Understanding the SAS® DATA Step and the Program Data Vector

SYSTEMS SEMINAR CONSULTANTS, INC.

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

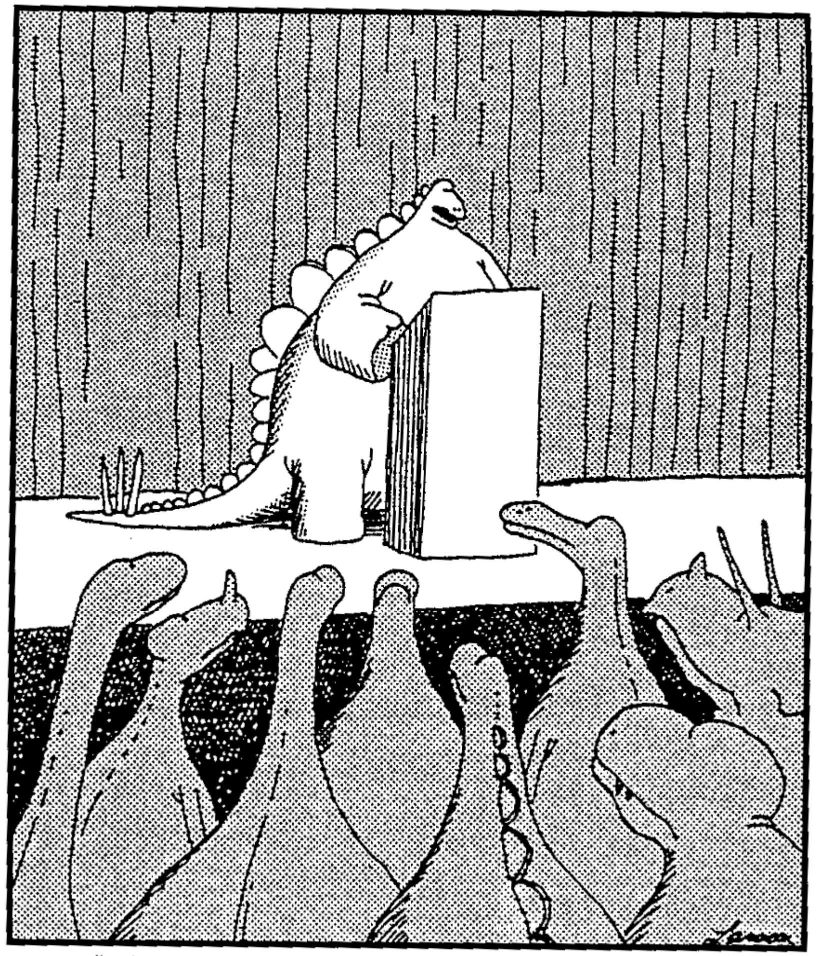
Understanding the SAS® DATA Step and the PDV

This presentation 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*™*.*

Global Warming Part 1



"The picture's pretty bleak, gentlemen. … The world's climates are changing, the mammals are taking over, and we all have a brain about the size of a walnut."

Abstract

The SAS system is made up of SAS PROCs and the DATA Step

The DATA Step has:

- an excellent, full fledged programming language
- statements to read and write almost any type of data value
- Conversion and calculation of new data
- Looping
- Interfaces
- Arrays
- Much, much more.

In many ways, the design of the DATA step along with its powerful statements, is what makes the SAS language so popular.

Abstract (continued)

This paper will address:

- How the DATA step fits with the rest of the SAS System
- DATA step assumptions and defaults
- internal structures such as buffers, and the Program Data Vector
- compiler statements
- executable statements

Introduction

The SAS system's origins are in the 1960's and 1970's with:

- James Goodnight
- John Sall
- A. J. Barr
- others

Concepts in the design include:

- "self defining files"
- a system of default assumptions
- procedures for commonly used routines
- data handling step that would evolve into the SAS DATA step

Introduction

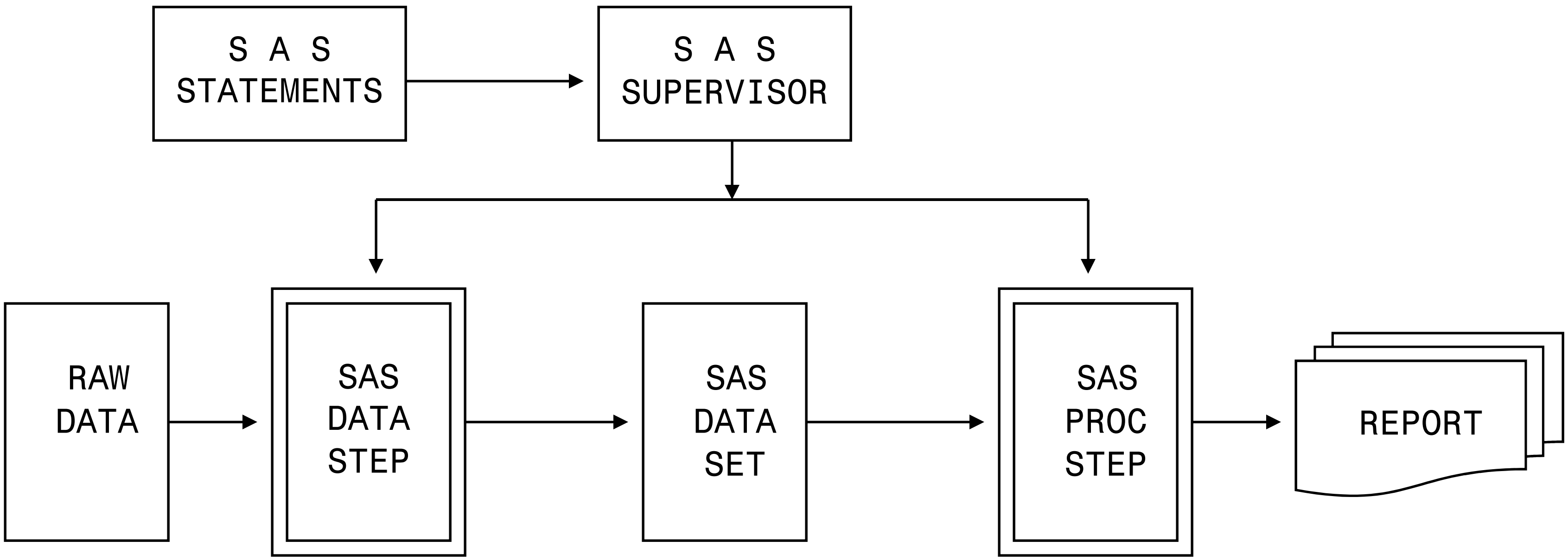
The DATA step in my opinion:

- extremely simple
- elegant design
- continues today
- 30 years of enhancements.

Structure of SAS

SAS consists of:

1. a data handling language (DATA step)
2. a library of pre-written procedures (PROC step)



Purpose of the DATA Step

"Get the data in shape" for later PROCs and DATA steps.

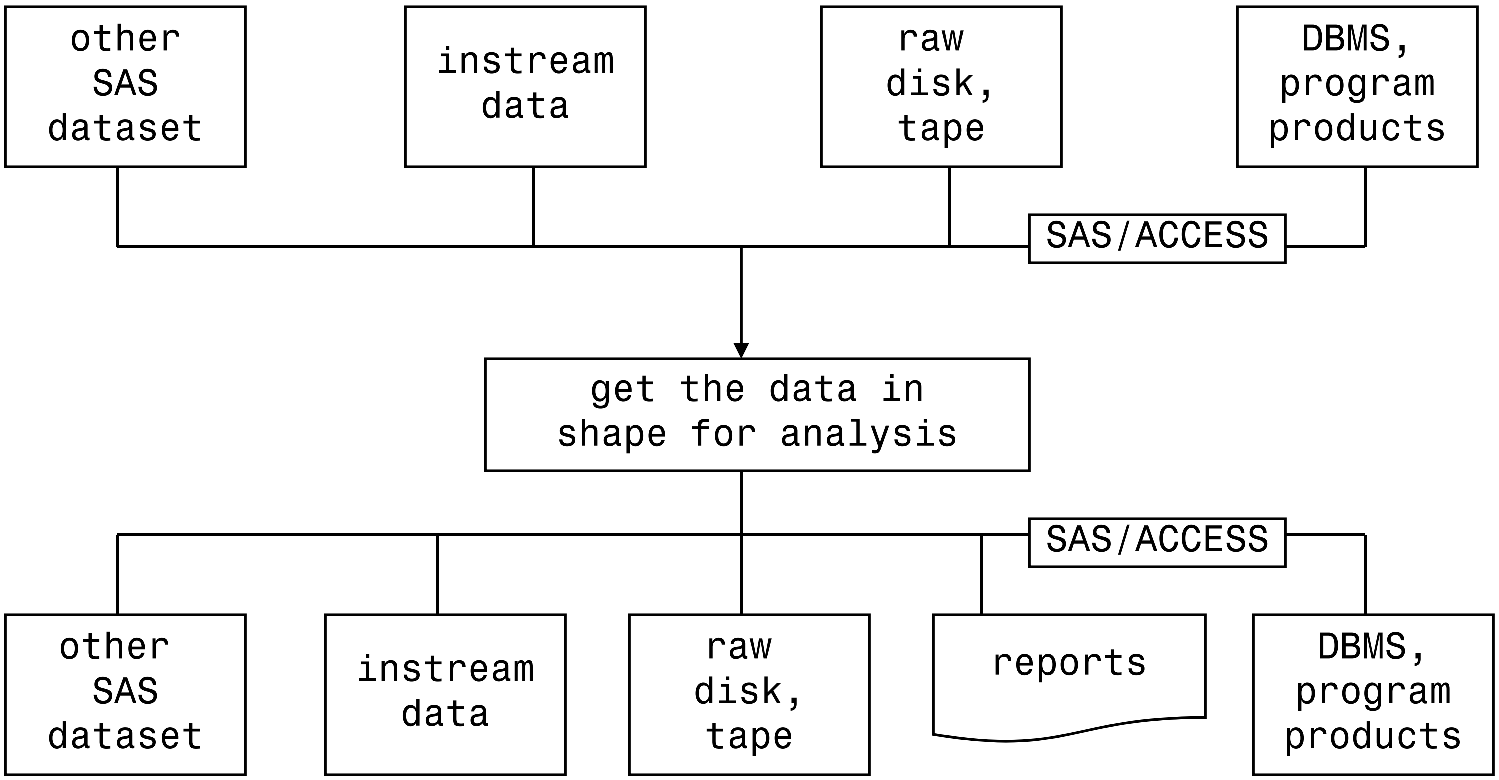
- SAS PROCs can only read SAS datasets
- We might have some other type of file to process
- SAS dataset has built in descriptor that keeps track of names and attributes
- later steps don't have to remember as many details

If we don't have well defined data

- DATA step gives us the power to read and write virtually any kind of file
- We can do calculations and computations on a single row of data
- It has a very powerful data handling language

SAS DATA Step Overview

DATA steps can read and write most types of data stored on your computer.



Notes:

- DATA step output is usually a SAS dataset but can be other files.
- Access to non-SAS database management systems requires a SAS/ACCESS product.

Default Assumptions

Many assumptions are made to save time and effort.

As computer scientist I study and use many programming languages

- Early on I was intrigued by the cleverness and common sense of SAS
- Many tedious programming tasks were eliminated through defaults
- System still provided a means to override those defaults when necessary
- Our job as a DATA step programmer then is different from other languages

In many ways our tasks are:

- understanding the defaults
- knowing how to work with them
- override them as necessary.

Default Assumptions (continued)

Examples Of SAS Defaults:

- Handling compile and execution naming and storage details
- A dataset descriptor that makes SAS datasets "self defining"
- Generating data set names if omitted
- When reading a data set, assume most recently created dataset if not specified
- Processing all the rows and columns in a file
- Automatically opening and closing of files
- Automatically controlling data initialization
- DATA step looping, data set output, and end of file checking

Default Assumptions (continued)

Examples Of SAS Defaults:

- Automatically defining storage areas for each variable referenced without need to predefine them
- A default length of 8 was assumed for all variables
- A assumption that a variable is numeric if not specified
- LIST input assumed that data values would be separated by blanks rather than specifying exact columns
- SUBSETTING IF statements which imply to continue processing if a condition is true, else delete the observation. (Ex. If rate > 10;)
- When no comparison is made in an IF statement, assume to be checking for 1 (true) (Ex. If eof then put ‘At end’;)
- Abreviated sum statements. (Ex. Salestot+sales)

Compiling a DATA Step

As most languages, the DATA step is first compiled then executed.

- Languages can be compiled, interpreted, SAS is a hybrid language
- Some features from other languages
- Some unique features
- Sometimes difficult to separate compile versus execution events with SAS
- DATA step compiler examines SAS statements for syntax data structures
- generates an executable program
- Unique to SAS compiler checking for the existence of resources
- Assumptions that it “inserts” into the source code.

Data Structures

“Getting the data in shape” needs data structures to hold data as processed.

- All computer languages need to address this
- Each language may name the structures differently
- There is a lot of similarity in the way most languages store data.

A Typical SAS Job

input buffer

```
1234567890123456789012345678901234
BETH       H  12  4822.12   982.10
CHRIS      H   2   233.11    94.12
JOHN       H   7   678.43   150.11
```

```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	NUM	NUM
Length:	10	1	8	8	8	8	8
Format:							
Informat:							
Label:							
Flags:						D	D
Value							

DATASET DESCRIPTOR PORTION (DISK)

Name:	NAME	DIVISION	YEARS	SALES	EXPENSE
Type:	CHAR	CHAR	NUM	NUM	NUM
Length:	10	1	8	8	8
Format:					
Informat:					
Label:					

DATASET DATA PORTION

BETH	H	12	4822.12	982.10
CHRIS	H	2	233.11	94.12
JOHN	H	7	678.43	150.11

Raw File Buffers

DATA steps reading/writing “raw” or non-SAS data need memory buffers.

- Needed to temporarily hold at least one input record at a time.
- Also times when multiple lines of input can be held in buffers
- Allows the program to logically read later rows before earlier ones.
- Buffer contains the complete input and output record, regardless of whether the INPUT statement reads all of the columns.
- SAS datasets and RDMS (which usually appear as SAS datasets) do not use raw buffers as the files are already in “shape”.

Raw File Buffers

```
              1234567890123456789012345678901234
              BETH          H  12  4822.12    982.10
input buffer  CHRIS         H   2   233.11     94.12
              JOHN          H   7   678.43    150.11
```

```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	NUM	NUM
Length:	10	1	8	8	8	8	8
Format:							
Informat:							
Label:							
Flags:						D	D
Value							

DATASET DESCRIPTOR PORTION (DISK)

	NAME	DIVISION	YEARS	SALES	EXPENSE
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE
Type:	CHAR	CHAR	NUM	NUM	NUM
Length:	10	1	8	8	8
Format:					
Informat:					
Label:					

DATASET DATA PORTION

BETH	H	12	4822.12	982.10
CHRIS	H	2	233.11	94.12
JOHN	H	7	678.43	150.11

The LOGICAL PROGRAM DATA VECTOR (PDV)

A second memory area for data manipulation, conversion, refinement.

Areas are needed for:

- Inputting and input formatting (informatting) desired variables
- Revising existing values
- Computing new variables
- System indicators and flags
- Called Logical Program Data Vector (PDV).
- Many languages have a similar working area. (COBOL Working Storage).
- “Retained” and “Non-retained” variables are stored in separate *physical* program data vectors, together make *Logical Program Data Vector*
- All variables referenced be automatically defined in the PDV by compiler using characteristics from the first reference of a variable.

Example: agemo=age/12;

The LOGICAL PROGRAM DATA VECTOR (PDV)

input buffer

```
1234567890123456789012345678901234
BETH         H  12  4822.12   982.10
CHRIS        H   2   233.11    94.12
JOHN         H   7   678.43   150.11
```

```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	NUM	NUM
Length:	10	1	8	8	8	8	8
Format:							
Informat:							
Label:							
Flags:						D	D
Value							

DATASET DESCRIPTOR PORTION (DISK)

	NAME	DIVISION	YEARS	SALES	EXPENSE
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE
Type:	CHAR	CHAR	NUM	NUM	NUM
Length:	10	1	8	8	8
Format:					
Informat:					
Label:					

DATASET DATA PORTION

BETH	H	12	4822.12	982.10
CHRIS	H	2	233.11	94.12
JOHN	H	7	678.43	150.11

The Final SAS Dataset

A “self-defining” dataset.

- The dataset descriptor contains attributes for all kept variables plus data set labeling information.
- The Dataset Data portion contains data values for all output rows and kept columns.
- Pseudo-variables are not kept.

The Final SAS Dataset

```
              1234567890123456789012345678901234
input buffer  BETH        H  12  4822.12   982.10
              CHRIS       H   2   233.11    94.12
              JOHN        H   7   678.43   150.11
```

```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	NUM	NUM
Length:	10	1	8	8	8	8	8
Format:							
Informat:							
Label:							
Flags:						D	D
Value							

DATASET DESCRIPTOR PORTION (DISK)

	NAME	DIVISION	YEARS	SALES	EXPENSE
Name:	NAME	DIVISION	YEARS	SALES	EXPENSE
Type:	CHAR	CHAR	NUM	NUM	NUM
Length:	10	1	8	8	8
Format:					
Informat:					
Label:					

DATASET DATA PORTION

BETH	H	12	4822.12	982.10
CHRIS	H	2	233.11	94.12
JOHN	H	7	678.43	150.11

Variable Attributes

The compiler assigns each defined variable several characteristics.

- Relative variable number
- Position in the dataset
- Name
- Data type
- Length in bytes
- Informat
- Format
- Variable label
- Flags to indicate dropping, retaining, handling of missing values for each variable

Data Types and Conversion

All SAS values are one of two data types: numeric or character.

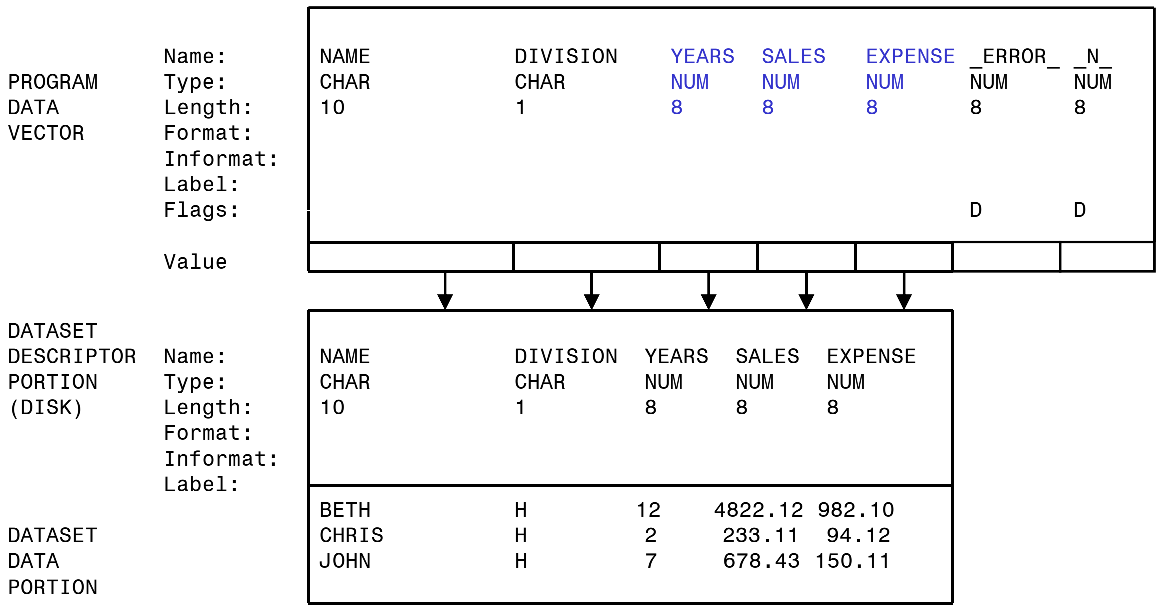
- Hundreds of different data types can be read or written via the PDV, only two stored
- In PDV, SAS Dataset every character value is stored as a native EBCDIC or ASCII value with length between 1 and at least 32767
- Numerics stored as double precision floating point values with length between 3 and 8 bytes.
- Storing only two data types greatly simplifies things for SAS datasets
- moves the complication of converting different data types (packed, binary, etc.) to the INPUT and PUT statement along with appropriate INFORMATS and FORMATS.
- Floating point for numbers with a length of 8 allows for storage of very large numbers (or small) without overflow
- Floating point does have minor mathematical issues of its own.

Data Types and Conversion

input buffer

```
1234567890123456789012345678901234
BETH       H  12  4822.12   982.10
CHRIS      H   2   233.11    94.12
JOHN       H   7   678.43   150.11
```

```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```



Pseudo Variables

Special variables that the compiler creates that are not added to output file.

Partial list:

_N_ contains the number of times the DATA step has looped.

ERROR is set to 0 if there were no input errors, otherwise 1.

FIRST.*variable* showing beginning of control break

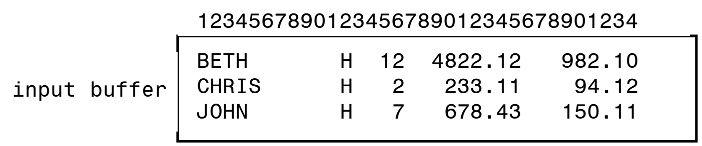
LAST.*variable* showing end of control break

INFILE contains entire input buffer

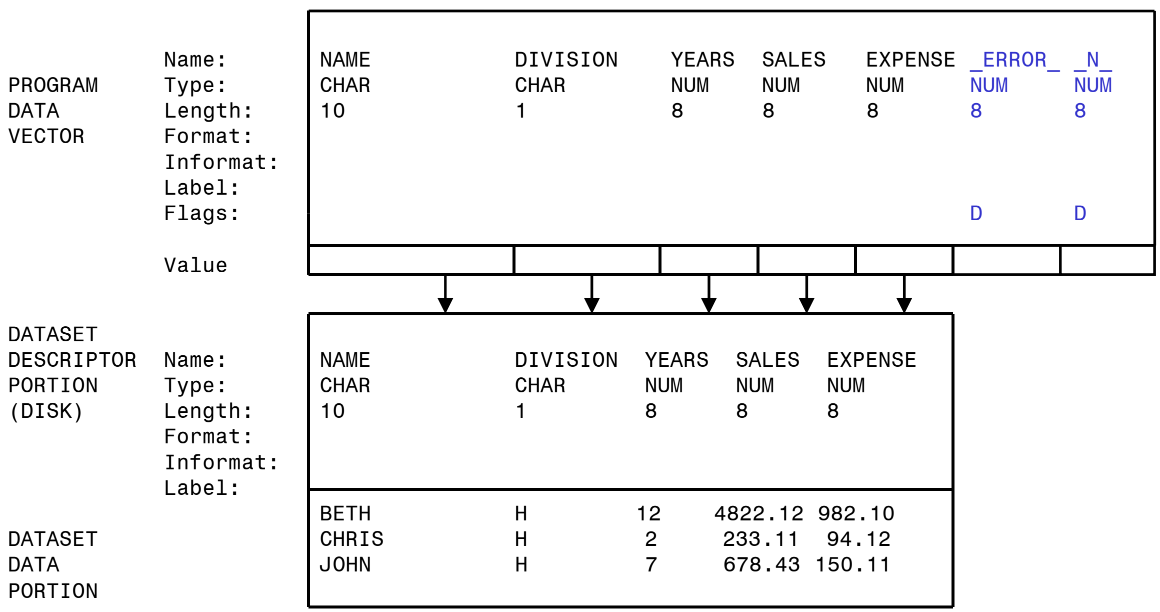
Others can be requested by the programmer to:

- Detect end of file
- Access control blocks
- Access and alter system information

Pseudo Variables




```
data softsale;
  infile rawin;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
run;
```



Output Datasets

Usually a SAS datafile, but raw files and reports can also be created.

- SAS data sets are “self-defining” via dataset descriptor
- Much simpler operation for the programmer
- SAS automatically builds the structures needed
- Outputs the record at DATA step return.
- In addition, all variables except dropped and pseudo variables are kept
- Descriptor info can be printed with PROC CONTENTS, dictionary tables

Notes:

- If a raw file or report is produced, buffers opposite those of INFILE are used.
- Raw files can pass info to other programs.

Questions About The Data Step

If traditional programming experience is applied this DATA step, questions might be:

Where are the opens and closes?

Where do we write out records?

What is looping?

When does the program stop?

Assumptions Made In The Data Step

The default actions of the DATA step easily accommodates *most* programs.

- All rows are read from files starting with the first record until last record.
- Input values from a previous row are cleared before reading next row.
- All variables referenced will be included on the output file.
- All records will be included on the resulting output file.
- All files should be opened at the beginning and closed at the end of step.
- Programs should not continue to loop if no data is read in previous pass.

Assumptions Made In The Data Step

The SAS compiler makes the following assumption and inserts code to do:

- Immediately upon entry check for infinite looping
- All values from non-SAS files are cleared before executing any statements.
- If any reading statement would read a record after end of file, step stops.
- If the program reaches the last statement in the step, or if a RETURN statement is executed, the current PDV contents (all columns for each row) is output to the SAS data set being built.
- A branch is executed to go to the top and enter the DATA step for another pass.

Assumptions Made In The Data Step

Another view it is that the compiler inserts the blue italic code.

```
data softsale;
Check for looping, initialize PDV
infile rawin;
if at EOF then stop
Input   Name       $1-10  Division  $12
        Years       15-16 Sales      19-25
        Expense     28-34 State     $36-37;
output to SAS Dataset
goto top of DATA step
run;
```

Notes:

- These save effort and make DATA steps virtually infinite-loop proof.

Executing the DATA Step

```
data softsale;
   init buffer, PDV
   infile rawin;
   if at EOF then stop
   input Name $1-10 Division $12 Years 15-16
      Sales 19-25 Expense 28-34 State $36-37;
   output to SAS Dataset
   goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS     H  2  233.11   94.12 WI
MARK      H  5  298.12   52.65 WI
SARAH     S  6  301.21   65.17 MN
```

```
                1         2         3         4   8
              1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                          Fileref rawin
                      1         2         3
             1234567890123456789012345678901234567
            +-------------------------------------+
            |CHRIS     H  2  233.11   94.12 WI    |
            |MARK      H  5  298.12   52.65 WI    |
            |SARAH     S  6  301.21   65.17 MN    |
            +-------------------------------------+
```

```
data softsale;
   init buffer, PDV
   infile rawin;
   if at EOF then stop
   input Name $1-10 Division $12 Years 15-16
      Sales 19-25 Expense 28-34 State $36-37;
   output to SAS Dataset
   goto top of Data Step
run;
```

```
                       1         2         3         4   8
              1234567890123456789012345678901234567890...0
Input buffer  [                                          ]
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                            Fileref rawin
                                           1         2         3
data softsale;                    1234567890123456789012345678901234567
  init buffer, PDV                CHRIS      H  2   233.11     94.12 WI
  infile rawin;                   MARK       H  5   298.12     52.65 WI
  if at EOF then stop             SARAH      S  6   301.21     65.17 MN
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                        1         2         3         4    8
                   1234567890123456789012345678901234567890...0
Input buffer      [                                            ]
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)						

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
                      1         2         3         4   8
                 1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)						

Executing the DATA Step

Fileref rawin

```
                    1         2         3
           1234567890123456789012345678901234567
           CHRIS      H  2   233.11     94.12 WI
           MARK       H  5   298.12     52.65 WI
           SARAH      S  6   301.21     65.17 MN
```

```
data softsale;
   init buffer, PDV
   infile rawin;
   if at EOF then stop
   input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
   output to SAS Dataset
   goto top of Data Step
run;
```

```
                       1         2         3         4   8
              1234567890123456789012345678901234567890...0
Input buffer  [                                            ]
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                   Fileref rawin
                                  1         2         3
data softsale;           1234567890123456789012345678901234567
  init buffer, PDV  -->  CHRIS     H  2   233.11    94.12 WI
  infile rawin;          MARK      H  5   298.12    52.65 WI
  if at EOF then stop    SARAH     S  6   301.21    65.17 MN
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                    1         2         3         4   8
                1234567890123456789012345678901234567890...0
Input buffer    CHRIS     H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
               1         2         3         4   8
Input buffer   1234567890123456789012345678901234567890...0
               CHRIS      H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                           Fileref rawin
                                         1         2         3
                                1234567890123456789012345678901234567
data softsale;            -->  CHRIS     H   2   233.11    94.12 WI
  init buffer, PDV              MARK      H   5   298.12    52.65 WI
  infile rawin;                 SARAH     S   6   301.21    65.17 MN
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                        1         2         3         4   8
               1234567890123456789012345678901234567890...0
Input buffer   CHRIS     H   2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                    Fileref rawin
                                       1         2         3
                              1234567890123456789012345678901234567
data softsale;          --->  CHRIS      H  2  233.11    94.12 WI
  init buffer, PDV            MARK       H  5  298.12    52.65 WI
  infile rawin;               SARAH      S  6  301.21    65.17 MN
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                 1         2         3         4   8
                1234567890123456789012345678901234567890...0
Input buffer    CHRIS      H  2  233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	**2**	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                  Fileref rawin
                                  1         2         3
                         1234567890123456789012345678901234567
data softsale;      ---> CHRIS     H  2   233.11    94.12 WI
  init buffer, PDV       MARK      H  5   298.12    52.65 WI
  infile rawin;          SARAH     S  6   301.21    65.17 MN
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
                    1         2         3         4   8
               12345678901234567890123456789012345678901234567890...0
Input buffer   CHRIS     H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	**233.11**	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
                                    Fileref rawin
                                    1         2         3
data softsale;             1234567890123456789012345678901234567
  init buffer, PDV    -->  CHRIS      H  2   233.11    94.12 WI
  infile rawin;            MARK       H  5   298.12    52.65 WI
  if at EOF then stop      SARAH      S  6   301.21    65.17 MN
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                   1         2         3         4   8
                  1234567890123456789012345678901234567890...0
Input buffer      CHRIS      H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	**94.12**	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)						

Executing the DATA Step

```
                                   Fileref rawin
                                1         2         3
data softsale;          1234567890123456789012345678901234567
  init buffer, PDV  --> CHRIS     H  2   233.11    94.12 WI
  infile rawin;         MARK      H  5   298.12    52.65 WI
  if at EOF then stop   SARAH     S  6   301.21    65.17 MN
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                    1         2         3         4   8
                  123456789012345678901234567890123456789 0...0
Input buffer      CHRIS     H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)						

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                    Fileref rawin
          1         2         3
 1234567890123456789012345678901234567
 CHRIS     H  2  233.11    94.12 WI
 MARK      H  5  298.12    52.65 WI
 SARAH     S  6  301.21    65.17 MN
```

```
                        1         2         3         4   8
               1234567890123456789012345678901234567890...0
Input buffer   CHRIS     H  2  233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	**H**	**2**	**233.11**	**94.12**	**WI**

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	**CHRIS**	**H**	**2**	**233.11**	**94.12**	**WI**

Executing the DATA Step

```
                                          Fileref rawin
                                        1         2         3
data softsale;                 1234567890123456789012345678901234567
  init buffer, PDV            +-------------------------------------+
  infile rawin;        -->    |CHRIS      H  2  233.11    94.12 WI  |
  if at EOF then stop         |MARK       H  5  298.12    52.65 WI  |
                              |SARAH      S  6  301.21    65.17 MN  |
                              +-------------------------------------+
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                     1         2         3         4   8
                1234567890123456789012345678901234567890...0
Input buffer    CHRIS      H  2  233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
                                    Fileref rawin
                           1         2         3
                  1234567890123456789012345678901234567
                  CHRIS     H  2   233.11    94.12 WI
                  MARK      H  5   298.12    52.65 WI
                  SARAH     S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                         1         2         3         4   8
                1234567890123456789012345678901234567890...0
Input buffer    CHRIS     H  2   233.11    94.12 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS     H  2  233.11    94.12 WI
MARK      H  5  298.12    52.65 WI
SARAH     S  6  301.21    65.17 MN
```

Input buffer

```
         1         2         3         4    8
1234567890123456789012345678901234567890...0
```

Logical Program Data Vector	Name	Division	Years	Sales	Expense	State
			.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Input buffer

```
         1         2         3         4    8
1234567890123456789012345678901234567890...0
```


Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11     94.12 WI
MARK       H  5   298.12     52.65 WI
SARAH      S  6   301.21     65.17 MN
```

Input buffer

```
         1         2         3         4   8
1234567890123456789012345678901234567890...0
```


Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS     H  2  233.11   94.12 WI
MARK      H  5  298.12   52.65 WI
SARAH     S  6  301.21   65.17 MN
```

Input buffer

```
         1         2         3         4   8
1234567890123456789012345678901234567890...0
MARK      H  5  298.12   52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS     H  2  233.11   94.12 WI
MARK      H  5  298.12   52.65 WI
SARAH     S  6  301.21   65.17 MN
```

Each row is handled the same way. Let's skip through MARK's input step.

Input buffer

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890...
MARK      H  5  298.12   52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

Input buffer

```
         1         2         3         4    8
1234567890123456789012345678901234567890...0
MARK       H  5   298.12    52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
MARK	**H**	**5**	**298.12**	**52.65**	**WI**

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

Input buffer

```
         1         2         3         4   8
1234567890123456789012345678901234567890...0
MARK       H  5   298.12    52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
MARK	**H**	**5**	**298.12**	**52.65**	**WI**

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	**H**	**5**	**298.12**	**52.65**	**WI**

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2  233.11    94.12 WI
MARK       H  5  298.12    52.65 WI
SARAH      S  6  301.21    65.17 MN
```

Input buffer

```
         1         2         3         4   8
1234567890123456789012345678901234567890...0
MARK       H  5  298.12    52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
MARK	H	5	298.12	52.65	WI

Dataset work.softsale

(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

Fileref rawin

```
           1         2         3
  1234567890123456789012345678901234567
  CHRIS     H  2  233.11    94.12 WI
  MARK      H  5  298.12    52.65 WI
  SARAH     S  6  301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                        1         2         3         4   8
              1234567890123456789012345678901234567890...0
Input buffer  MARK      H  5  298.12    52.65 WI
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
MARK	H	5	298.12	52.65	WI

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                   1         2         3         4   8
          1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

Input buffer

```
         1         2         3         4   8
1234567890123456789012345678901234567890...0
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                      1         2         3         4   8
             1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

```
                                          Fileref rawin
                                        1         2         3
data softsale;                 1234567890123456789012345678901234567
  init buffer, PDV             CHRIS     H  2   233.11    94.12 WI
  infile rawin;                MARK      H  5   298.12    52.65 WI
  if at EOF then stop    --->  SARAH     S  6   301.21    65.17 MN
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;                   1         2         3         4   8
               1234567890123456789012345678901234567890...0
Input buffer   SARAH     S  6   301.21    65.17 MN
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)						
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

```
                    Fileref rawin
               1         2         3
      1234567890123456789012345678901234567
      CHRIS     H  2   233.11    94.12 WI
      MARK      H  5   298.12    52.65 WI
  --> SARAH     S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Each row is handled the same way.
Let's skip through SARAH'S input step.

```
                        1         2         3         4
               1234567890123456789012345678901234567890...
Input buffer   SARAH     S  6   301.21    65.17 MN
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

```
                Fileref rawin
          1         2         3
 1234567890123456789012345678901234567
 CHRIS      H  2   233.11    94.12 WI
 MARK       H  5   298.12    52.65 WI
 SARAH      S  6   301.21    65.17 MN
```

```
                       1         2         3         4   8
               1234567890123456789012345678901234567890...0
Input buffer   SARAH      S  6   301.21    65.17 MN
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
SARAH	S	6	301.21	65.17	**MN**

Dataset work.softsale

(Descriptor) Name	Division	Years	Sales	Expense	State
(Data) CHRIS	H	2	233.11	94.12	WI
MARK	H	5	298.12	52.65	WI

Executing the DATA Step

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS     H  2   233.11    94.12 WI
MARK      H  5   298.12    52.65 WI
SARAH     S  6   301.21    65.17 MN
```

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Input buffer

```
         1         2         3         4    8
1234567890123456789012345678901234567890...0
SARAH     S  6   301.21    65.17 MN
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
SARAH	**S**	**6**	**301.21**	**65.17**	**MN**

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	**S**	**6**	**301.21**	**65.17**	**MN**

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

Input buffer

```
         1         2         3         4    8
1234567890123456789012345678901234567890...0
SARAH      S  6   301.21    65.17 MN
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
SARAH	S	6	301.21	65.17	MN

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	S	6	301.21	65.17	MN

Executing the DATA Step

```
                                   Fileref rawin
                                   1         2         3
                          1234567890123456789012345678901234567
                          CHRIS     H  2   233.11   94.12 WI
                          MARK      H  5   298.12   52.65 WI
                          SARAH     S  6   301.21   65.17 MN

data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
                 1         2         3         4    8
        1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	S	6	301.21	65.17	MN

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
          1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
                 1         2         3         4   8
                 1234567890123456789012345678901234567890...0
Input buffer     [                                            ]
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	S	6	301.21	65.17	MN

Executing the DATA Step

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
1234567890123456789012345678901234567
CHRIS      H  2   233.11    94.12 WI
MARK       H  5   298.12    52.65 WI
SARAH      S  6   301.21    65.17 MN
```

```
              1         2         3         4    8
Input buffer  1234567890123456789012345678901234567890...0
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	S	6	301.21	65.17	MN

Executing the DATA Step

We're done!
Go to the next step.

```
data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
     Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;
```

Fileref rawin

```
         1         2         3
234567890123456789012345678901234567
CHRIS     H  2  233.11    94.12 WI
MARK      H  5  298.12    52.65 WI
SARAH     S  6  301.21    65.17 MN
```

```
                    1         2         3         4   8
           1234567890123456789012345678901234567890...0
Input buffer
```

Logical Program Data Vector

Name	Division	Years	Sales	Expense	State
		.	.	.	

Dataset work.softsale

	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)	CHRIS	H	2	233.11	94.12	WI
	MARK	H	5	298.12	52.65	WI
	SARAH	S	6	301.21	65.17	MN

Overriding Data Step Assumptions

Not all programs fit scenario above; can we alter behavior of the program?

- The FIRSTOBS= and OBS= system options begin and end logically after the first record and before the last record respectively.
- The RETAIN compiler statement instructs the step to *not* initialize variables.
- The STOP statement (usually used with IF statement) stops step early.
- The DELETE, subsetting IF statements, exit the DATA step early; never reach OUTPUT.
- RETURN exits the DATA step early but does output to the SAS file.
- DROP, KEEP statements, dataset options exclude or include columns.
- GOTO and various DO groups alter the looping path for the program.

Retaining Data Values

Sometimes variables should not be cleared upon exit/reentry.

The RETAIN compiler statement:

- Specifies listed variables should not be initialized
- Can also set an initial value
- Iif first reference sets compiler attributes
- Is essentially a flag set by compiler to tell execution don’t clear
- Can be coded anywhere in DATA Step
- Variables read with SET, MERGE, or UPDATE, MODIFY are retained.

Retaining Data Values

Example: Read a file with CITY in the first row, RETAIN the value.

```
1234567890123456789012345678901234
 MADISON
BETH          H  12  4822.12   982.10
CHRIS         H   2   233.11    94.12
JOHN          H   7   678.43   150.11
```

```
data softsale;
  infile rawin missover;
  if _N_ = 1 then do;
     input city $2-8;
     delete;  end;
  input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
  retain city;
run;
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	CITY	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	CHAR	NUM	NUM
Length:	10	1	8	8	8	7	8	8
Format:								
Informat:								
Label:								
Flags:						R	D	D
Value						MADISON		

DATASET DESCRIPTOR PORTION (DISK)

Name:	NAME	DIVISION	YEARS	SALES	EXPENSE	CITY
Type:	CHAR	CHAR	NUM	NUM	NUM	CHAR
Length:	10	1	8	8	8	7
Format:						
Informat:						
Label:						

DATASET DATA PORTION

BETH	H	12	4822.12	982.10	MADISON
CHRIS	H	2	233.11	94.12	MADISON
JOHN	H	7	678.43	150.11	MADISON

Stopping Early

Normally jobs run until EOF, OBS= or STOP can stop step early.

Example: Stop after 50 records

```
 data softsale;
  infile rawin;
  if _N_ = > 50 then stop;
  input name $1-10 division $12 years 15-16 sales 19-25
        expense 27-34;
run;
```

Stopping Late

Sometimes a program should continue even if the last record was read.

Example: Calculate a percentage of total

```
DATA PCTDS;
 IF _N_ = 1 THEN
     DO UNTIL(EOF);
        SET CONCAT(KEEP=NAME
                        SALES)
              END=EOF;
        TOTSALES+SALES;
     END;
 SET CONCAT(KEEP=NAME SALES);

SALESPCT=(SALES*100)/TOTSALES;
RUN;
```

CONCAT DATASET

OBS	NAME	YEARS	SALES
1	BETH	12	4822.12
2	CHRIS	2	233.11
3	JOHN	7	678.43
4	MARK	5	298.12
5	ANDREW	24	1762.11
6	BENJAMIN	3	201.11
7	JANET	1	98.11

PDV

Name	Sales	TOTSALES	SALESPCT

Outputting And Deleting Observations

Default is to output the current row if DATA step reaches implied OUTPUT.

Statements to discard rows:

- DELETE (usually after an IF) leaves the DATA step without OUTPUT.
- False cases of Subsetting IF statements exit without OUTPUTting.
- RETURN exits the step but does OUTPUT.
- Explicit OUTPUT statement OUTPUTs when executed, but no implied OUTPUT at bottom.
- You may prefer those positive statements such as Subsetting IF, OUTPUT etc.
- You might use a negative statement such as DELETE to filter unwanted rows.

Note:

- WHERE also filters rows in engine, not in DATA step.

Outputting And Deleting Observations

Example: Only output records with more than 4000 in Sales.

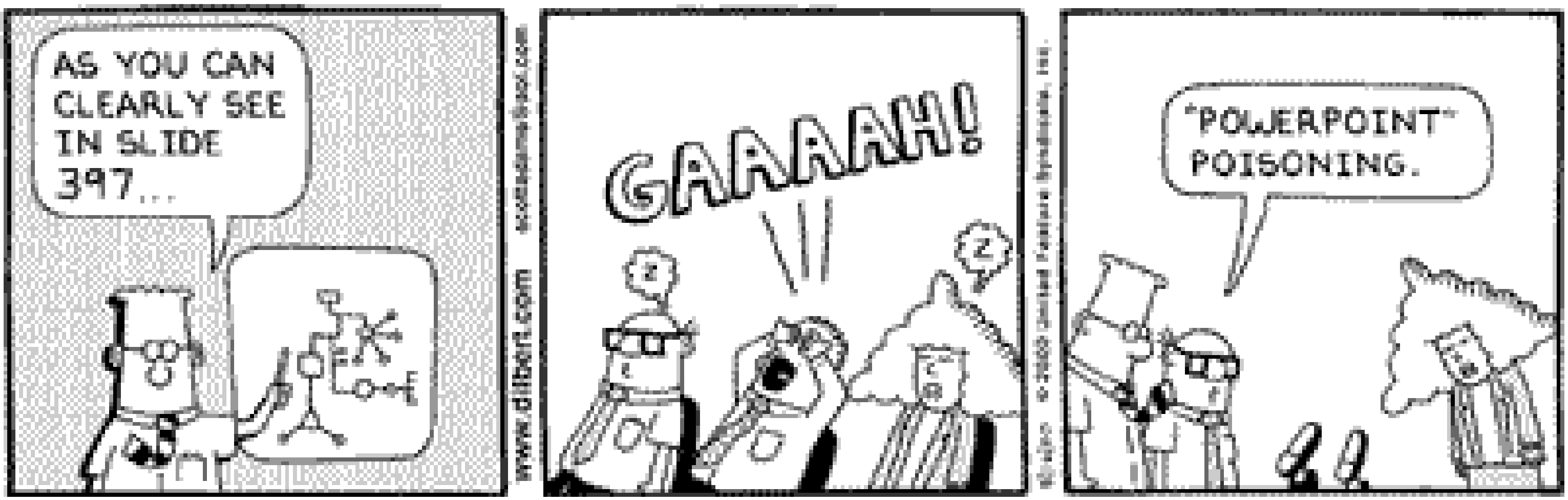
.

```
data softsale;
   if no input last time thru then stop
   initialize PDV
   infile rawin;
     if at EOF then stop
     input Name      $1-10
           Division  $12
           Years      15-16
           Sales      19-25
           Expense    28-34
           State     $36-37;
     If sales > 4000;
     If sales not gt 4000 then goto top of DATA step
     output to SAS Dataset
     goto top of DATA step
   run;
```

Note:

- WHERE also filters rows in engine, not in DATA step.

Losing the Crowd



Accumulating Totals in a DATA Step

We should be able to write formulas to count or sum across observations.

In a DATA step, read the following dataset and do the following:

- Count all employees and print running counts.
- Sum hours and print running sums.

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

Accumulating Totals in a DATA Step (continued)

```
data timecard;
  infile rawin;
  input Name $ Hours;
  Ktr=ktr+1;
  Hourstot=hourstot+hours;
run;

proc print data=timecard;
  title 'SOFTCO PAYROLL';
run;
```

Fileref rawin

JOE	40
PETE	20
STEVE	.
TOM	35

SOFTCO PAYROLL

OBS	Name	Hours	Ktr	Hourstot
1	JOE	40	.	.
2	PETE	20	.	.
3	STEVE	.	.	.
4	TOM	35	.	.

Question: Why didn't the above program work correctly?

Accumulating Totals in a DATA Step (continued)

Adding values across records has three problems:

1. All variables are set to missing on every record.
2. Totals normally need to start at 0, not missing.
3. When a missing value is added to any value, the result is a missing value.

The SUM statement is a special assignment statement to add values across observations.

SYNTAX:

variable+expression;

Notes:

- SUM variables start at 0, not missing.
- SUM variables are retained from pass to pass.
- Any missing values encountered are ignored.
- RETAIN and SUM function could also be used.

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags				
PDV				

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	0	0

Set at compile time.

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	0	0

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	JOE	40	0	0

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

```
   0
+  1
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	JOE	40	1	0

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

```
   0
+ 40
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	JOE	40	1	**40**

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	**JOE**	**40**	**1**	**40**

Obs	Name	Hours	Ktr	Hourstot
1	**JOE**	**40**	**1**	**40**

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	1	40

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

JOE	40
PETE	20
STEVE	.
TOM	35

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	1	40

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	**PETE**	**20**	1	40

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

```
   1
+  1
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	PETE	20	**2**	40

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

```
  40
+ 20
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	PETE	20	2	**60**

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	PETE	20	2	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	**PETE**	**20**	**2**	**60**

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	2	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE      40
PETE     20
STEVE     .
TOM      35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	2	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	STEVE	.	2	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

```
  2
+ 1
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	STEVE	.	**3**	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

Fileref RAWIN

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

60
\+ .

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	STEVE	.	3	**60**

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	STEVE	.	3	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	**STEVE**	**.**	**3**	**60**

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

JOE	40
PETE	20
STEVE	.
TOM	35

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	3	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV		.	3	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	**TOM**	**35**	3	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

  3
+ 1

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	TOM	35	4	60

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

60
\+ 35

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	TOM	35	4	**95**

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE    40
PETE   20
STEVE   .
TOM    35
```

	Name	Hours	Ktr	Hourstot
flags			RM	RM
PDV	TOM	35	4	95

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60
4	**TOM**	**35**	**4**	**95**

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

An Accumulation Solution

```
data timecard;
   infile rawin;
   input Name $ Hours;
   Ktr+1;
   Hourstot+hours;
run;

proc print data=timecard;
   title 'SOFTCO PAYROLL';
run;
```

fileref rawin

```
JOE     40
PETE    20
STEVE    .
TOM     35
```

SOFTCO PAYROLL

Obs	Name	Hours	Ktr	Hourstot
1	JOE	40	1	40
2	PETE	20	2	60
3	STEVE	.	3	60
4	TOM	35	4	95

Another Accumulation Solution

Retain and the SUM function can also accumulate correctly.

```
data timecard;
 infile rawin;
 input Name $ Hours;
 ktr=ktr+1;
 hourstot=sum(hourstot,hours);
 retain Ktr Hourstot 0;
run;
proc print data=timecard;
 title 'SOFTCO PAYROLL';
run;
```

Compiler Instructions

A series of statements that instruct the compiler to alter attributes.

- In general, declarative statements
- Can be coded in any order
- Allow the program to be very explicit in the definition of SAS structures.

Examples of these statements are:

- LENGTH statement to set a variables internal length
- INFORMAT to set input format
- FORMAT to set output display format
- LABEL to define a variable label
- ATTRIB to define any or all of the above in one statement
- DROP to indicate which variables are to be left behind on SAS file
- KEEP to indicate which variables to include on the SAS file
- RETAIN to set initial values and instruct SAS to never clear

Compiler Statements to Alter Variable Attributes

```
data softsale;
  infile rawin;
  length name $20;
  attr   division length=$2;
  format sales comma10.2;
  input name $1-10 division $12 years 15-16 sales 19-25
        expense 27-34;
  drop sales years;
  run;
```

input buffer

```
1234567890123456789012345678901234
BETH        H  12  4822.12   982.10
CHRIS       H   2   233.11    94.12
JOHN        H   7   678.43   150.11
```

PROGRAM DATA VECTOR

	NAME	DIVISION	YEARS	SALES	EXPENSE	_ERROR_	_N_
Type:	CHAR	CHAR	NUM	NUM	NUM	NUM	NUM
Length:	20	2	8	8	8	8	8
Format:				comma.			
Informat:							
Label:							
Flags:			D	D			
Value							

DATASET DESCRIPTOR PORTION (DISK)

	NAME	DIVISION	EXPENSE
Type:	CHAR	CHAR	NUM
Length:	20	2	8
Format:			
Informat:			
Label:			

DATASET DATA PORTION

BETH	H	982.10
CHRIS	H	94.12
JOHN	H	150.11

Debugging Features

There are a number of debugging features in the DATA step.

- Excellent Interactive DATA step debugger available.
- Simple DATA step statements that display data.
- The LIST statement can display the input buffer from the most recent INPUT.
- The FILE LOG with PUT can display any text and variable from the PDV in the SAS log.
- The PUTLOG statement can also display text to the SAS log.
- PROC CONTENTS and PROC PRINT/REPORT can be used to display the dataset descriptor and data values of the final dataset.

Notes:

- By putting the LIST and PUT/PUTLOG statements at strategic points in the DATA step may be the simplest and best way to debug.

A Debugging Example

Example: The program below selects 0 records. Which statement is causing the problem?

```
1234567890123456789012345678901234

BETH       H  12  4822.12   982.10
CHRIS      H   2   233.11    94.12
JOHN       H   7   678.43   150.11
```

```
data softsale;
  infile rawin missover;
  input name $1-10 division $12 years 15-16 sales 19-25
  expense 27-34;
  if sales > 4000;
  if division='h';
run;

NOTE: The data set WORK.SOFTSALE has 0 observations and 5 variables.
```

A Debugging Example

Now use PUTLOG to display messages and values around the IF statements.

```
data softsale;
   infile rawin missover;
   input name $1-10 division $12 years 15-16 sales 19-25 expense 27-34;
   putlog '$$$ before sales if ' _n_= sales= division=;
   if sales > 4000;
   putlog '$$$ before division if ' _n_= sales= division=;
   if division='h';
   run;
```

```
$$$ before sales if _N_=1 sales=4822.12 division=H
$$$ before division if _N_=1 sales=4822.12 division=H
$$$ before sales if _N_=2 sales=233.11 division=H
$$$ before sales if _N_=3 sales=678.43 division=H
```

```
NOTE: The data set WORK.SOFTSALE has 0 observations and 5 variables.
```

Other Data Step Statements

The SAS DATA step has many, many more statements that can read, write, and process data in almost any form.

- over 500 DATA step functions
- interfaces to the SAS macro system
- much more
- much written about those features
- beyond the scope of this paper to try to cover them all.
- the DATA step is an extremely versatile and full featured programming language.

Other Topics

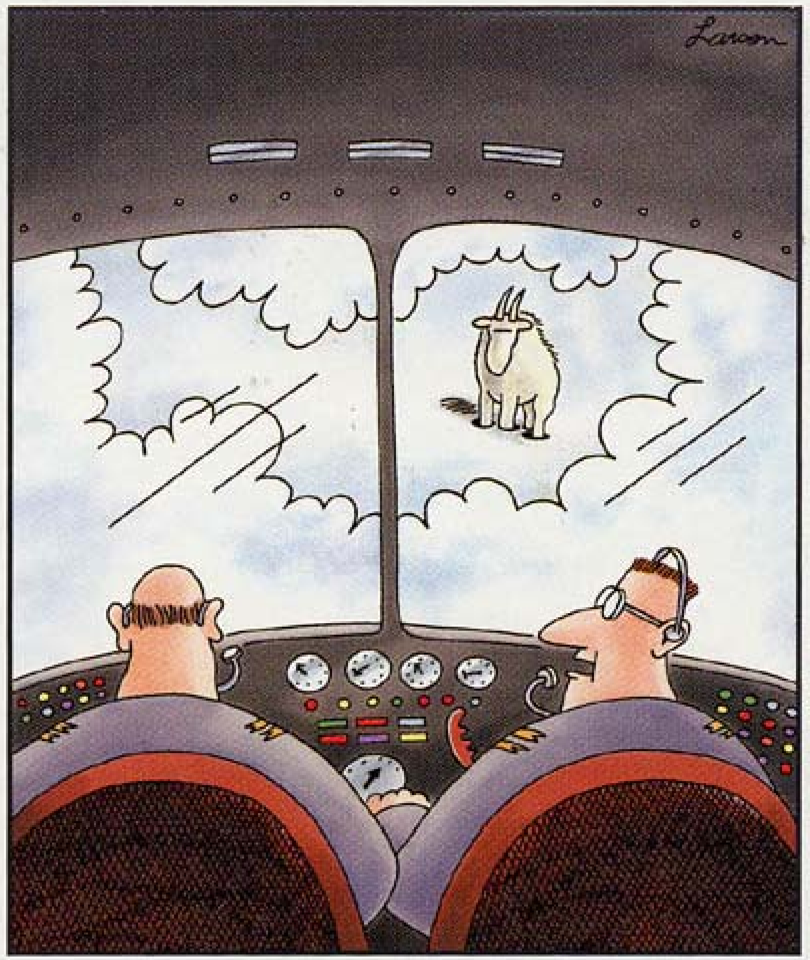
As powerful and well designed as the DATA step is, it is different from other languages.

- Perhaps a more standardized language such as SQL might be more auditable and more desirable.
- Terrible DATA step code can be written.
- Good design and adequate documentation make a well written program.
- PROC SQL is also a great tool that adds that language’s features to SAS.
- SAS programs can be well written programs that can be used for everything from one time programs to full fledged production programs.

Conclusions

The SAS DATA step is an excellent programming language with unique features and extremely versatile features.

Not Good On the Flight Home



"Say … what's a mountain goat doing way up here in a cloud bank?"

Contact Us

Questions, comments or to subscribe to the free "Missing Semicolon" newslettter.

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

Steven J. First

President

(608) 278-9964

FAX (608) 278-0065

sfirst@sys-seminar.com

www.sys-seminar.com

2997 Yarmouth Greenway Drive • Madison, WI 53711

QUALITY PARTNER™ SAS INSTITUTE INC.