Paper 86-28

More _Infile_ Magic

Peter Crawford, Crawford Software Consultancy Limited, Surrey, UK

ABSTRACT

Some facilities, limited to INPUT statements, have become more generally available (for example, reorganizing data from other rdbms) because _INFILE_ can be updated (since at least v8). This is hardware-platform independent so it is available everywhere INPUT statements are available (more difficult in SAS/SCL). This paper is targeted at programmers and other users of intermediate and advanced skillsets

INTRODUCTION

This paper looks at lesser-appreciated and understood aspects of _INFILE_ showing how usefully this has been extended.

OUTLINE

Intro.	
Nutshell	…If I could describe this in one sentence...
Background	Briefly, what went before
Syntax	The rules, the rules
Examples & demo	Scanning mixed case, DSD extract from string, Name-constant parsing
Conclusions	
Questions	

IN A NUTSHELL

For input files, the SAS System® has 25+ years of polish on the good ideas. Now that we can update _INFILE_ buffers, these facilities are available throughout DATA step processing - not just on INPUT statements.

BACKGROUND

We have some special ways of parsing data with INPUT statements

1. Named input (name=value pairs)
2. Input scanning (input @string_expression target_variable)
3. CSV support (option DSD)

To apply these techniques to data, other than external, requires complexity like:

1. Functions SCAN(), SUBSTR(), INDEX(), INPUT() and PUT() and sometimes the concatenation operator (||) as well

or

2. DATA _null_ step to write to an external file with the text. Then a DATA step to use INPUT statements reading that data.

Fortunately we don't need this complexity any longer.

However, the alternatives I describe are limited to a DATA step. Perhaps further updates might make some of this available outside of DATA steps.

SYNTAX

The INFILE statement is enhanced allowing a name to be assigned to the buffer for that infile.

```
*! do NOT define Length my_buffer $32000 ;
Infile dumfile  infile= my_buffer
          col= col dlm= my_dlm
         lrecl= 32000 truncover  /*etc*/ ;
input @1 @@ ;
my_buffer = my_problem_string ;
```

When we don't need to distinguish between buffers, option _INFILE_= need not be used. Instead, we can use the _INFILE_ pseudo-variable.

EXAMPLE 1

Case insensitive scan;

* when the data may have any case at any point in the string;

```
my_buffer = upcase( my_problem_string );
length found_it $32;
input @1 @'MYPROBLEM' found_it @@;
```

EXAMPLE 2

Using INFILE statement option DSD to parse a string which has delimiters embedded as data within quotes ;

* always start by loading buffer and/or ensuring you know where you are within it... @ column 1 is easy;

```
input @1 @@ ;
```

* Load my difficult string into the buffer;

```
my_buffer = problem_string ;
```

* Establish the data delimiter ;

```
my_dlm = '05'x; * ebcdic TAB character;
```

or

```
my_dlm = '090D'x ; * ask me why !! ;
```

* OK, I need this complex set of delimiters when I save Excel sheets as tab-delimited text files directly into unix. Excel delivers also the carriage return '0D'x that the SAS System on unix treats as data. Placing '0D'x as a delimiter along with the '09'x allows INFILE processing to hide it;

EXAMPLE 3

* Skip over the first 20 "data columns" then read;

```
_infile_ = string_of_many_columns ;
do i = 1 to 20; input dum $ @@; end;
input wanted  @@;
```

* The data I wanted (column21)!;

EXAMPLE4 - UNKNOWN

As more "foreign" rdbms introduce column names outside the "near-standard" 32 character form that we might refer as VALIDVARNAME= v7, we start needing to process lists of variables which are more like VALIDVARNAME= any. Then a variable list ceases to be scanned easily with functions INDEXW() and SCANW(). For example:

```
"Business date"n, "A/C, B/Fwd"n
```

This is the header for a very simple list of two columns, but it needs much more careful handling than a VALIDVARNAME= v7 list. However with this approach the code can reduce to:

* first put the trouble into a table;

```
data demo1;
   infile datalines truncover;
   input trouble  $char100. ;
   put  (_all_)( = $quote3000. );*checking;
datalines4;
"Business  date"n, "A|C; B|Fwd"n
"Business, date",  "A|C, B|Fwd"
"Business, date"n, "A|C, B|Fwd"n
;;;;
```

* next, extract the column names out of trouble;

```
filename dumfile 'x.x'; *** some dummy file;
data demo2;
      *...other processing;
   set demo1; *** load trouble-some column;
   infile dumfile dsd truncover;
   input @1 @@;       *establishing buffer;
   _infile_ = trouble ;      * load buffer;
   length name1 name2 $32;   *  define and;
   input  name1 name2 @@ ;   *  read names;
   put ( _all_ )(/=) ;     * just checking;
      *...other processing;
run;
```

Which produces these lines in the log

```
trouble="Business  date"n, "A|C; B|Fwd"n
```

```
name1="Business  date"n
name2="A|C; B|Fwd"n

trouble="Business, date",  "A|C, B|Fwd"
name1=Business, date
name2=A|C, B|Fwd

trouble="Business, date"n, "A|C, B|Fwd"n
name1="Business
name2=date"n
NOTE:
```

The last "trouble" fails because option DSD does not recognise the standard "name-constant" with the trailing "n". I expect in the next release, there is a function to cope ! Sometimes we should know the rules of the data we try to process.

CONCLUSION

Update-able infile buffers allow new solution designs. Some of them seem to work "just like magic"!

Here I have scratched the surface. The full potential will not be realized until the potential is fully understood.

REFERENCES

SAS Online Doc for access to the _INFILE_ buffer
"http://v8doc.sas.com/sashtml/lgref/z0146932.htm#z1017828"

ACKNOWLEDGMENTS (HEADER 1)

Acknowledgment must go to certain individuals on SAS-L whose comments encouraged me to offer this paper. In particular, Ian Whitlock and Howard Schreier, but of course there are many more encouraging voices on SAS-L… too many to name them all.

CONTACT INFORMATION (HEADER 1)

Your comments and questions are valued and encouraged. Contact the author at:

Peter Crawford
Crawford Software Consultancy Limited
Address: 31 Sefton Road,
Croydon, Surrey CR0 7HS, UK
Phone: +44 7802 732 254
Fax: +44 20 82400880
Email: Peter@CrawfordSoftware.demon.co.uk