYMC; /* END OF MACRO */
%COUNTYMC /* INVOKE MACRO */

Notes:
• PROC SQL must produce a report when creating macro variables.
• PROC PRINTTO could route it to a null file.
Exercise 7:

- A very common problem is to run a proc against all members of a SAS library.
- The following program produces a SAS dataset containing an observation for each variable within each dataset in the library. A partial print is shown below.

```sas
PROC CONTENTS DATA=WORK._ALL_ NOPRINT OUT=CONTOUT;
RUN;
PROC PRINT DATA=CONTOUT;
VAR LIBNAME MEMNAME NAME;
TITLE 'CONTOUT';
RUN;
```
The Resulting Dataset

<table>
<thead>
<tr>
<th>OBS</th>
<th>LIBNAME</th>
<th>MEMNAME</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WORK</td>
<td>FREQ</td>
<td>DEPT</td>
</tr>
<tr>
<td>2</td>
<td>WORK</td>
<td>FREQ</td>
<td>SALARIES</td>
</tr>
<tr>
<td>3</td>
<td>WORK</td>
<td>FREQ</td>
<td>SALES</td>
</tr>
<tr>
<td>4</td>
<td>WORK</td>
<td>FREQ</td>
<td>YEAR</td>
</tr>
<tr>
<td>5</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>EMPLOYDT</td>
</tr>
<tr>
<td>6</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>EXPENSES</td>
</tr>
<tr>
<td>7</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>NAME</td>
</tr>
<tr>
<td>8</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>REGION</td>
</tr>
<tr>
<td>9</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>SALES</td>
</tr>
<tr>
<td>10</td>
<td>WORK</td>
<td>YEAR1988</td>
<td>STATE</td>
</tr>
</tbody>
</table>

Directions:

- Write a SAS macro that will generate a separate PROC PRINT of all members in the library with an appropriate title.
- Note you will only want to produce one print per member, not one per variable.
Exercise 7 Solution (first step)

PROC CONTENTS DATA=WORK._ALL_ NOPRINT OUT=CONTOUT;
   RUN;
PROC PRINT DATA=CONTOUT;
   VAR LIBNAME MEMNAME NAME;
   TITLE 'CONTOUT';
   RUN;
Exercise 7 Solution (remainder)

DATA ONEMEM;
  SET CONTOUT END=EOF;
  BY MEMNAME;
  IF LAST.MEMNAME;
  KTR+1;
  CALL SYMPUT ('MMEM'||LEFT(PUT(KTR,5.)),MEMNAME);
  IF EOF;
  CALL SYMPUT ('MTOTOBS',LEFT(PUT(KTR,5.)));"
The Generated SAS Code

PROC PRINT DATA=FREQ;
  TITLE "FREQ";
RUN;
PROC PRINT DATA=YEAR1998;
  TITLE "YEAR1998";
RUN;
The Generated Output

<table>
<thead>
<tr>
<th>FREQ OBS</th>
<th>DEPT</th>
<th>YEAR</th>
<th>SALARIES</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>97</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>98</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>97</td>
<td>500</td>
<td>3000</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>98</td>
<td>600</td>
<td>2000</td>
</tr>
</tbody>
</table>
### The Generated Output

<table>
<thead>
<tr>
<th>YEAR1998</th>
<th>OBS</th>
<th>NAME</th>
<th>STATE</th>
<th>REGION</th>
<th>EMPLOYDT</th>
<th>SALES</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>ANDERSON, CHRIS</td>
<td>FL</td>
<td>1</td>
<td>02/08/93</td>
<td>56,000</td>
<td>12,000</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>PHILLIPS, HENRY</td>
<td>TX</td>
<td>1</td>
<td>04/14/96</td>
<td>63,432</td>
<td>23,500</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>ANDERSEN, JANET</td>
<td>WI</td>
<td>4</td>
<td>06/15/95</td>
<td>101,000</td>
<td>25,000</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>FRANK, TIMOTHY</td>
<td>FL</td>
<td>3</td>
<td>03/02/96</td>
<td>95,900</td>
<td>10,000</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>WILSON, MARGARET</td>
<td>TX</td>
<td>2</td>
<td>06/01/98</td>
<td>15,000</td>
<td>5,000</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>JONES, JACKIE</td>
<td>CA</td>
<td>1</td>
<td>03/14/96</td>
<td>35,000</td>
<td>12,000</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>SMITH, WILLIAM</td>
<td>WI</td>
<td>3</td>
<td>07/01/97</td>
<td>66,666</td>
<td>6,666</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>WILLIAMS, STEVEN</td>
<td>TX</td>
<td>2</td>
<td>04/01/95</td>
<td>43,000</td>
<td>10,000</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>JOHNSON, JOY</td>
<td>CA</td>
<td>1</td>
<td>05/10/96</td>
<td>20,000</td>
<td>4,000</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>MATHERS, MARK</td>
<td>WI</td>
<td>3</td>
<td>01/15/94</td>
<td>55,555</td>
<td>13,500</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>JENSEN, LORI</td>
<td>FL</td>
<td>3</td>
<td>10/10/96</td>
<td>103,500</td>
<td>30,000</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>O'HARE, PATTY</td>
<td>WI</td>
<td>1</td>
<td>02/28/98</td>
<td>25,000</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>FRANKLIN, SUSAN</td>
<td>CA</td>
<td>1</td>
<td>09/30/92</td>
<td>95,000</td>
<td>15,000</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>HARRISON, ANTHONY</td>
<td>WI</td>
<td>2</td>
<td>11/15/95</td>
<td>45,900</td>
<td>10,000</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>THOMPSON, ELIZABETH</td>
<td>FL</td>
<td>3</td>
<td>12/10/97</td>
<td>10,000</td>
<td>500</td>
</tr>
</tbody>
</table>
The SSCFLAT Macro

A general purpose macro:
- converts any SAS ds to a comma delimited file for input to spreadsheets etc.
- will run on all platforms
- automatically reads dictionary tables to get ds definition
- honors SAS formatting
- can create a header line
- dropping, reordering variables etc. can be done in an earlier SAS step.
An Introduction to SAS Macros

The SSCFLAT Macro Partial Source

/***********************************************************/
/* MACRO SSCFLAT VERSION 1.4 */
/* CREATES A COMMA DELIMITED FILE FROM ANY SAS DATA SET */
/* IT CAN BE THEN DOWNLOADED & IMPORTED INTO A SPREADSHEET*/
/* */
/* SAMPLE WINDOWS CALL: */
/* %SSCFLAT(MSASDS=MAIL.TEMPMAIL.MPREFIX=C:\TEMP\) */
/* */
/* SAMPLE MVS CALL: */
/* %SSCFLAT(MSASDS=WORK.XYZ.MPREFIX=MYUSER) */
/* */
/* STEVEN FIRST */
/* (C) SYSTEMS SEMINAR CONSULTANTS 1999 608 278-9964 */
/* */
/* PERMISSION GIVEN TO FORMER SSC SAS STUDENTS TO USE */
/* IN PERSONAL SAS JOBS. */
/* FOR PERMISSION TO DISTRIBUTE TO OTHERS, OR FOR */
/* COMPANY WIDE USE, CONTACT SSC AT 608 278-9964 */
/* */
/***********************************************************/
SSCFLAT Macro Partial Source (continued)

```sas
%MACRO SSCFLAT(MSASDS=, /*INPUT SAS DS (REQUIRED */
   MPREFIX=&SYSPref.., /*OUT PREF, OR DIR OUT */
   MFLATOUT=&MPREFIX&MMEMNAME..DAT, /*FLATFILE OUT */
   MHEADER=YES, /*FIELD NAMES IN FIRST REC */
   MLIST=YES, /*PRINT FLAT FILE IN LOG? */
   MTRIMNUM=YES, /*TRIM NUM TRAIL BLANKS? */
   MTRACE=NO, /*DEBUGGING OPTION */
   MMISSING=".", /*MISSING VALUE CHARACTER */
   MLRECL=6160, /*LARGEST RECORD LENGTH */
   MVSOPT=UNIT=3390, /*MVS UNIT OPTIONS */
   MSPACE=1, /*MVS SPACE IN CYLS */
);

%PUT ***** SSCMAC COPYRIGHT (C) 1999 SYSTEMS SEMINAR;
%PUT ***** CONSULTANTS 608 278-9964;
```
A SSCFLAT Macro Example

In windows, convert WORK.ADDRDATA to a FLAT file.

<table>
<thead>
<tr>
<th>OBS</th>
<th>NAMES</th>
<th>DEPT</th>
<th>AGE</th>
<th>RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STEVE</td>
<td>ACCT</td>
<td>43</td>
<td>12.22</td>
</tr>
<tr>
<td>2</td>
<td>DAVID</td>
<td>PAYR</td>
<td></td>
<td>11.21</td>
</tr>
</tbody>
</table>

%INC ‘insert file ref\sscflat.sas.’;
%SSCFLAT(MSASDS=WORK.ADDRDATA,
    mprefix=c:\temp\) *invoke macro;
A SSCFLAT Macro Example (continued)

The c:\temp\addrdata.dat flat file created:

```
"NAME", "DEPT", "AGE", "RATE"
"STEVE", "ACCT ", 43, 12.22
"DAVID", "PAYR ", 11.21
```

The flat file is now available for download and/or import to spreadsheets, etc.
An Introduction to SAS Macros