

The PRINT Procedure

Overview 783 Procedure Syntax 785 PROC PRINT Statement 786 BY Statement 791 ID Statement 792 PAGEBY Statement 793 SUM Statement 793 SUMBY Statement 794 VAR Statement 795 Results 795 Procedure Output 795 Page Layout 795 Column Headings 797 Column Width 797 Examples 798 Example 1: Selecting Variables to Print 798 Example 2: Customizing Text in Column Headers 799 Example 3: Creating Separate Sections of a Report for Groups of Observations 801 Example 4: Summing Numeric Variables with One BY Group 803 Example 5: Summing Numeric Variables with Multiple BY Variables 805 Example 6: Limiting the Number of Sums in a Report 808 Example 7: Controlling the Layout of a Report with Many Variables 810 Example 8: Creating a Customized Layout with BY Groups and ID Variables 813 Example 9: Printing All the Data Sets in a SAS Library 815

Overview

The PRINT procedure prints the observations in a SAS data set, using all or some of the variables. You can create a variety of reports ranging from a simple listing to a highly customized report that groups the data and calculates totals and subtotals for numeric variables.

Output 28.1 on page 783 illustrates the simplest kind of report that you can produce. The statements that produce the output follow. A DATA step on page 798 creates the data set EXPREV.

```
options nodate pageno=1 linesize=64
        pagesize=60;
proc print data=exprev;
run;
```

Output 28.1 Simple Report Produced with PROC PRINT

The SAS System							
Obs	Region	State	Month	Expenses	Revenues		
1	Southern	GA	JAN95	2000	8000		
2	Southern	GA	FEB95	1200	6000		
3	Southern	FL	FEB95	8500	11000		
4	Northern	NY	FEB95	3000	4000		
5	Northern	NY	MAR95	6000	5000		
6	Southern	FL	MAR95	9800	13500		
7	Northern	MA	MAR95	1500	1000		

Output 28.2 on page 784 is a customized report that is produced by PROC PRINT. The statements that create this report

- $\hfill\square$ customize the title and the column headings
- $\hfill\square$ place dollar signs and commas in numeric output
- $\hfill\square$ selectively include and control the order of variables in the report

 $\hfill\square$ group the data by JobCode

 $\hfill\square$ sum the values for Salary for each job code and for all job codes.

For an explanation of the program that produces this report, see Example 8 on page 813.

1

Ex Salaries for	xpenses Incu Flight Atte	urred for endants and Mechanics	
Job Code =======	Gender =====	Annual Salary	
FA1	F F	\$23,177.00 \$22,454.00	
	М	\$22,268.00	
FA1		\$67,899.00	
FA2	F F	\$28,888.00 \$27,787.00	
 FA2	М	\$28,572.00	
FA3	F	\$32,886.00	
	F M	\$33,419.00 \$32,217.00	
FA3		\$98,522.00	
ME1	M M M	\$29,769.00 \$28,072.00 \$28.619.00	
 ME1		\$86,460.00	
ME2	F F M	\$35,108.00 \$34,929.00 \$35,345.00	
	M M M	\$36,925.00 \$35,090.00 \$35,185.00	
 ME2		\$212,582.00	
ME 3	М	\$43,025.00	
		\$593,735.00	

Output 28.2 Customized Report Produced with PROC PRINT

Procedure Syntax

Tip: Supports the Output Delivery System (see Chapter 2, "Fundamental Concepts for Using Base SAS Procedures")

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," for details. You can also use any global statements as well. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC PRINT <*option(s)*>;

BY <DESCENDING> variable-1 <...<DESCENDING> variable-n> <NOTSORTED>;

PAGEBY BY-variable; SUMBY BY-variable; ID variable(s); SUM variable(s); VAR variable(s);

To do this	Use this statement
Produce a separate section of the report for each BY group	ВҮ
Identify observations by the formatted values of the variables that you list instead of by observation numbers	ID
Control page ejects that occur before a page is full	PAGEBY
Limit the number of sums that appear in the report	SUMBY
Total values of numeric variables	SUM
Select variables that appear in the report and determine their order	VAR

PROC PRINT Statement

PROC PRINT <*option(s)*>;

To d	o this	Use this option
Spec	ify the input data set	DATA=
Cont	crol general format	
	Write a blank line between observations	DOUBLE
	Print the number of observations in the data set, in BY groups, or both, and specify explanatory text to print with the number	N=
	Suppress the column in the output that identifies each observation by number	NOOBS
	Specify a column header for the column that identifies each observation by number	OBS=
	Round unformatted numeric values to two decimal places	ROUND
Cont	crol page format	
	Format the rows on a page	ROWS=
	Use each variable's formatted width as its column width on all pages	WIDTH=UNIFORM

To c	lo this	Use this option
Con	trol column format	
	Control the orientation of the column headings	HEADING=
	Use variables' labels as column headings	LABEL or SPLIT=
	Specify the split character, which controls line breaks in column headings	SPLIT=
	Determine the column width for each variable	WIDTH=

Options

DATA=SAS-data-set

specifies the SAS data set to print.

Main discussion: "Input Data Sets" on page 18

DOUBLE

writes a blank line between observations.

Alias: D

Restriction: This option has no effect on the HTML output.

Featured in: Example 1 on page 798

HEADING=direction

controls the orientation of the column headings, where *direction* is one of the following:

HORIZONTAL

prints all column headings horizontally.

Alias: H

VERTICAL

prints all column headings vertically.

Alias: V

- **Default:** Headings are either all horizontal or all vertical. If you omit HEADING=, PROC PRINT determines the direction of the column headings as follows:
 - □ If you do not use LABEL, spacing dictates whether column headings are vertical or horizontal.
 - □ If you use LABEL and at least one variable has a label, all headings are horizontal.

LABEL

uses variables' labels as column headings.

Alias: L

- **Default:** If you omit LABEL, PROC PRINT uses the variable's name as the column heading even if the PROC PRINT step contains a LABEL statement. If a variable does not have a label, PROC PRINT uses the variable's name as the column heading.
- **Interaction:** By default, if you specify LABEL and at least one variable has a label, PROC PRINT prints all column headings horizontally. Therefore, using LABEL may increase the number of pages of output. (Use HEADING=VERTICAL in the PROC PRINT statement to print vertical column headings.)

- **Interaction:** PROC PRINT sometimes conserves space by splitting labels across multiple lines. Use SPLIT= in the PROC PRINT statement to control where these splits occur. You do not need to use LABEL if you use SPLIT=.
- **Tip:** To create a blank column header for a variable, use this LABEL statement in your PROC PRINT step:

label variable-name='00'x;

See also: For information on using the LABEL statement to create temporary labels in procedures see Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 67.

For information on using the LABEL statement in a DATA step to create permanent labels, see the section on statements in SAS Language Reference: Dictionary.

Featured in: Example 3 on page 801

Note: The SAS system option LABEL must be in effect in order for any procedure to use labels. For more information see the section on system options in SAS Language Reference: Dictionary \triangle

N<="string-1" <"string-2">>

prints the number of observations in the data set, in BY groups, or both and specifies explanatory text to print with the number.

If you use the	
N option	PROC PRINT
with neither a BY nor a SUM statement	prints the number of observations in the data set at the end of the report and labels the number with the value of <i>string-1</i> .
with a BY statement	prints the number of observations in the BY group at the end of each BY group and labels the number with the value of <i>string-1</i> .
with a BY statement and a SUM statement	prints the number of observations in the BY group at the end of each BY group and prints the number of observations in the data set at the end of the report. The numbers for BY groups are labeled with <i>string-1</i> ; the number for the entire data set is labeled with <i>string-2</i> .

Featured in: Example 2 on page 799 (alone)

Example 3 on page 801 (with a BY statement)

Example 4 on page 803 (with a BY statement and a SUM statement)

NOOBS

suppresses the observation number in the output. **Featured in:** Example 3 on page 801

OBS="column-header"

specifies a column header for the column that identifies each observation by number. **Tip:** OBS= honors the split character (see the discussion of SPLIT= on page 790). **Featured in:** Example 2 on page 799

ROUND

rounds unformatted numeric values to two decimal places. (Formatted values are already rounded by the format to the specified number of decimal places.) For both

formatted and unformatted variables, PROC PRINT uses these rounded values to calculate any sums in the report.

If you omit ROUND, PROC PRINT adds the actual values of the rows to obtain the sum *even though it displays the formatted (rounded) values*. Any sums are also rounded by the format, but they include only one rounding error, that of rounding the sum of the actual values. The ROUND option, on the other hand, rounds values before summing them, so there may be multiple rounding errors. The results without ROUND are more accurate, but ROUND is useful for published reports where it is important for the total to be the sum of the printed (rounded) values.

Be aware that the results from PROC PRINT with the ROUND option may differ from the results of summing the same data with other methods such as PROC MEANS or the DATA step. Consider a simple case in which

 \Box the data set contains three values for X: .003, .004, and .009.

 \square X has a format of 5.2.

Depending on how you calculate the sum, you can get three different answers: 0.02, 0.01, and 0.016. Figure 28.1 on page 789 shows the results of calculating the sum with PROC PRINT (without and with the ROUND option) and PROC MEANS.

Figure 28.1 Three Methods of Summing Variables

Actual Values	PROC PI the ROU	RINT without ND option		PROC the RO	PRINT with DUND option	PROC MEANS
	OBS	Х		OBS	Х	 Analysis Variable : X
.003	1	0.00	11	1	0.00	Sum
.004 .009	2 3	0.00 0.01		2 3	0.00 0.01	0.0160000
===== .016		0.02			===== 0.01	

Notice that the sum produced without the ROUND option (.02) is closer to the actual result (0.16) than the sum produced with ROUND (0.01). However, the sum produced with ROUND reflects the numbers displayed in the report.

Alias: R

CAUTION:

Do not use ROUND with PICTURE formats. ROUND is for use with numeric values. SAS procedures treat variables that have picture formats as character variables. Using ROUND with such variables may lead to unexpected results. \triangle

ROWS=page-format

formats rows on a page. Currently, PAGE is the only value that you can use for *page-format*:

PAGE

prints only one row of variables for each observation per page. When you use ROWS=PAGE, PROC PRINT does not divide the page into sections; it prints as many observations as possible on each page. If the observations do not fill the last page of the output, PROC PRINT divides the last page into sections and prints all the variables for the last few observations.

- **Restriction:** Physical page size does not mean the same thing in HTML output as it does in traditional procedure output. Therefore, HTML output from PROC PRINT appears the same whether or not you use ROWS=.
- **Tip:** The PAGE value can reduce the number of pages in the output if the data set contains large numbers of variables and observations. However, if the data set contains a large number of variables but few observations, the PAGE value can increase the number of pages in the output.

See also: "Page Layout" on page 795 for discussion of the default layout.

Featured in: Example 7 on page 810

SPLIT='split-character'

specifies the split character, which controls line breaks in column headers. It also uses labels as column headers. PROC PRINT breaks a column heading when it reaches the split character and continues the header on the next line. The split character is not part of the column heading although each occurrence of the split character counts toward the 256-character maximum for a label.

Alias: S=

Interaction: You do not need to use both LABEL and SPLIT= because SPLIT= implies the use of labels.

Interaction: The OBS= option honors the split character. (See the discussion of OBS= on page 788).

Featured in: Example 2 on page 799

Note: PROC PRINT does not split labels of BY variables in the heading preceding each BY group even if you specify SPLIT=. Instead, PROC PRINT treats the split character as part of the label. Therefore, you probably want to avoid using a split character when you are using the BY statement. \triangle

UNIFORM

See WIDTH=UNIFORM on page 790.

WIDTH=column-width

determines the column width for each variable. The value of *column-width* must be one of the following:

FULL

uses a variable's formatted width as the column width. If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the default width. For a character variable, the default width is the length of the variable. For a numeric variable, the default width is 12. When you use WIDTH=FULL, the column widths do not vary from page to page.

Tip: Using WIDTH=FULL can reduce execution time.

MINIMUM

uses for each variable the minimum column width that accommodates all values of the variable.

Alias: MIN

UNIFORM

uses each variable's formatted width as its column width on all pages. If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value as the column width. When you specify WIDTH=UNIFORM, PROC PRINT normally needs to read the data set twice. However, if all the variables in the data set have formats that explicitly specify a field width (for example, BEST12. but not BEST.), PROC PRINT reads the data set only once.

Alias: U

- Tip: If the data set is large and you want a uniform report, you can save computer resources by using formats that explicitly specify a field width so that PROC PRINT reads the data only once.
- Tip: WIDTH=UNIFORM is the same as UNIFORM.
- Restriction: When not all variables have formats that explicitly specify a width, you cannot use WIDTH=UNIFORM with an engine that supports concurrent access if another user is updating the data set at the same time.
- UNIFORMBY

formats all columns uniformly within a BY group, using each variable's formatted width as its column width. If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value as the column width.

Alias: UBY

Restriction: You cannot use UNIFORMBY with a sequential data set.

- **Default:** If you omit WIDTH= and do not specify the UNIFORM option, PROC PRINT individually constructs each page of output. The procedure analyzes the data for a page and decides how best to display them. Therefore, column widths may differ from one page to another.
- **Tip:** Column width is affected not only by variable width but also by the length of column headings. Long column headings may lessen the usefulness of WIDTH=.
- **See also:** For a discussion of default column widths, see "Column Width" on page 797.

BY Statement

Produces a separate section of the report for each BY group.

Main discussion: "BY" on page 68

Featured in: Example 3 on page 801, Example 4 on page 803, Example 5 on page 805, Example 6 on page 808, and Example 8 on page 813

BY <DESCENDING> variable-1 <...<DESCENDING> variable-n> <NOTSORTED>;

Required Arguments

variable

specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING

specifies that the data set is sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED

specifies that observations are not necessarily sorted in alphabetic or numeric order. The data are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

Using the BY Statement with an ID Statement

PROC PRINT uses a special layout if all BY variables appear in the same order at the beginning of the ID statement. (See Example 8 on page 813.)

Using the BY Statement with the NOBYLINE Option

If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output produced with BY-group processing, PROC PRINT always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, the information in the titles matches the report on the pages. (See "Creating Titles That Contain BY-Group Information" on page 54.)

ID Statement

Identifies observations by using the formatted values of the variables that you list instead of by using observation numbers.

Featured in: Example 7 on page 810 and Example 8 on page 813

ID *variable(s)*;

Required Arguments

variable(s)

specifies one or more variables to print instead of the observation number at the beginning of each row of the report.

Restriction: If the ID variables occupy so much space that no room remains on the line for at least one other variable, PROC PRINT writes a warning to the SAS log and does not treat all ID variables as ID variables.

Interaction: If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable.

Using the BY Statement with an ID Statement

PROC PRINT uses a special layout if all BY variables appear in the same order at the beginning of the ID statement. (See Example 8 on page 813.)

PAGEBY Statement

Controls page ejects that occur before a page is full.

Requirements: BY statement Featured in: Example 3 on page 801

PAGEBY BY-variable;

Required Arguments

BY-variable

identifies a variable appearing in the BY statement in the PROC PRINT step. If the value of the BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT begins printing a new page.

Interaction: If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output produced with BY-group processing, PROC PRINT always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, the information in the titles matches the report on the pages. (See "Creating Titles That Contain BY-Group Information" on page 54.)

SUM Statement

Totals values of numeric variables.

Featured in: Example 4 on page 803, Example 5 on page 805, Example 6 on page 808, and Example 8 on page 813

SUM *variable(s)*;

Required Arguments

variable(s)

identifies the numeric variables to total in the report.

Using the SUM and BY Statements Together

When you use a SUM statement and a BY statement with one BY variable, PROC PRINT sums the SUM variables for each BY group that contains more than one observation and totals them over all BY groups (see Example 4 on page 803).

When you use a SUM statement and a BY statement with multiple BY variables, PROC PRINT sums the SUM variables for each BY group that contains more than one observation, just as it does if you use only one BY variable. However, it provides sums only for those BY variables whose values change when the BY group changes. (See Example 5 on page 805.)

Note: When the value of a BY variable changes, the SAS System considers that the values of all variables listed after it in the BY statement also change. \triangle

SUMBY Statement

Limits the number of sums that appear in the report.

Requirements: BY statement Featured in: Example 6 on page 808

SUMBY *BY*-variable;

Required Arguments

BY-variable

identifies a variable that appears in the BY statement in the PROC PRINT step. If the value of the BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT prints the sums of all variables listed in the SUM statement.

What Variables Are Summed?

If you use a SUM statement, PROC PRINT subtotals only the SUM variables. Otherwise, PROC PRINT subtotals all the numeric variables in the data set except those listed in the ID and BY statements.

VAR Statement

Selects variables that appear in the report and determines their order.

Tip: If you omit the VAR statement, PROC PRINT prints all variables in the data set. Featured in: Example 1 on page 798 and Example 8 on page 813

VAR *variable(s)*;

Required Arguments

variable(s)

identifies the variables to print. PROC PRINT prints the variables in the order that you list them.

Interaction: In the PROC PRINT output, variables that are listed in the ID statement precede variables that are listed in the VAR statement. If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable.

Results

Procedure Output

PROC PRINT always produces a printed report. You control the appearance of the report with statements and options. See "Examples" on page 798 for a sampling of the types of reports that the procedure produces.

Page Layout

By default, PROC PRINT uses an identical layout for all observations on a page of output. First, it attempts to print observations on a single line (see Figure 28.2 on page 795).

Figure 28.2 Printing Observations on a Single Line

1 Obs Var 2 Var 1 Var 3 1 ~~~~ ~~~~ ~~~~ $\mathbf{2}$ ~~~~ ~~~~ 3 ~~~~ 4 ~~~~ ~~~~ ~~~~ $\mathbf{5}$ ~~~~ ~~~~ 6

If PROC PRINT cannot fit all the variables on a single line, it splits the observations into two or more sections and prints the observation number or the ID variables at the beginning of each line. For example, in Figure 28.3 on page 796, PROC PRINT prints the values for the first three variables in the first section of each page and the values for the second three variables in the second section of each page.



					1			
Obs	Var_1	Var	2	Var_3				
1	~~~~	~~;	~~	~~~~				
2	~~~~	~~·	~~	~~~~				
3	~~~~	~~·	~~	~~~~				
								0
Obs	Var_4	Var	_5	Var_6				2
1	~~~~	~~	~~	~~~~		r_2	Var_3	
						~~	~~~~	
2	~~~~	~~·	~~	~~~~			~~~~	
3	~~~~	~~-	~~	~~~~				
						_~~	~~~~	
			Ob	s Vai	:_4 V	/ar_5	Var_6	
			4	~~	~~ ~	~~~~	~~~~	
			5	~~	~~ ~	~~~~	~~~~	
			6	~~	~~ ~	~~~~	~~~~	

If PROC PRINT cannot fit all the variables on one page, the procedure prints subsequent pages with the same observations until it has printed all the variables. For example, in Figure 28.4 on page 796, PROC PRINT uses the first two pages to print values for the first three observations and the second two pages to print values for the rest of the observations.

Figure 28.4 Splitting Observations across Multiple Pages

				1					
01	¥7 1	W 0	V 9	1					2
Obs	var_1	var_z	var_5		Obs	Var 7	Var 8	Var 9	_
1	~~~~	~~~~	~~~~		1	var_i	var_o	var_o	
2	~~~~	~~~~	~~~~		1	~~~~	~~~~	~~~~	
3	~~~~	~~~~	~~~~		2	~~~~	~~~~	~~~~	
Obs	Var_4	Var_5	Var_6		3	~~~~	~~~~	~~~~	
1	~~~~	~~~~	~~~~		Obs	Var_10	Var_11	Var_12	
2	~~~~	~~~~	~~~~		1	~~~~	~~~~	~~~~	
3	~~~~	~~~~	~~~~		2	~~~~	~~~~	~~~~	
					3	~~~~	~~~~	~~~~	
				L					
				3					
Obs	Var_1	Var_2	Var_3						4
4	~~~~	~~~~	~~~~		Obs	Var_7	Var 8	Var 9	
5									
	~~~~	~~~~	~~~~		4	~~~~	~~~~	~~~~	
6	~~~~	~~~~	~~~~		4 5	~~~~	~~~~	~~~~	
6	~~~~	~~~~	~~~~		4 5 6	~~~~	~~~~	~~~~	
6 Obs	~~~~ Var_4	~~~~ ~~~~ Var_5	~~~~ ~~~~ Var_6		4 5 6	~~~~	~~~~	~~~~	
6 Obs 4	~~~~ Var_4	~~~~ Var_5 ~~~~	~~~~ Var_6 ~~~~		4 5 6 Obs	~~~~ ~~~~ Var_10	~~~~ ~~~~ Var_11	~~~~ ~~~~ Var_12	
6 Obs 4 5	~~~~ Var_4 ~~~~	~~~~ Var_5 ~~~~	~~~~ Var_6 ~~~~		4 5 6 Obs 4	~~~~ ~~~~ Var_10	  Var_11	  Var_12	
6 Obs 4 5 6	~~~~ Var_4 ~~~~	~~~~ Var_5 ~~~~	~~~~ Var_6 ~~~~		4 5 6 Obs 4 5	  Var_10 	 Var_11	Var_12	
6 Obs 4 5 6	Var_4	~~~~ Var_5 ~~~~	~~~~ Var_6 ~~~~		4 5 6 Obs 4 5 6	 Var_10 	Var_11	Var_12	

*Note:* You can alter the page layout with the ROWS= option in the PROC PRINT statement (see the discussion of ROWS= on page 789).  $\triangle$ 

*Note:* PROC PRINT may produce slightly different output if the data set is not RADIX addressable. Version 6 compressed files are not RADIX addressable, while, beginning with Version 7, compressed files are RADIX addressable. (The integrity of the data is not compromised; the procedure simply numbers the observations differently.)  $\triangle$ 

### **Column Headings**

By default, spacing dictates whether PROC PRINT prints column headings horizontally or vertically. Figure 28.2 on page 795, Figure 28.3 on page 796, and Figure 28.4 on page 796 all illustrate horizontal headings. Figure 28.5 on page 797 illustrates vertical headings.

Figure 28.5 Using Vertical Headings

				1
	v	V	v	
	а	а	а	
0	r	r	r	
b	-	-	-	
s	1	<b>2</b>	3	
1	~~~~	~~~~	~~~~	
2	~~~~	~~~~	~~~~	
3	~~~~	~~~~	~~~~	
4	~~~~	~~~~	~~~~	
5	~~~~	~~~~	~~~~	
6	~~~~	~~~~	~~~~	

*Note:* If you use LABEL and at least one variable has a label, PROC PRINT prints all column headings horizontally unless you specify HEADING=VERTICAL.  $\triangle$ 

### **Column Width**

By default, PROC PRINT uses a variable's formatted width as the column width. (The WIDTH= option overrides this default behavior.) If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value for that variable on that page as the column width.

If the formatted value of a character variable or the data width of an unformatted character variable exceeds the linesize minus the length of all the ID variables, PROC PRINT may truncate the value. Consider the following situation:

- $\Box$  The linesize is 80.
- $\Box$  IdNumber is a character variable with a length of 10. It is used as an ID variable.
- $\Box$  State is a character variable with a length of 2. It is used as an ID variable.
- $\Box$  Comment is a character variable with a length of 200.

When PROC PRINT prints these three variables on a line, it uses 14 print positions for the two ID variables and the space after each one. This leaves 80–14, or 66, print positions for COMMENT. Longer values of COMMENT are truncated.

WIDTH= controls the column width.

*Note:* Column width is affected not only by variable width but also by the length of column headings. Long column headings may lessen the usefulness of WIDTH=.  $\triangle$ 

## **Examples**

## **Example 1: Selecting Variables to Print**

Procedure features:

PROC PRINT statement options: DOUBLE VAR statement

This example

 $\hfill\square$  selects three variables for the report

□ uses variable labels as column headings

 $\Box$  double spaces between rows of the report.

## **Program**

options nodate pageno=1 linesize=70 pagesize=60;

The data set EXPREV contains information on a company's monthly expenses and revenues for two regions of the United States.

```
data exprev;
    input Region $ State $ Month monyy5.
        Expenses Revenues;
    format month monyy5.;
    datalines;
Southern GA JAN95 2000 8000
Southern GA FEB95 1200 6000
Southern FL FEB95 8500 11000
Northern NY FEB95 3000 4000
Northern NY MAR95 6000 5000
Southern FL MAR95 9800 13500
Northern MA MAR95 1500 1000
;
```

DOUBLE writes a blank line between observations. (This option has no effect on the HTML output.)

proc print data=exprev double;

The VAR statement creates columns for Month, State, and Expenses, in that order.

```
var month state expenses;
```

The TITLE statement specifies a title for the report.

```
title 'Monthly Expenses for Offices in Each State';
run;
```

### Output

By default, PROC PRINT identifies each observation by number under the column heading Obs.

Monthly	y Expenses	for Offices	in Each S	tate 1
Obs	s Month	State	Expenses	
1	JAN95	GA	2000	
2	FEB95	GA	1200	
3	FEB95	FL	8500	
4	FEB95	NY	3000	
5	MAR95	NY	6000	
6	MAR95	FL	9800	
7	MAR95	MA	1500	

## **Example 2: Customizing Text in Column Headers**

Procedure features: PROC PRINT statement options: N OBS= SPLIT= Other features: LABEL statement Data set: EXPREV on page 798

This example

- $\hfill\square$  customizes and underlines the text in column headings for variables
- $\hfill\square$  customizes the column header for the column that identifies observations by number
- $\hfill\square$  shows the number of observations in the report

 $\Box$  writes the values of Expenses with commas.

## Program

options nodate pageno=1 linesize=70 pagesize=60;

SPLIT= identifies the asterisk as the character that starts a new line in column headers. The N option prints the number of observations at the end of the report. OBS= specifies the column header for the column that identifies each observation by number. The split character (*) starts a new line in the column heading. Therefore, the equal signs (=) in the value of OBS= underline the column header.

proc print data=exprev split='*' n obs='Observation*Number*==========;;

The VAR statement creates columns for Month, State, and Expenses, in that order.

var month state expenses;

The LABEL statement associates a label with each variable for the duration of the PROC PRINT step. When you use SPLIT= in the PROC PRINT statement, the procedure uses labels for column headers. The split character (*) starts a new line in the column heading. Therefore, the equal signs (=) in the labels underline the column headers.

```
label month='Month**====='
state='State**====='
expenses='Expenses**=======';
```

The FORMAT statement assigns a format to use for Expenses in the report. The TITLE statement specifies a title.

```
format expenses commal0.;
title 'Monthly Expenses for Offices in Each State';
run:
```

## Output

## Example 3: Creating Separate Sections of a Report for Groups of Observations

Procedure features: PROC PRINT statement options: LABEL N= NOOBS BY statement PAGEBY statement Other features: SORT procedure LABEL statement Data set: EXPREV on page 798

This example

- $\hfill\square$  suppresses the printing of observation numbers at the beginning of each row
- $\Box$  presents the data for each state in a separate section of the report
- $\Box$  begins a new page for each region.

## Program

```
options pagesize=60 pageno=1 nodate linesize=70;
```

PROC SORT sorts the observations by Region, State, and Month.

```
proc sort data=exprev;
    by region state month;
```

#### run;

N= prints the number of observations in a BY group at the end of that BY group. The explanatory text that the N= option provides precedes the number. NOOBS suppresses the printing of observation numbers at the beginning of the rows. LABEL uses variables' labels as column headings.

The VAR statement creates columns for Month, Expenses, and Revenues, in that order.

var month expenses revenues;

The BY statement produces a separate section of the report for each BY group and prints a heading above each one. The PAGEBY statement starts a new page each time the value of Region changes.

```
by region state;
pageby region;
```

The LABEL statement associates a label with the variable Region for the duration of the PROC PRINT step. When you use the LABEL option in the PROC PRINT statement, the procedure uses labels for column headings.

```
label region='Sales Region';
```

The FORMAT statement assigns a format to Expenses and Revenues for this report. The TITLE statement specifies a title.

```
format revenues expenses commal0.;
  title 'Sales Figures Grouped by Region and State';
run;
```

## Output

Sales Figur	es Grouped by Reg	gion and State	e 1
Sales	Region=Northern	State=MA	
Month	Expenses	Revenues	
MAR95	1,500	1,000	
Number of	observations for	the state: 1	
Sales	Region=Northern	State=NY	
Month	Expenses	Revenues	
FEB95 MAR95	3,000 6,000	4,000 5,000	
Number of	observations for	the state: 2	

Sales	Figures Grou	ped by Regi	on and Stat	e	2
	Sales Regior	n=Southern S	tate=FL		
Мо	onth Exp	benses	Revenues		
FE	B95	8,500	11,000		
Numbe	er of observa	tions for t	he state: 2	!	
	Sales Regior	Southern S	tate=GA		
Мо	onth Exp	benses	Revenues		
JA	N95	2,000	8,000		
FE	B95	1,200	6,000		
Numbe	er of observa	ations for t	he state: 2	:	

# **Example 4: Summing Numeric Variables with One BY Group**

Procedure features: PROC PRINT statement options: N= BY statement SUM statement Other features: SORT procedure TITLE statement #BYVAL specification SAS system options : BYLINE NOBYLINE Data set: EXPREV on page 798

This example

- □ sums expenses and revenues for each region and for all regions
- $\Box$  shows the number of observations in each BY group and in the whole report
- □ creates a customized title, containing the name of the region. This title replaces the default BY line for each BY group.

### Program

The SAS system option NOBYLINE suppresses the printing of the default BY line. When you use PROC PRINT with NOBYLINE, each BY group starts on a new page.

options nodate pageno=1 linesize=70 pagesize=60 nobyline;

PROC SORT sorts the observations by Region.

```
proc sort data=exprev;
    by region;
run;
```

NOOBS suppresses the printing of observation numbers at the beginning of the rows. N= prints the number of observations in a BY group at the end of that BY group and (because of the SUM statement) prints the number of observations in the data set at the end of the report. The first piece of explanatory text that N= provides precedes the number for each BY group. The second piece of explanatory text that N= provides precedes the number for the entire data set.

The SUM statement alone sums the values of Expenses and Revenues for the entire data set. Because the PROC PRINT step contains a BY statement, the SUM statement also sums the values of Expenses and Revenues for each region that contains more than one observation.

```
sum expenses revenues;
by region;
```

The FORMAT statement assigns the COMMA10. format to Expenses and Revenues for this report.

format revenues expenses comma10.;

The TITLE statement specifies a title. The #BYVAL specification places the current value of the BY variable Region in the title. Because NOBYLINE is in effect, each BY group starts on a new page, and the title serves as a BY line.

```
title 'Revenue and Expense Totals for the
#byval(region) Region';
run;
```

The SAS system option BYLINE resets the printing of the default BY line.

options byline;

## Output

State	Month	Expenses	Revenues	
NY	FEB95	3,000	4,000	
NY	MAR95	6,000	5,000	
MA	MAR95	1,500	1,000	
	-			
Region	n	10,500	10,000	

	Revenue and	Expense	Totals for	the	Southern Region	2
	State	Month	Expense	S	Revenues	
	GA	JAN95	2,00	0	8,000	
	GA	FEB95	1,20	0	6,000	
	FL	FEB95	8,50	0	11,000	
	FL	MAR95	9,80	0	13,500	
				-		
	Region		21,50	0	38,500	
	-		========	=	=======	
			32,00	0	48,500	
Number Number of	of observat: observations	ions for s for the	the state: e data set:	4 7		

# **Example 5: Summing Numeric Variables with Multiple BY Variables**

Procedure features: BY statement SUM statement

Other features: SORT procedure

Data set: EXPREV on page 798

This example

□ sums expenses and revenues for

- $\square$  each region
- $\Box$  each state with more than one row in the report
- $\Box$  all rows in the report.
- $\Box$  shows the number of observations in each BY group and in the whole report.

### Program

```
options nodate pageno=1 linesize=70 pagesize=60;
```

PROC SORT sorts the observations by Region and State.

```
proc sort data=exprev;
    by region state;
run;
```

The N option prints the number of observations in a BY group at the end of that BY group and prints the total number of observations used in the report at the bottom of the report. NOOBS suppresses the printing of observation numbers at the beginning of the rows.

proc print data=exprev n noobs;

The BY statement produces a separate section of the report for each BY group. The SUM statement alone sums the values of Expenses and Revenues for the entire data set. Because the program contains a BY statement, the SUM statement also sums the values of Expenses and Revenues for each BY group that contains more than one observation.

```
by region state;
sum expenses revenues;
```

The LABEL statement associates a label with the variable Region for the duration of the PROC PRINT step. The BY line at the beginning of each BY group uses the label. The FORMAT statement assigns a format to Expenses and Revenues for this report. The TITLE statement specifies a title.

```
label region='Sales Region';
format revenues expenses commal0.;
title 'Revenue and Expense Totals for Each State and Region';
run;
```

## Output

The report uses default column headers (variable names) because neither the SPLIT= nor the LABEL option is used. Nevertheless, the BY line at the top of each section of the report shows the BY variables' labels and their values. The name of a BY variable identifies the subtotals in the report.

PROC PRINT sums Expenses and Revenues for each BY group that contains more than one observation. However, sums are shown only for the BY variables whose values change from one BY group to the next. For example, in the third BY group, where the sales region is **Southern** and the state is **FL**, expenses and revenues are summed only for the state because the next BY group is for the same region.

Revenue and Exper	nse Totals for Ea	ach State and	Region 1
Sales	Region=Northern	State=MA	
Month	Expenses	Revenues	
MAR95	1,500	1,000	
	N = 1		
Sales	Region=Northern	State=NY	
Month	Evnenses	Revenues	
nonen	пуреньев	Revenues	
FEB95	3,000	4,000	
MAR95	6,000	5,000	
 State	9.000	9.000	
Region	10,500	10,000	
	N = 2		
Sales	Region=Southern	State=FL	
Month	Expenses	Revenues	
FEB95	8,500	11,000	
MAR95	9,800	13,500	
State	18,300	24,500	
	N = 2		
Sales	Pogion-Southorn	State-CA	
Jales	Kegron-souchern	State-GA	
Month	Expenses	Revenues	
JAN95	2,000	8,000	
FEB95	1,200	6,000	
 State	3,200	14,000	
Region	21,500	38,500	
	32,000	48,500	
	N = 2		
	Total N = 7		

## Example 6: Limiting the Number of Sums in a Report

#### Features:

BY statement SUM statement SUMBY statement Other features: SORT procedure LABEL statement Data set: EXPREV on page 798

This example

- □ creates a separate section of the report for each combination of state and region
- □ sums expenses and revenues only for each region and for all regions, not for individual states.

### Program

```
options nodate pageno=1 linesize=70 pagesize=60;
```

PROC SORT sorts the observations by Region and State.

```
proc sort data=exprev;
    by region state;
run;
```

NOOBS suppresses the printing of observation numbers at the beginning of the rows.

```
proc print data=exprev noobs;
```

The SUM and BY statements work together to sum the values of Revenues and Expenses for each BY group as well as for the whole report. The SUMBY statement limits the subtotals to one for each region.

```
by region state;
sum revenues expenses;
sumby region;
```

The LABEL statement associates a label with the variable Region for the duration of the PROC PRINT step. This label is used in the BY lines.

```
label region='Sales Region';
```

The FORMAT statement assigns the COMMA10. format to Expenses and Revenues for this report. The TITLE statement specifies the title.

```
format revenues expenses commal0.;
  title 'Revenue and Expense Figures for Each Region';
run;
```

## Output

The report uses default column headers (variable names) because neither the SPLIT= nor the LABEL option is used. Nevertheless, the BY line at the top of each section of the report shows the BY variables' labels and their values. The name of a BY variable identifies the subtotals in the report.

Revenue and	Expense Figures :	for Each Region	1
Sales	Region=Northern	State=MA	
Month	Expenses	Revenues	
MAR95	1,500	1,000	
Sales	Region=Northern	State=NY	
Month	Expenses	Revenues	
FEB95 MAR95	3,000 6,000	4,000 5,000	
Region	10,500	10,000	
Sales	Region=Southern	State=FL	
Month	Expenses	Revenues	
FEB95	8,500	11.000	
MAR95	9,800	13,500	
Sales	Region=Southern	State=GA	
Month	Expenses	Revenues	
JAN95	2,000	8,000	
FEB95	1,200	6,000	
 Region	21,500	38,500	
	32,000	48,500	

## **Example 7: Controlling the Layout of a Report with Many Variables**

Procedure features: PROC PRINT statement options: ROWS= ID statement Other features: SAS data set options: OBS=

This example shows two ways of printing a data set with a large number of variables: one is the default, and the other uses ROWS=. For detailed explanations of the layouts of these two reports, see the ROWS= option on page 789 and see "Page Layout" on page 795.

These reports use a pagesize of 24 and a linesize of 64 to help illustrate the different layouts.

*Note:* When the two reports are written as HTML output, they do not differ.  $\triangle$ 

### Program

```
options nodate pageno=1 linesize=64 pagesize=24 ;
```

The data set EMPDATA contains personal and job-related information about a company's employees. A DATA step on page 1501 creates this data set.

```
data empdata;
   input IdNumber $ 1-4 LastName $ 9-19 FirstName $ 20-29
        City $ 30-42 State $ 43-44 /
        Gender $ 1 JobCode $ 9-11 Salary 20-29 @30 Birth date7.
         @43 Hired date7. HomePhone $ 54-65;
   format birth hired date7.;
   datalines;
1919
       Adams
                  Gerald
                            Stamford
                                         СТ
       TA2
                  34376
                            15SEP48
                                         07JUN75
                                                    203/781-1255
М
1653
       Alexander Susan
                            Bridgeport
                                         СТ
                  35108
F
       ME2
                            180CT52
                                         12AUG78
                                                    203/675-7715
. . . more lines of data . . .
1407
       Grant
                  Daniel
                            Mt. Vernon
                                         NY
М
       PT1
                  68096
                            26MAR57
                                         21MAR78
                                                    914/468-1616
1114
       Green
                  Janice
                            New York
                                         NY
       TA2
                            21SEP57
                                                    212/588-1092
F
                  32928
                                         30.TUN75
;
```

The OBS= data set option uses only the first 12 observations to create the report. (This is just to conserve space here.) The ID statement identifies observations with the formatted value of IdNumber rather than with the observation number. This report is in Output 28.3 on page 811.

```
proc print data=empdata(obs=12);
    id idnumber;
    title 'Personnel Data';
run;
```

ROWS=PAGE prints only one row of variables for each observation on a page. This report is in Output 28.4 on page 812.

```
proc print data=empdata(obs=12) rows=page;
    id idnumber;
    title 'Personnel Data';
run;
```

## Output

Output 28.3 Default Layout for a Report with Many Variables

In the traditional procedure output, each page of this report contains values for all variables in each observation. In the HTML output, this report is identical to the report that uses ROWS=PAGE.

Note that PROC PRINT automatically splits the variable names that are used as column headers at a change in capitalization if the entire name does not fit in the column. Compare, for example, the column headers for LastName (which fits in the column) and FirstName (which does not fit in the column).

			Personn	el Da	ta		1
Id Number	LastNa	F ame N	'irst Mame		City	State	Gender
1919 1653 1400 1350 1401 1499	Adams Alexar Apple Arthur Avery Barefo	der S nder S I E J Dot J	erald usan roy arbara erry oseph	St Br Ne Pa Pr	amford idgeport w York w York terson inceton	CT CT NY NY NJ NJ	M F M F M
1101 Id Number	Baucon Job Code	n W Salary	alter Bi	Ne rth	w York Hired	NY HomeF	M Phone
1919 1653 1400 1350 1401 1499 1101	TA2 ME2 ME1 FA3 TA3 ME3 SCP	34376 35108 29769 32886 38822 43025 18723	15SE 18OC 08NO 03SE 16DE 29AP 09JU	P48 T52 V55 P53 C38 R42 N50	07JUN75 12AUG78 190CT78 01AUG78 20NOV73 10JUN68 040CT78	203/78 203/67 212/58 718/38 201/73 201/81 212/58	31-1255 75-7715 36-0808 33-1549 32-8787 .2-5665 36-8060

			Personr	nel Dat	a		2
Id			First				
Number	LastNam	е	Name		City	State	Gender
1333	Blair		Justin	Sta	mford	СТ	М
1402	Blalock		Ralph	New	York	NY	м
1479	Bostic		Marie	New	York	NY	F
1403	Bowden		Earl	Bri	dgeport	СТ	м
1739	Воусе		Jonathan	New	York	NY	М
Id	Job						
Number	Code	Salar	у Ві	rth	Hired	HomeP	hone
1333	PT2	8860	6 02AE	PR49	13FEB69	203/78	1-1777
1402	TA2	3261	.5 20JA	N51	05DEC78	718/38	4-2849
1479	TA3	3878	5 25DE	C56	080CT77	718/38	4-8816
1403	ME1	2807	2 31JA	N57	24DEC79	203/67	5-3434
1739	PT1	6651	.7 28DE	C52	30JAN79	212/58	7-1247

Output 28.4 Layout Produced by the ROWS=PAGE Option

Each page of this report contains values for only some of the variables in each observation. However, each page contains values for more observations than the default report does.

		Personne	l Data		1
Id	LastNama	First	City	State	Condor
Number	LastName	Name	CIty	State	Gender
1919	Adams	Gerald	Stamford	СТ	М
1653	Alexander	Susan	Bridgeport	СТ	F
1400	Apple	Тгоу	New York	NY	М
1350	Arthur	Barbara	New York	NY	F
1401	Avery	Jerry	Paterson	NJ	М
1499	Barefoot	Joseph	Princeton	NJ	М
1101	Baucom	Walter	New York	NY	М
1333	Blair	Justin	Stamford	СТ	М
1402	Blalock	Ralph	New York	NY	М
1479	Bostic	Marie	New York	NY	F
1403	Bowden	Earl	Bridgeport	СТ	М
1739	Воусе	Jonathan	New York	NY	М

		P	ersonnel Da	ta	2
Id Number	Job Code	Salary	Birth	Hired	HomePhone
1919	TA2	34376	15SEP48	07JUN75	203/781-1255
1653	ME2	35108	180CT52	12AUG78	203/675-7715
1400	ME1	29769	08NOV55	190CT78	212/586-0808
1350	FA3	32886	03SEP53	01AUG78	718/383-1549
1401	TA3	38822	16DEC38	20NOV73	201/732-8787
1499	ME 3	43025	29APR42	10JUN68	201/812-5665
1101	SCP	18723	09JUN50	040CT78	212/586-8060
1333	PT2	88606	02APR49	13FEB69	203/781-1777
1402	TA2	32615	20JAN51	05DEC78	718/384-2849
1479	TA3	38785	25DEC56	080CT77	718/384-8816
1403	ME1	28072	31JAN57	24DEC79	203/675-3434
1739	PT1	66517	28DEC52	30JAN79	212/587-1247

## **Example 8: Creating a Customized Layout with BY Groups and ID Variables**

Procedure features: BY statement ID statement SUM statement VAR statement Other features: SORT procedure Data set: EMPDATA on page 810

This customized report

- $\hfill\square$  selects variables to include in the report and controls their order
- $\hfill\square$  selects observations to include in the report
- $\hfill\square$  groups the selected observations by JobCode
- $\hfill\square$  sums the salaries for each job code and for all job codes
- $\hfill\square$  displays numeric data with commas and dollar signs.

### Program

PROC SORT creates a temporary data set in which the observations are sorted by JobCode and Gender.

```
options nodate pageno=1 linesize=64 pagesize=60;
proc sort data=empdata out=tempemp;
    by jobcode gender;
run;
```

SPLIT= identifies the asterisk as the character that starts a new line in column headers.

```
proc print data=tempemp split='*';
```

The VAR statement and the ID statement together select the variables to include in the report. The ID statement and the BY statement produce the special format.

id jobcode; by jobcode; var gender salary;

The SUM statement totals the values of Salary for each BY group and for the whole report.

sum salary;

The LABEL statement associates a label with each variable for the duration of the PROC PRINT step. When you use SPLIT= in the PROC PRINT statement, the procedure uses labels for column headings.

```
label jobcode='Job Code*======'
gender='Gender*====='
salary='Annual Salary*=========';
```

The FORMAT statement assigns a format to Salary for this report. The WHERE statement selects for the report only the observations for job codes that contain the letters 'FA' or 'ME'. The TITLE statements specify two titles.

```
format salary dollar11.2;
where jobcode contains 'FA' or jobcode contains 'ME';
title 'Expenses Incurred for';
title2 'Salaries for Flight Attendants and Mechanics';
run;
```

### Output

The ID and BY statements work together to produce this layout. The ID variable is listed only once for each BY group. The BY lines are suppressed. Instead, the value of the ID variable, JobCode, identifies each BY group.

Ex Salaries for	xpenses Incu Flight Atte	urred for endants and Mechanics	
Job Code =======	Gender =====	Annual Salary	
FA1	F F M	\$23,177.00 \$22,454.00 \$22,268.00	
 FA1		\$67,899.00	
FA2	F F M	\$28,888.00 \$27,787.00 \$28,572.00	
 FA2		\$85,247.00	
FA3	F F M	\$32,886.00 \$33,419.00 \$32,217.00	
 FA3		\$98,522.00	
ME1	M M M	\$29,769.00 \$28,072.00 \$28,619.00	
 ME1		\$86,460.00	
ME2	F F M	\$35,108.00 \$34,929.00 \$35,345.00	
	M M M	\$36,925.00 \$35,090.00 \$35,185.00	
ME2		\$212,582.00	
ME3	М	\$43,025.00 ===================================	

## **Example 9: Printing All the Data Sets in a SAS Library**

Features: Macro facility DATASETS procedure PRINT procedure Data set: EXPREV on page 798 and LIST on page 492

This example prints all the data sets in a SAS library. You can use the same programming logic with any procedure. Just replace the PROC PRINT step near the

end of the example with whatever procedure step you want to execute. The example uses the macro language. For details about the macro language, see SAS Guide to Macro Processing, Version 6, Second Edition.

### Program

```
libname printlib 'SAS-data-library'
options nodate pageno=1 linesize=80 pagesize=60;
```

PROC DATASETS copies two data sets out of the WORK library into the PRINTLIB library to limit the number of data sets available to the example.

```
proc datasets library=work memtype=data nolist;
    copy out=printlib;
        select list exprev;
run;
```

The %MACRO statement creates the macro PRINTALL. When you call the macro, you can pass it one or two parameters. The first parameter is the name of the library whose data set you want to print. The second parameter is a library used by the macro. If you do not specify this parameter, the WORK library is the default.

```
%macro printall(libname,worklib=work);
```

The %LOCAL statement creates two local macro variables, NUM and I, to use in a loop.

```
%local num i;
```

This PROC DATASETS step reads the library LIBNAME that you specify as a parameter when you invoke the macro. The CONTENTS statement produces an output data set called TEMP1 in WORKLIB. This data set contains an observation for each variable in each data set in the library LIBNAME. By default, each observation includes the name of the data set that the variable is in as well as other information about the variable. However, the KEEP= data set option writes only the name of the data set to TEMP1.

```
proc datasets library=&libname memtype=data nodetails;
    contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
run;
```

This DATA step increments the value of N each time that it reads the last occurrence of a data set name (when IF LAST.MEMNAME is true). The CALL SYMPUT statement uses the current value of N to create a macro variable for each unique value of MEMNAME in the data set TEMP1. The TRIM function removes extra blanks in the TITLE statement in the PROC PRINT step that follows.

```
data _null_;
   set &worklib..temp1 end=final;
```

```
by memname notsorted;
if last.memname;
n+1;
call symput('ds'||left(put(n,8.)),trim(memname));
```

When it reads the last observation in the data set (when FINAL is true), the DATA step assigns the value of N to the macro variable NUM. At this point in the program, the value of N is the number of observations in the data set.

```
if final then call symput('num',put(n,8.));
```

The RUN statement is crucial. It forces the DATA step to run, thus creating the macro variables used in the CALL SYMPUT statements before the %DO loop, which uses them, executes.

run;

The %DO loop issues a PROC PRINT step for each data set. The %MEND statement ends the macro.

```
%do i=1 %to #
    proc print data=&libname..&&ds&i noobs;
        title "Data Set &libname..&&ds&i";
        run;
        %end;
%mend printall;
```

This invocation of the PRINTALL macro prints all the data sets in the library PRINTLIB.

```
options nodate pageno=1 linesize=70 pagesize=60;
%printall(printlib)
```

### Output

	Data S	et printl	ib.EXPREV	
Region	State	Month	Expenses	Revenues
Northern	MA	MAR95	1500	1000
Northern	NY	FEB95	3000	4000
Northern	NY	MAR95	6000	5000
Southern	FL	FEB95	8500	11000
Southern	FL	MAR95	9800	13500
Southern	GA	JAN95	2000	8000
Southern	GA	FEB95	1200	6000

	Data Set printlib.LIST		2	
Name	Street	City	State	Zip
Gabrielli, Theresa	24 Ridgetop Rd.	Westboro	MA	01581
Clayton, Aria	314 Bridge St.	Hanover	NH	03755
Dix, Martin L.	4 Shepherd St.	Norwich	VT	05055
Slater, Emily C.	2009 Cherry St.	York	PA	17407
Ericson, Jane	211 Clancey Court	Chapel Hill	NC	27514
An, Ing	95 Willow Dr.	Charlotte	NC	28211
Jacobson, Becky	7 Lincoln St.	Tallahassee	FL	32312
Misiewicz, Jeremy	43-C Lakeview Apts.	Madison	WI	53704
Ahmadi, Hafez	5203 Marston Way	Boulder	CO	80302
Archuleta, Ruby	Box 108	Milagro	NM	87429

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS[®] Procedures Guide, Version 8, Cary, NC: SAS Institute Inc., 1999. 1729 pp.

#### SAS[®] Procedures Guide, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-482-9

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 Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999

 $SAS^{\circledast}$  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.

 $\rm IBM^{\circledast}$  and  $\rm DB2^{\circledast}$  are registered trademarks or trademarks of International Business Machines Corporation. ORACLE  $^{\circledast}$  is a registered trademark of Oracle Corporation.  $^{\circledast}$  indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.