Print

# Summary of Lesson 8: Reading Raw Data Files

This summary contains topic summaries, syntax, and sample programs.

## Topic Summaries

*To go to the movie where you learned a task or concept, select a link.*

### Introduction to Reading Raw Data Files

A raw data file is an external text file that contains one record per line, and a record typically contains multiple fields. The fields can be delimited or arranged in fixed columns. Typically, there are no column headings. The file is usually described in an external document called a *record layout*.

In order for SAS to read a raw data file, you must specify the location of each data value in the record, along with the names and types of the SAS variables in which to store the values. Three styles of input are available: list input, column input, and formatted input. List input reads delimited files, and column and formatted input read fixed column files. List input and formatted input can read both standard and nonstandard data, and column input can read only standard data. In this course, we are reading delimited raw data files, so list input is used.

### Reading Standard Delimited Data

You use a DATA step with INFILE and INPUT statements to read data from a raw data file. The INFILE statement identifies the name and location of the input file. You use the DLM= option if the file has a delimiter other than a blank space. The INPUT statement tells SAS how to read the values, and specifies the name and type for each variable to be created. In the INPUT statement, you list the variables in the order that the corresponding values appear in the raw data file, from left to right. You specify character variables by adding a dollar sign after the variable name. With list input, the default length for all variables is 8 bytes, regardless of type.

```
DATA output-SAS-data-set-name;
      INFILE 'raw-data-file-name' DLM='delimiter';
      INPUT variable1 <$> variable2 <$> ... variableN <$>;
RUN;
```

SAS processes the DATA step in two phases: compilation and execution. During compilation, SAS creates an input buffer to hold a record from the raw data file. The input buffer is an area of memory that SAS creates only when reading raw data, not when reading a SAS data set. SAS also creates the PDV, an area of memory where an observation is built. In addition to the variables named in the INPUT statement, SAS creates the iteration counter, **_N_**, and the error indicator, **_ERROR_**, in the PDV. These temporary variables are not written to the output data set. At the end of the compilation, SAS creates the descriptor portion of the output data set.

At the start of the execution phase, SAS initializes the PDV and then reads the first record from the raw data file into the input buffer. It scans the input buffer from non-delimiter to delimiter and assigns each value to the corresponding variable in the PDV. SAS ignores delimiters. At the bottom of the DATA step, SAS writes the values from the PDV to the new SAS data set and then returns to the top of the DATA step.

Truncation often occurs with list input, because character variables are created with a length of 8 bytes, by default. You can use a LENGTH statement before the INPUT statement in a DATA step to explicitly define the length of character variables. Numeric variables can be included in the LENGTH statement to preserve the order of variables, but you need to specify a length of 8 for each numeric variable.

```
LENGTH variable(s) <$> length;
```

### Reading Nonstandard Delimited Data

You can use modified list input to read standard and nonstandard data from a delimited raw data file. Modified list input uses an informat and a colon format modifier for each field to be read. An informat tells

SAS how to read data values, including the number of characters. When SAS reads character data, a standard character informat, such as $12., is often used instead of a LENGTH statement. With list input, the data fields vary in length. The colon format modifier tells SAS to ignore the specified length when it reads data values, and instead to read only until it reaches a delimiter. Omitting the colon format modifier is likely to result in data errors.

**INPUT** *variable* <$> *variable* <:*informat*>**;**

An informat is required to read nonstandard numeric data, such as calendar dates, and numbers with dollar signs and commas. Many SAS informats are available for nonstandard numeric values. Every informat has a width, whether stated explicitly or set by default.

When reading a raw data file, you can use a DROP or KEEP statement to write a subset of variables to the new data set. You must use a subsetting IF statement to select observations, because the variables are not coming from an input SAS data set. You can use LABEL and FORMAT statements to permanently store label and format information in the new data set.

A DATA step can also read instream data, which is data that is within a SAS program. To specify instream data, you use a DATALINES statement in a DATA step, followed by the lines of data, followed by a null statement.

**DATALINES;**
*<data line 1>*
*<data line 2>*
...
;

**Validating Data**

When data values in the input file aren't appropriate for the INPUT statement in a program, a data error occurs during program execution. SAS records the error in the log by writing a note about the error, along with a ruler and the contents of the input buffer and the PDV. The variable **_ERROR_** is set to *1*, a missing value is assigned to the corresponding variable, and execution continues.

You can use the DSD option in the INFILE statement if data values are missing in the middle of a record. When you use the DSD option, SAS assumes that the file is comma delimited, treats consecutive delimiters as missing data, and allows embedded delimiters in a field that is enclosed in quotation marks. If you have missing data values at the end of a record, you can use the MISSOVER option in the INFILE statement. SAS sets the variable values to missing.

**INFILE** '*raw-data-file-name*' <**DLM=**> <**DSD**> <**MISSOVER**>**;**

## Sample Programs

### Creating a SAS Data Set from a Delimited Raw Data File

```
data work.sales1;
   infile "&path/sales.csv" dlm=',';
   input Employee_ID First_Name $
         Last_Name $ Gender $ Salary
         Job_Title $ Country $;
run;

proc print data=work.sales1;
run;
```

### Specifying the Lengths of Variables Explicitly

```
data work.sales2;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path/sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
```

```
         Gender $ Salary Job_Title $ Country $;
run;

proc contents data=work.sales2;
run;

proc print data=work.sales2;
run;


data work.sales2;
   length Employee_ID  8 First_Name $ 12
          Last_Name $ 18 Gender $ 1
          Salary  8 Job_Title $ 25
          Country $ 2;
   infile "&path/sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;

proc contents data=work.sales2 varnum;
run;

proc print data=work.sales2;
run;
```

**Specifying Informats in the INPUT Statement**

```
data work.sales2;
   infile "&path/sales.csv" dlm=',';
   input Employee_ID First_Name :$12. Last_Name :$18.
         Gender :$1. Salary Job_Title :$25. Country :$2.
         Birth_Date :date. Hire_Date :mmddyy.;
run;

proc print data=work.sales2;
run;
```

**Subsetting and Adding Permanent Attributes**

```
data work.subset;
   infile "&path/sales.csv" dlm=',';
   input Employee_ID First_Name :$12.
         Last_Name :$18. Gender :$1. Salary
         Job_Title :$25. Country :$2.
         Birth_Date :date. Hire_Date :mmddyy.;
   if Country='AU';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary dollar12. Hire_Date monyy7.;
run;

proc print data=work.subset label;
run;
```

**Reading Instream Data**

```
data work.newemps;
   input First_Name $ Last_Name $
         Job_Title $ Salary :dollar8.;
   datalines;
Steven Worton Auditor $40,450
Merle Hieds Trainee $24,025
Marta Bamberger Manager $32,000
;

proc print data=work.newemps;
run;
```

```
data work.newemps2;
   infile datalines dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary :dollar8.;
   datalines;
Steven,Worton,Auditor,$40450
Merle,Hieds,Trainee,$24025
Marta,Bamberger,Manager,$32000
;

proc print data=work.newemps2;
run;
```

### Reading a Raw Data File That Contains Data Errors

```
data work.sales4;
   infile "&path/sales3inv.csv" dlm=',';
   input Employee_ID First $ Last $
         Job_Title $ Salary Country $;
run;

proc print data=work.sales4;
run;
```

### Reading a Raw Data File That Contains Missing Data

```
data work.contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path/phone2.csv" dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=work.contacts noobs;
run;
```

### Reading a Raw Data File Using the MISSOVER Option

```
data work.contacts2;
   infile "&path/phone.csv" dlm=',' missover;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts2 noobs;
run;


data work.contacts2;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path/phone.csv" dlm=',' missover;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts2 noobs;
run;
```

*SAS Programming 1: Essentials*

Close