Using _INFILE_, Sunstr, Translate and TranWrd to Manipulate Strings of Text Within the Input Buffer

Peter Wobus, Westat 1650 Research Blvd., Rockville, MD 20850-3195

Abstract

Version 8 of SAS offers many useful enhancements, among them the automatic _INFILE_ variable. As SAS points out in its documentation, the _INFILE_ variable can be used to reference the contents of the current input buffer for the most recent execution of the INFILE statement. This paper shows how to manipulate strings of text within the input buffer using the _INFILE_ variable and output a reformatted ASCII file.

Keywords

INFILE , translate, SubStr, TranWrd

An Overview

The _INFILE_ variable is like any other character variable in SAS. Its default length is 32,767, the maximum for a character variables allowed by SAS. Also like other SAS variables, _INFILE_ can be used in an assignment statement or in a PUT statement modified by a format.

The program illustrated here will make liberal use of the following SAS functions: SubStr, Translate, and TranWrd.

Review of the Functions Used

SubStr(var, position, length)

The SubStr function reads in a character variable and returns a portion of it defined by the values in *position* and *length*. For example, in order to extract the substring "sun" from "sunset" we would tell SAS to start at the first position and read 3 characters ending at the third position.

substr('"sunset", 1, 3) →"sun"

To extract "set" from "sunset" we would tell SAS to start at the fourth position and and read 3 characters:

SubStr('"sunset", 4, 3) → "set"

Translate(source, to, from)

The Translate function replaces specific characters in a character expression. The character expression is specified in *source*, the characters to be replaced in *from*, and the substitute characters in *to*. To illustrate, in the following example Translate

converts "south" to "north".

translate('south','nor','sou')→ "north"

Tranwrd(source, from, to)

The Tranwrd function replaces a group of characters in a character expression. It differs from the Translate function, which searches for every occurrence of individual characters within a string, converting them to another character. In contrast the Tranwrd function acts on groups of characters or words. Note too the difference in the order of the arguments in Tranwrd compared to Translate. In this example Tranwrd performs a conversion on a portion of the source string "applepie" and returns the string "cherrypie".

Tranwrd("applepie","apple","cherry")→ "cherrypie"

The Program

The program outlined here manipulates the contents of an ASCII file within the input buffer using _INFILE_. It creates a variable called OUTFILE that contains the reformatted values from _INFILE_ and outputs the results to an ASCII file.

The first step is to read the external ASCII file using an INPUT statement. In this case the input ASCII file has 95 columns of data and a length of 809. The LRECL option specifies the logical record length of the input file. In addition, PAD tells SAS to insert blanks for rows that are shorter than 809 columns. The variable OUTFILE eventually will contain the entire output file. The LENGTH statement will give it a length of 193.

```
data _null_;
    infile  "external-file" pad lrecl=809 ;
    input ;
    length OUTFILE $193  ;
```

The next several lines of code use the automatic _INFILE_ variable to modify the contents of the input buffer. Several changes will be made including repositioning columns of data, recoding missing values, changing numeric values to character values and replacing blanks with leading zeros.

The following line of code extracts from columns 7 and 8 of the input file the record ID and writes the result to columns 181 and 182 of OUTFILE.

```
subStr(OUTFILE,181,10) = (subStr(_INFILE_,7,2)) ;
```

On the input file missing values were coded “99”. The next two lines of code illustrate how the 99s on the input file are replaced with blanks within the input buffer and written to columns 41 and 42 of OUTFILE.

```
If ( subStr(_INFILE_,31,2) eq "99" ) then
    subStr(OUTFILE,41,1) = "" ;
else subStr(OUTFILE,41,1) = subStr(_INFILE_,32,1) ;

If ( subStr(_INFILE_,39,2) eq "99" ) then
    subStr(OUTFILE,42,1) = "" ;
 else subStr(OUTFILE,42,1) = subStr(_INFILE_,40,1) ;
```

Another objective is to write all numeric values on the input file as character values on the output file. Column 44 of the input file contains numeric ethnicity data. The following code writes these character values to columns 44 through 49 on OUTFILE.

```
Select;
    when ( subStr(_infile_,55,2) eq "99"
      subStr(outfile,44,1) = " " ;
    when ( subStr(_infile_,56,1) eq "1" )
      subStr(outfile,44,1) = "A" ;
    when ( subStr(_infile_,56,1) eq "2" )
      subStr(outfile,45,1) = "B" ;
    when ( subStr(_infile_,56,1) eq "3" )
      subStr(outfile,46,1) = "C" ;
    when ( subStr(_infile_,56,1) eq "4" )
      subStr(outfile,47,1) = "D" ;
    when ( subStr(_infile_,56,1) eq "5" )
      subStr(outfile,48,1) = "E" ;
    when ( subStr(_infile_,56,1) eq "6" )
      subStr(outfile,49,1) = "F" ;
    when ( subStr(_infile_,56,1) eq "7" )
      subStr(outfile,50,1) = "G" ;
    when ( subStr(_infile_,56,1) eq "8" )
      subStr(outfile,51,1) = "H" ;
Otherwise ; End;
```

The next several lines of code deal with blanks or 99s in columns 76 and 77 on _INFILE_. Non-missing in these columns have either one or two digits and do not have leading zeros.

Recoded output is written to columns 54 and 55 on OUTFILE. Where there are 99s in columns 76 and 77 on _INFILE_, two zeros are written to columns 54 and 55 on OUTFILE. Where there is a blank in column 76 and a non-missing value in column 77 on _INFILE_, a zero is written to column 54 and the non-missing value to columns 55 on OUTFILE.

```
if (subStr(_INFILE_,76,2) = " ")  then
    subStr(_INFILE_,76,2) = "00" ; else
if (subStr(_INFILE_,76,1) = "")  then
    subStr(_INFILE_,76,1) = "0" ; else
if (subStr(_INFILE_,76,2) = "99") then
    subStr(_INFILE_,76,2) = " "  ;
subStr(OUTFILE,54,2) = subStr(_INFILE_,76,2) ;
```

For the remaining columns of data in _INPUT_ (95 through 809) the only required operation is to remove missing values and output the results to OUTFILE, starting with column 59. Missing values are denoted by blanks and by 99. Using the TranWrd function we substitute an asterisk preceded by a single blank for the 99s on _INFILE_. The blanks and asterisks will serve as a temporary placeholders until the OUTFILE is finalized. Later they will be removed. Within a DO loop we substitute an explanation point (!) for existing blanks. Again, the explanation points serve as temporary placeholders and will later be removed from the OUTFILE.

```
SubStr(_INFILE_,95,714) =
    TranWrd(subStr(_INFILE_,95,714 ),"99"," *");
Do i=96 to 808 by 8;
    if (subStr(_INFILE_,i,1) eq " ") then
    subStr(_INFILE_,i,1) = "!" ;
End;
    subStr(OUTFILE,59,95) =
compress(subStr(_INFILE_,95,714)) ;
```

Using the SubStr function the contents of _INFILE_ are written to OUTFILE starting with column 59.

```
subStr(OUTFILE,59,95 ) =
    compress(SubStr(_INFILE_,95,714 )) ;
```

The OUTFILE is now mostly complete, however it still contains asterisks with leading blanks and explanation points as place holders for missing values. The following two lines of code remove these values from OUTFILE, substituting blanks.

```
 OUTFILE = translate(OUTFILE," ","*") ;

 OUTFILE = translate(OUTFILE," ","!") ;
```

In this last section the contents of OUTFILE are output to a text file named NewFile.dat. The length of the file is 193 .

```
  file "\NewFile.dat" lrecl = 193 ;
  put OUTFILE $Char193.;

  Run ;
```

REFERENCE

SAS Institute, Inc., SAS Language Reference, Version 8, Cary, NC: SAS Institute Inc.,1999. 1256 pp.

AUTHOR

If you have any questions or comments, please write or call:

Peter Wobus
Westat
1650 Research Blvd.
Rockville, MD 20850-3195
(301) 610-4997 (Voice)
wobusp@westat.com