

Paper SBC-138 - 2008

Using the SAS® INPUT Statement for Not So Friendly File Formats

Inga Allred and Barbara Felts,
RTI International, Research Triangle Park, NC

ABSTRACT

The SAS® INPUT statement is simple, yet is a very powerful tool allowing data from many types of formats to be included in SAS datasets. However, it often leaves users confused and uncertain about how exactly the statement is best utilized. In this paper, we intend to show how to use a basic SAS® INPUT statement and will differentiate between those pesky trailing @ and double trailing @@'s. We will also give explanations on the many different ways of handling missing values in input data by using INFILE options like FLOWOVER, STOPOVER, MISCOVER, and TRUNCOVER. The examples used throughout this paper will showcase how SAS® can be efficiently used in transforming documentation into a useful SAS® INPUT statement creating a single SAS® record from multiple lines of input text.

INTRODUCTION

Documentation comes in many different formats, and usually, it is in a format that is difficult and cumbersome to use. With the creative use of SAS® INPUT statements combined with some simple syntax, we have been able to take some impractical documentation and turn it into a functioning SAS® INPUT statement that will transform text data into a useable SAS® dataset resulting in a large amount of time and tedious labor saved.

BACKGROUND

Our data are collected for a large national survey via paper and pencil instrument. We use a data entry software package to key the data we receive. Some data entry packages are very effective in creating a SAS® INPUT statement from the written data entry code; others do not have that capability. The data entry software package we use for our survey purposes is one of the latter. Our survey has over 400 variables and we needed an efficient, quick and easy way to transform data entry documentation into a useful SAS® INPUT statement that can be used in creating a SAS® dataset. We will show how we created two separate SAS® INPUT statements: one to format the documentation into a SAS® INPUT statement, and the resulting SAS® INPUT statement used to transform a single data record created from multiple raw data lines into a SAS® dataset.

PROBLEM

Our problem occurs because we have data records containing over 400 variables and this is our only documentation. It would take hours to manually extract that many individual variables. When creating a SAS® INPUT statement, the variable name, length and data type are needed. We can retrieve all of the required information from the documentation above by using a SAS® INPUT statement.

The documentation output from our data entry package can be seen in Figure 1.

Figure 1: Subset of the output documentation from our Data Entry Software Package

Format 001 -- Output Control:					
Type	Record	Column	Length	Contents	
Field	1	1	6	String	FMT_001
Field	1	7	6	Date	KEY_DATE
Field	1	13	2	Number	CI_DR_M
Field	1	15	2	Number	CI_DR_D
Field	1	17	2	Number	CI_DR_Y
Field	1	19	20	String	CI_CQC
Field	1	39	1	String	ESC_001
End of Record	1	40	2	End of Record	
Format 002 -- Output Control:					

Type	Record	Column	Length	Contents		
Field	1	1	3	Number	FMT_002	Integer
Field	1	4	2	Number	CI_ISTH	Integer
Field	1	6	2	Number	CI_ISTM	Integer
Field	1	21	1	Number	CI_S1	Integer
Field	1	22	1	Number	CI_S2	Integer
Field	1	23	1	Number	CI_S3	Integer
Field	1	30	1	String	ESC_002	
End of Record	1	31	2	End of Record		

PROGRAM STEPS

The documentation in Figure 1 above gives us all the information needed to create an effective SAS® INPUT statement. The documentation provides us with the variable name, the column in which its data starts, the variable's length, and its type, indicating whether it's a character or numeric variable. Since our documentation is in a fixed column format, we can use the SAS® INPUT statement to create our final SAS® INPUT statement. Our first SAS® INPUT statement used to read the documentation can be seen in Figure 2.

Figure 2: Initial SAS® INPUT statement

```
data layout;
    infile "C:\doc_layout.txt" trunccover;
        input @1 line      $char5.
              @23 column  $char5.
              @31 varlen   $char5.
              @39 AorN     $char1.
              @49 Varname  $char10.
    ;
    if line = "Field";
run;
```

In Figure 2, we read a text file called DOC_LAYOUT.TXT that is located in the C:\ folder. When reading this text data, we chose to use the TRUNCOVER option which tells SAS® to read a variable length record, just in case the input data is not the length that we think it is. There are other options that are available for use with the INFILE statement are FLOWOVER, MISCOVER, and STOPOVER. FLOWOVER is used in the case where a variable's data needs to be captured from the current line of data that is being read and the next line as well. MISCOVER lets SAS® read past the end of the data line and sets missing values for those variables read past the end of the line. STOPOVER says that reading past the end of the file is not allowed and will return an end of file error.

In our INPUT statement, the first variable we want to look at is called *LINE*. Since our DOC_LAYOUT.TXT file contains other documentation that we may not need, we use this *LINE* variable to insure that the line of data we are reading actually contains the variable information we need. The *COLUMN* variable contains the start column for the data. The *VARLEN* variable contains the actual character length of the data. The *AORN* variable contains the information about the data being character or numeric data. Lastly, the *VARNAME* field contains the name that we will give the variable holding our data. At the end of the data step, we subset our documentation based on the *LINE* variable, keeping only those lines of documentation that begin with the word "Field", as we know those lines contain our variable information.

After this data step, we are able to create our final SAS® INPUT statement that will be used as our survey data input statement. The code is in Figure 3.

Figure 3: Code to create final SAS® INPUT statement

```
data _null_;
    set layout end=last;
    file "c:\layout_input.txt";
```

```

col = "@" || column;
varlenx = trim(left(varlen)) || trim(left("."));

if _n_ = 1 then
  put "INPUT";

  put @5 col
      @10 varname @;

if AorN ^= "N" then do;
  put @25 "$char" @;
end;

  put @30 varlenx;

check = substr(varname,1,3);
if check = "ESC" and ^last then
  put @1 "/";
else if last then
  put @1 ";";

run;

```

The first part of this code creates the beginning part of our SAS® INPUT statement, the *COL* variable is used as a column pointer. The variable *VARLENX* is created to represent the input length of the variable and the period is concatenated to the end of the length as required by SAS®. If it is the first observation of the dataset, the word INPUT is put into the resulting output file to mark the beginning of our final SAS® INPUT statement. After that, the column and the variable name are put to the output file. Next, we have to use the SAS® line-hold specifier "@". This tells SAS® that we are not quite finished with that particular output line and that we need to execute a conditional statement to see what information comes next. There is also a double trailing @@. The trailing @ will hold the data line for the next iteration of the same data step. The double trailing @@ holds the data line across data steps. The next statement checks to see if the variable is a character or numeric variable using the variable *AORN*. We then check to see if we are at the end of our multiple record input file by looking for a specific set of characters, in our case, those characters are "ESC". If we are at the end of our input record, we simply use the syntax character "/" to tell SAS® that we need to continue with our input but we need to go to the next line of the input text file. However, if it is the last record of the dataset, we simply put a ";" to mark the end of our input statement.

When the code in Figure 3 is executed, we get the results seen in Figure 4.

Figure 4: Resulting SAS® INPUT Statement

```

INPUT
  @1   FMT_001           $char6.
  @7   KEY_DATE          $char6.
  @13  CI_DR_M           2.
  @15  CI_DR_D           2.
  @17  CI_DR_Y           2.
  @19  CI_CQC            $char20.
  @29  ESC_001           $char1.
/
  @1   FMT_002           3.
  @4   CI_ISTH           2.
  @6   CI_ISTM           2.
  @21  CI_S1             1.
  @22  CI_S2             1.
  @23  CI_S3             1.
  @30  ESC_002           $char1.
;

```

This SAS® INPUT statement is then used to create the final dataset from our raw data captured by the data entry program. Although our example only shows two lines of input data and fourteen variables, the actual text files we receive have 25 lines of input data and over 400 variables for each respondent. We use this SAS® INPUT statement to combine these 25 lines of text and create one record for each respondent in our final SAS® dataset.

SUMMARY

Although the concept is very basic, even for the beginning SAS® user, it shows how to cleverly transform unfriendly documentation into a very effective tool. With a SAS® INPUT statement, we read our documentation and use it to create an additional SAS® INPUT statement that reads our text survey data. Without the SAS® INPUT statement, our task would have taken hours, been monotonous and prone to human error.

CONTACT INFORMATION

Inga Allred, irb@rti.org, (919) 541-7072

Barbara Felts, bfelts@rti.org, (919) 541-6938

Each author can be reached at the mailing address:

RTI International
P.O. Box 12194
Research Triangle Park, NC 27709-2194
www.rti.org

TRADEMARK NOTICE

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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