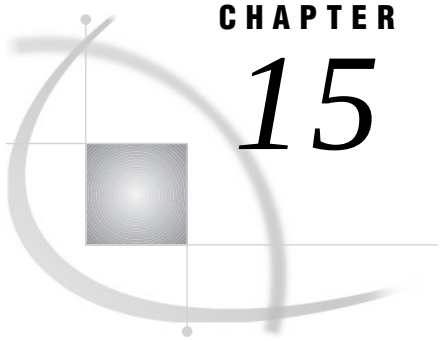

CHAPTER

15

Error Processing and Debugging

Definitions

SAS performs *error processing* during both the compilation and the execution phases of SAS processing. You can *debug* SAS programs by understanding processing messages in the SAS log and then fixing your code. You can use the DATA Step Debugger to detect logic errors in a DATA step during execution.

SAS recognizes five types of errors.

This type of error ...	occurs when ...	and is detected at ...
syntax	programming statements do not conform to the rules of the SAS language	compile time
semantic	the language element is correct, but the element may not be valid for a particular usage	compile time
execution-time	SAS attempts to execute a program and execution fails	execution time
data	data values are invalid	execution time

This type of error ...	occurs when ...	and is detected at ...
macro-related	you use the macro facility incorrectly	macro compile time or execution time, DATA or PROC step compile time or execution time

Types of Errors

Syntax Errors

Syntax errors occur when program statements do not conform to the rules of the SAS language. Examples of syntax errors include

- misspelling a SAS keyword
- using unmatched quotation marks
- forgetting a semicolon
- specifying an invalid statement option
- specifying an invalid data set option.

When SAS encounters a syntax error, it first attempts to correct the error by attempting to interpret what you meant, then continues processing your program based on its assumptions. If SAS cannot correct the error, it prints an error message to the log.

In the following example, the DATA statement is misspelled, and SAS prints a warning message to the log. Because SAS could interpret the misspelled word, the program runs and produces output.

```
date temp;
   x=1;
run;

proc print data=temp;
run;
```

Output 15.1 SAS Log: Syntax Error (misspelled key word)

```
1    date temp;
     ----
     14
WARNING 14-169: Assuming the symbol DATA was misspelled as date.

2       x=1;
3    run;

NOTE: The data set WORK.TEMP has 1 observations and 1 variables.
NOTE: DATA statement used:
      real time           0.17 seconds
      cpu time            0.04 seconds


4
5    proc print data=temp;
6    run;

NOTE: PROCEDURE PRINT used:
      real time           0.14 seconds
      cpu time            0.03 seconds
```

Some errors are explained fully by the message that SAS prints in the log; other error messages are not as easy to interpret because SAS is not always able to detect exactly where the error occurred. For example, when you fail to end a SAS statement with a semicolon, SAS does not always detect the error at the point where it occurs because SAS statements are free-format (they can begin and end anywhere). In the following example, the semicolon at the end of the DATA statement is missing. SAS prints the word ERROR in the log, identifies the possible location of the error, prints an explanation of the error, and stops processing the DATA step.

```
data temp
   x=1;
run;

proc print data=temp;
run;
```

Output 15.2 SAS Log: Syntax Error (missing semicolon)

```
1    data temp
2       x=1;
         -
         76
ERROR 76-322: Syntax error, statement will be ignored.

3    run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: DATA statement used:
      real time           0.11 seconds
      cpu time            0.02 seconds


4
5    proc print data=temp;
ERROR: File WORK.TEMP.DATA does not exist.
6    run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used:
      real time           0.06 seconds
      cpu time            0.01 seconds
```

Whether subsequent steps are executed depends on which method of running SAS you use, as well as on your operating environment.

Operating Environment Information: For more information, see the SAS documentation for your operating environment. △

Semantic Errors

Semantic errors occur when the form of the elements in a SAS statement is correct, but the elements are not valid for that usage. Semantic errors are detected at compile time and can cause SAS to enter syntax check mode. (For a description of syntax check mode, see "Syntax Check Mode" on page 192.)

Examples of semantic errors include

- specifying the wrong number of arguments for a function
- using a numeric variable name where only a character variable is valid
- using illegal references to an array.

In the following example, SAS detects an illegal reference to the array ALL.

```
data _null_;
   array all{*} x1-x5;
   all=3;
   datalines;
1 1.5
. 3
2 4.5
3 2 7
3 . .
;

run;
```

Output 15.3 SAS Log: First Example of a Semantic Error

```
      cpu time            0.02 seconds

1     data _null_;
2       array all{*} x1-x5;
ERROR: Illegal reference to the array all.
3       all=3;
4       datalines;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: DATA statement used:
      real time           2.28 seconds
      cpu time            0.06 seconds

10    ;
11
```

The following is another example of a semantic error. In this DATA step, the libref SOMELIB has not been previously assigned in a LIBNAME statement.

```
data test;
   set somelib.old;
run;
```

Output 15.4 SAS Log:Second Example of a Semantic Error

```
      cpu time            0.00 seconds

1   data test;
ERROR: Libname SOMELIB is not assigned.
2      set somelib.old;
3   run;
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TEST may be incomplete.  When this step was stopped
         there were 0 observations and 0 variables.
NOTE: DATA statement used:
      real time           0.17 seconds
```

Execution-Time Errors

Definition

Execution-time errors occur when SAS executes a program that contains data values. Most execution-time errors produce warning messages or notes in the SAS log but allow the program to continue executing. *The location of an execution-time error is usually given as line and column numbers in a note or error message.

Common execution-time errors include the following:

- illegal arguments to functions
- illegal mathematical operations (for example, division by 0)
- observations in the wrong order for BY-group processing

* When you run SAS in noninteractive mode, more serious errors can cause SAS to enter syntax check mode and stop processing the program.

- reference to a nonexistent member of an array (occurs when the array's subscript is out of range)
- open and close errors on SAS data sets and other files in INFILE and FILE statements
- INPUT statements that do not match the data lines (for example, an INPUT statement in which you list the wrong columns for a variable or fail to indicate that the variable is a character variable).

Out-of-Resources Condition

An execution-time error can also occur when you encounter an out-of-resources condition, such as a full disk, or insufficient memory for a SAS procedure to complete. When these conditions occur, SAS attempts to find resources for current use. For example, SAS may ask the user for permission to delete temporary data sets that might no longer be needed, or to free the memory in which macro variables are stored.

When an out-of-resources condition occurs in a windowing environment, you can use the SAS CLEANUP system option to display a requestor panel that enables you to choose how to resolve the error. When you run SAS in batch, noninteractive, or interactive line mode, the operation of CLEANUP depends on your operating environment. For more information about this system option, see CLEANUP in the "SAS System Options" chapter in *SAS Language Reference: Dictionary*, and in the SAS documentation for your operating environment.

Examples

In the following example, an execution-time error occurs when SAS uses data values from the second observation to perform the division operation in the assignment statement. Division by 0 is an illegal mathematical operation and causes an execution-time error.

```
options linesize=64 nodate pageno=1 pagesize=25;

data inventory;
   input Item $ 1-14 TotalCost 15-20
         UnitsOnHand 21-23;
   UnitCost=TotalCost/UnitsOnHand;
   datalines;
Hammers       440   55
Nylon cord    35    0
Ceiling fans  1155  30
;

proc print data=inventory;
   format TotalCost dollar8.2 UnitCost dollar8.2;
run;
```

Output 15.5 SAS Log: Execution-Time Error

```
      cpu time            0.02 seconds

1
2    options linesize=64 nodate pageno=1 pagesize=25;
3
4    data inventory;
5      input Item $ 1-14 TotalCost 15-20
6            UnitsOnHand 21-23;
7      UnitCost=TotalCost/UnitsOnHand;
8      datalines;
NOTE: Division by zero detected at line 12 column 22.
RULE:----+----1----+----2----+----3----+----4----+----5----+----
10   Nylon cord    35    0
Item=Nylon cord TotalCost=35 UnitsOnHand=0 UnitCost=. _ERROR_=1
_N_=2
NOTE: Mathematical operations could not be performed at the
      following places. The results of the operations have been
      set to missing values.
      Each place is given by:
      (Number of times) at (Line):(Column).
      1 at 12:22
NOTE: The data set WORK.INVENTORY has 3 observations and 4
      variables.
NOTE: DATA statement used:
      real time           2.78 seconds
      cpu time            0.08 seconds

12   ;
13
14   proc print data=inventory;
15      format TotalCost dollar8.2 UnitCost dollar8.2;
16   run;
NOTE: There were 3 observations read from the dataset
      WORK.INVENTORY.
NOTE: PROCEDURE PRINT used:
      real time           2.62 seconds
```

Output 15.6 SAS Output: Execution-Time Error

```
                        The SAS System                        1

                                Total     Units
     Obs    Item                Cost     OnHand    UnitCost

      1     Hammers           $440.00       55        $8.00
      2     Nylon cord         $35.00        0           .
      3     Ceiling fans     $1155.00       30       $38.50
```

SAS executes the entire step, assigns a missing value for the variable UnitCost in the output, and writes the following to the SAS log:

- a note
- the values stored in the input buffer
- the contents of the program data vector at the time the error occurred
- a note explaining the error.

Note that the values listed in the program data vector include the _N_ and _ERROR_ automatic variables. These automatic variables are assigned temporarily to each observation and are not stored with the data set.

In the following example of an execution-time error, the program processes an array and SAS encounters a value of the array's subscript that is out of range. SAS prints an error message to the log and stops processing.

```
options linesize=64 nodate pageno=1 pagesize=25;

data test;
   array all{*} x1-x3;
   input I measure;
   if measure > 0 then
      all{I} = measure;
   datalines;
1 1.5
. 3
2 4.5
;

proc print data=test;
run;
```

Output 15.7

```
      cpu time            0.02 seconds

1    options linesize=64 nodate pageno=1 pagesize=25;
2
3    data test;
4       array all{*} x1-x3;
5      input I measure;
6      if measure > 0 then
7         all{I} = measure;
8      datalines;
ERROR: Array subscript out of range at line 12 column 7.
RULE:----+----1----+----2----+----3----+----4----+----5----+----
10   . 3
x1=. x2=. x3=. I=. measure=3 _ERROR_=1 _N_=2
NOTE: The SAS System stopped processing this step because of
      errors.
WARNING: The data set WORK.TEST may be incomplete.  When this
         step was stopped there were 1 observations and 5
         variables.
NOTE: DATA statement used:
      real time           0.90 seconds
      cpu time            0.09 seconds

12   ;
13
14   proc print data=test;
15   run;
NOTE: There were 1 observations read from the dataset WORK.TEST.
NOTE: PROCEDURE PRINT used:
      real time           0.81 seconds
```

Data Errors

Data errors occur when some data values are not appropriate for the SAS statements that you have specified in the program. For example, if you define a variable as

numeric, but the data value is actually character, SAS generates a data error. SAS detects data errors during program execution and continues to execute the program, and does the following:

- writes an invalid data note to the SAS log.
- prints the input line and column numbers that contain the invalid value in the SAS log. Unprintable characters appear in hexadecimal. To help determine column numbers, SAS prints a rule line above the input line.
- prints the observation under the rule line.
- sets the automatic variable _ERROR_ to 1 for the current observation.

In this example, a character value in the Number variable results in a data error during program execution:

```
options linesize=64 nodate pageno=1 pagesize=25;

data age;
   input Name $ Number;
   datalines;
Sue 35
Joe xx
Steve 22
;

proc print data=age;
run;
```

The SAS log shows that there is an error in line 61, position 5–6 of the program.

Output 15.8 SAS Log: Data Error

```
      cpu time            0.01 seconds

1
2   options linesize=64 nodate pageno=1 pagesize=25;
3
4   data age;
5      input Name $ Number;
6      datalines;
NOTE: Invalid data for Number in line 61 5-6.
RULE:----+----1----+----2----+----3----+----4----+----5----+----
8   Joe xx
Name=Joe Number=. _ERROR_=1 _N_=2
NOTE: The data set WORK.AGE has 3 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.06 seconds
      cpu time            0.02 seconds

10   ;
11
12   proc print data=age;
13   run;
NOTE: There were 3 observations read from the dataset WORK.AGE.
NOTE: PROCEDURE PRINT used:
      real time           0.01 seconds
```

Output 15.9 SAS Output: Data Error

```
                          The SAS System                          1

                    Obs    Name     Number

                     1     Sue        35
                     2     Joe         .
                     3     Steve      22
```

You can also use the INVALIDDATA= system option to assign a value to a variable when your program encounters invalid data. For more information, see the INVALIDDATA= system option in *SAS Language Reference: Dictionary*.

Format Modifiers for Error Reporting

The INPUT statement uses the ? and the ?? format modifiers for error reporting. The format modifiers control the amount of information that is written to the SAS log. Both the ? and the ?? modifiers supress the invalid data message. However, the ?? modifier also sets the automatic variable _ERROR_ to 0. For example, these two sets of statements are equivalent:

- `input x ?? 10-12;`
- ```
  input x ? 10-12;
    _error_=0;
  ```

In either case, SAS sets the invalid values of X to missing values.

Macro-related Errors

Several types of macro-related errors exist:

- macro compile time and macro execution-time errors, generated when you use the macro facility itself
- errors in the SAS code produced by the macro facility.

For more information about macros, see *SAS Macro Language: Reference*.

Error Processing

Syntax Check Mode

If a DATA step has a syntax error, SAS can enter syntax check mode. SAS internally sets the OBS= option to 0 and the REPLACE/NOREPLACE option to NOREPLACE. When these options are in effect, SAS

- reads the remaining statements in the DATA step
- checks that statements are valid SAS statements
- executes global statements
- identifies any other errors that it finds
- creates the descriptor portion of any output data sets that are specified in program statements
  ```

- does not write any observations to new data sets that SAS creates
- does not execute most of the subsequent DATA steps or procedures in the program (exceptions include PROC DATASETS and PROC CONTENTS).

Note: Any data sets that are created after SAS has entered syntax check mode do not replace existing data sets with the same name. △

How Different Modes Process Errors

When SAS encounters most syntax or semantic errors, SAS underlines the point where it detects the error and identifies the error by number. If SAS encounters a syntax error when you run noninteractive SAS programs or batch jobs, it enters syntax check mode and remains in this mode until the program finishes executing.

When you run SAS in interactive line mode or in a windowing environment, syntax check mode is in effect only during the step where SAS encountered the error. When the system detects an error, it stops executing the current step and continues processing the next step.

Processing Multiple Errors

Depending on the type and severity of the error, the method you use to run SAS, and your operating environment, SAS either stops program processing or flags errors and continues processing. SAS continues to check individual statements in procedures after it finds certain kinds of errors. Thus, in some cases SAS can detect multiple errors in a single statement and may issue more error messages for a given situation, particularly if the statement containing the error creates an output SAS data set.

The following example illustrates a statement with two errors:

```
data temporary;
   Item1=4;
run;

proc print data=temporary;
   var Item1 Item2 Item3;
run;
```

Output 15.10 SAS Log: Multiple Program Errors

```
      cpu time            0.00 seconds

1  data temporary;
2     Item1=4;
3  run;
NOTE: The data set WORK.TEMPORARY has 1 observations and 1
      variables.
NOTE: DATA statement used:
      real time           0.10 seconds
      cpu time            0.01 seconds

4
5  proc print data=temporary;
ERROR: Variable ITEM2 not found.
ERROR: Variable ITEM3 not found.
6     var Item1 Item2 Item3;
7  run;
NOTE: The SAS System stopped processing this step because of
      errors.
NOTE: PROCEDURE PRINT used:
      real time           0.53 seconds
      cpu time            0.01 seconds
```

SAS displays two error messages, one for the variable Item2 and one for the variable Item3.

When running debugged production programs that are unlikely to encounter errors, you may want to force SAS to abend after a single error occurs. You can use the ERRORABEND system option to do this.

Using System Options to Debug a Program

You can use the following system options to control error handling (resolve errors) in your program:

BYERR	controls whether SAS generates an error message and sets the error flag when a _NULL_ data set is used in the SORT procedure.
DKRICOND=	controls the level of error detection for input data sets during the processing of DROP=, KEEP=, and RENAME= data set options.
DKROCOND=	controls the level of error detection for output data sets during the processing of DROP=, KEEP=, and RENAME= data set options and the corresponding DATA step statements.
DSNFERR	controls how SAS responds when a SAS data set is not found.
ERRORABEND	specifies how SAS responds to errors.
ERRORCHECK=	controls error handling in batch processing.
ERRORS=	controls the maximum number of observations for which complete error messages are printed.
FMTERR	determines whether SAS generates an error message when a format of a variable cannot be found.
INVALIDDATA=	specifies the value that SAS assigns to a variable when invalid numeric data is encountered.
MERROR	controls whether SAS issues a warning message when a macro-like name does not match a macro keyword.
SERROR	controls whether SAS issues a warning message when a defined macro variable reference does not match a macro variable.
VNFERR	controls how SAS responds when a _NULL_ data set is used.

For more information, see "SAS System Options" in *SAS Language Reference: Dictionary*.

Using Return Codes

In some operating environments SAS passes a return code to the system, but accessing return codes is specific to your operating environment.

Operating Environment Information: For more information about return codes, see the SAS documentation for your operating environment. △

Other Error Checking Options

To help determine your programming errors, you can use:

- the _IORC_ automatic variable that SAS creates (and the associated IORCMSG function) when you use the MODIFY statement or the KEY= data set option in the SET statement
- the ERROR= system option to limit the number of identical errors that SAS writes to the log
- the SYSRC and SYSMSG functions to return information when a data set or external file access function encounters an error condition
- the SYSRC and SYSERR macro variables

- log control options:

MSGLEVEL=	controls the level of detail in messages that are written to the SAS log.
PRINTMSGLIST	controls the printing of extended lists of messages to the SAS log.
SOURCE	controls whether SAS writes source statements to the SAS log.
SOURCE2	controls whether SAS writes source statements included by %INCLUDE to the SAS log.

Debugging with the DATA Step Debugger

You can debug logical errors in DATA steps by using the DATA step debugger. This tool allows you to issue commands to execute DATA step statements one by one and then pause to display the resulting variables' values in a window. By observing the results that are displayed, you can determine where the logic error lies. Because the debugger is interactive, you can repeat the process of issuing commands and observing results as many times as needed in a single debugging session. To invoke the debugger, add the DEBUG option to the DATA statement and execute the program. For detailed information about how to use the debugger, see "DATA Step Debugger" in *SAS Language Reference: Dictionary*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Language Reference: Concepts*, Cary, NC: SAS Institute Inc., 1999. 554 pages.

SAS Language Reference: Concepts

ISBN 1–58025–441–1

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

1st printing, November 1999