USING SAS SOFTWARE TO COMPARE STRINGS OF VOLSERS IN A JCL JOB AND A TSO CLIST

RANDALL M NICHOLS, Mississippi Dept of ITS, Jackson, MS

ABSTRACT

The TRANSLATE function of SAS can be used to strip out punctuation and other unwanted characters resulting in a string of words separated by blanks which can then be compared word by word. This process is generally considered a word processing function and at first might not seem relevant or appropriate to working with JCL, SYSLOGS, CLISTS, or other system related files and records.

Nevertheless, these types of files and records can be visualized as a series of words rather than individual bytes. The third generation language model would be to do a byte by byte comparison, build tables, make comparisons, keep or reject, write the record. Although SAS can be used to do a byte by byte comparison, it also allows us to consider a different solution that in fact might be more intuitive and easy to code.

INTRODUCTION

On a weekly basis we run three JCL JOBS that consist of a series of FDR steps to dump disk packs to tape. The VOL parameter of this JCL contains a string of tape VOLSERS which are added to over time as the disk packs become more saturated with data and more tapes are needed for the dump.

Sometimes re-running a particular step is necessary for the operators or the on-call person. To simplify this process a TSO CLIST was written that can be invoked to build and submit the JCL to back up one disk pack at a time.

Different personnel maintain the FDR backup JCL and the TSO CLIST, so a process was needed to simplify keeping the FDR JCL and the CLIST in sync as VOLSERS were added and removed from the backup JCL JOBS. To do this manually was a time consuming and error prone process.

THE SOLUTION

A solution was to write a SAS program to read the FDR JCL and the TSO CLIST, find the string of VOLSERS in each, sort and match-merge the VOLSERS, then print a report of VOLSERS that appear in the FDR JCL but not in the CLIST. This report is used to update the CLIST so that the strings of VOLSERS in it match those in the FDR JCL.

SAS FEATURES USED

- TRANSLATE FUNCTION
- INDEX FUNCTION
- INPUT /
- $VARYING INFORMAT
- ASSIGNMENT STATEMENT
- PROC SORT
- MERGE
- INPUT @@
- SUBSETTING IF
- FILE
- PUT
- INFILE

OPERATING ENVIRONMENT

MVS/ESA

SAS V6.06

THE PROGRAM

```
  DATA;
    INFILE AUTO;
      INPUT @1 REC $72.@;
      X=INDEX(REC,'SER=(');
      IF X > 0 THEN DO;
      LENVAR = (72 - (X+5));
      INPUT @X+5 VOLS $VARYING72.
     LENVAR
     / @3 BLANK $1.
       @4 VOL2   $69.
     / @3 BLK2    $1.
       @4 VOL3   $69.;
IF  BLANK  NE  ' '  AND  BLK2  NE  ' '
THEN DO;
REC1=
TRANSLATE(VOLS,' '','/,=,(,)');
FILE AUTO2;
  PUT @1   REC1 $80.;
                                    END;
IF BLANK EQ ' ' AND BLK2 NE ' ' THEN
DO;
REC1=
TRANSLATE(VOLS,' ',','/,=,(,)');
REC2=
TRANSLATE(VOLS,' ',','/,=,(,)');
FILE AUTO2;
  PUT @1    REC1 $80.
      @80  REC2 $80.;
                                    END;
IF BLK2 EQ ' ' AND BLANK EQ ' ' THEN
DO;
REC1=
TRANSLATE(VOLS,' ',','/,=,(,)');
REC2=
TRANSLATE(VOL2,' ',','/,=,(,)');
REC3=
TRANSLATE(VOL3,' ',','/,=,(,)');
FILE AUTO2;
  PUT  @1      REC1 $80.
        @80    REC2 $80.
       @160  REC3 $80.;

END;
                        END;
DATA CLIST;
  INFILE CLIST;
    INPUT @1 REC $72.@;
    X=INDEX(REC,'VOL');
    IF X > 0;
    LENVAR = (72 - (X+5));
   input   @X+5   VOLS   $VARYING72.
LENVAR;
REC1=
  TRANSLATE(VOLS,' ',','/,='(,)');
       FILE CLIST2;
        PUT @1 REC1;
  DATA VOL1;
   INFILE AUTO2;
   INPUT VOLSER $ @@;
   AUTO='AUTO';
   PROC SORT;
   BY VOLSER;
  DATA VOL2;
   INFILE CLIST2;
```

```
   INPUT VOLSER $ @@;
   CLIST='CLIST';
PROC SORT;
 BY VOLSER;
DATA MERGE;
 MERGE VOL1 VOL2; BY VOLSER;
 IF CLIST=' ';
 PROC PRINT;
 ID VOLSER;
 VAR CLIST AUTO;
```

PROGRAM DETAILS

INFILE AUTO

Reads a concatenation of three PDS members as in:

```
//auto dd dsn=pds(disk1),disp=shr
//      dd dsn=pds(disk2),disp=shr
//      dd dsn=pds(disk3),disp=shr
```

INPUT @1 REC $72.@;

This statement reads a record for a length of 72 which is the maximum length these records can be because the last eight bytes are line numbers (80-8). The @ holds this record for additional processing.

```
X=INDEX(REC,'SER=(');
IF X > 0 THEN DO;
```

The INDEX function is used to find the strings that start with `ser=(‘. The VOLSERS will follow this JCL parameter. If x is greater than zero, then a string has been found and you want the following calculating to occur. .

LENVAR = (72 - (X+5));

This statement calculates the length of the string of VOLSERS - maximum of 72 minus the value of x + 5. ‘Ser=(‘ is five bytes.

```
  INPUT @X+5 VOLS $VARYING72.
LENVAR
   / @3 BLANK $1.
     @4 VOL2   $69.
   / @3 BLK2    $1.
     @4 VOL3   $69.;
```

The $VARYING*w*. INFORMAT reads variable-length fields of character data. The w specifies the maximum width of a character field for all the records in the raw file. It is used when the length of a character value differs from record to record. LENVAR is a length variable that was calculated previously and must be used in conjunction with the $varying INFORMAT. This variable contains the actual width of the character field in the current record.

The ‘ /’ advances the pointer to the next input line. The BLANK variable is used to determine if more VOLSERS follow; the VOL2 variable is the possible VOLSER string. The next two lines of code are for the same purpose. This would normally take care of the possible VOLSER strings, but if not another series of these lines can be added.

```
  IF BLANK NE ' ' AND BLK2 NE ' '
THEN DO;
  REC1=
  TRANSLATE(VOLS,' ','/,=,(,)');
  FILE AUTO2;
   PUT @1   REC1 $80.;

END;
```

```
  IF BLANK EQ ' ' AND BLK2 NE ' '
 THEN DO;
  REC1=
  TRANSLATE(VOLS,' ','/,=,(,)');
  REC2=
  TRANSLATE(VOLS,' ','/,=,(,)');
  FILE AUTO2;
    PUT @1    REC1 $80.
        @80 REC2 $80.;

END;
  IF BLK2 EQ ' ' AND BLANK EQ ' '
 THEN DO;
   REC1=
   TRANSLATE(VOLS,' ','/,=,(,)');
   REC2=
   TRANSLATE(VOL2,' ','/,=,(,)');
   REC3=
   TRANSLATE(VOL3,' ','/,=,(,)');
    FILE AUTO2;
      PUT  @1      REC1 $80.
           @80     REC2 $80.
          @160   REC3 $80.;

END;
                    END;
```

For an explanation of the previous lines of code, let's examine the following JCL statements:

```
//TAPE1 DD DSN=B.DSK876,
// UNIT=TAPE,DISP=(,CATLG),
// VOL=(,,,9),SER=(DS8761,
//   DS8763,DS8764))
//SYSPRIN2 DD SYSOUT=*
//DISK2 DD VOL=SER=DSK877
//TAPE2 DD DSN=B.DSK877,
// UNIT=TAPE,DISP=(,CATLG),
// VOL=(,,,25,SER=(DS8771,
//    DS8772,DS8773,DS8774,
//    DS8775,DS8776))
//SYSIN DD DSN=PDS1(FDROPT)
```

After the initial VOLSER string is located using the INDEX function to determine where the character string 'ser=(' begins , it is then necessary to know how many additional strings of VOLSERS follow. The BLANK and BLK2 variables are used for this check. If the third column of the next line is blank and the third column of the next line is not blank, then there is only one additional line. If the third column of both lines is blank then there two additional lines. Additional checks can be added for more strings of VOLSERS if needed.

The TRANSLATE function is used to substitue a blank for the following characters: / , =( ). This in effect puts a space between each VOLSER making the record a string of words which allows the use of the INPUT @@ statement later.

FILE and PUT statements are used to write these strings of volsers to a temporary data set.

The DO statements have their respective END statements.

```
  DATA CLIST;
    INFILE CLIST;
      INPUT @1 REC $72.@;
       X=INDEX(REC,'VOL');
       IF X > 0;
    LENVAR = (72 - (X+5));
   INPUT @X+5 VOLS $VARYING72.
LENVAR;
  REC1=
  TRANSLATE(VOLS,' ','/,='(,)');
        FILE CLIST2;
         PUT @1 REC1;
```

To follow the preceding lines of code, we need to examine the following CLIST fragment:

```
Else if &pack=dsk877 then do
Set vol1=ds8771,ds8772,ds8773
Set vol2=ds8774,ds8775,ds8776
Set vol3=ds8777,ds8778,ds8779
Set vol4=ds877a,ds877b,ds877c
Goto four
End
```

INFILE CLIST reads a TSO CLIST from a PDS as in:

```
//clist dd dsn=clists(backup),disp=shr
```

A record for a maximum of 72 characters is held for additional processing by using the trailing @. The INDEX function is used to find the string that starts with VOL. If this string is found, x will be a number greater than 0. A length is calculated and stored in variable LENVAR.

The TRANSLATE function is used to strip out the following: , / = ( );

The record is then written a temporary file referenced by CLIST2.

```
DATA VOL1;
 INFILE AUTO2;
  AUTO='AUTO';
  INPUT VOLSER $ @@;
  PROC SORT; BY VOLSER;

DATA VOL2;
 INFILE CLIST2;
   INPUT VOLSER $ @@;
    CLIST='CLIST';
   PROC SORT; BY VOLSER;
```

The preceding two SAS DATA steps create SAS datasets which consists of the VOLSERS from the FDR jobs and the CLIST. ASSIGNMENT statements are used to create a field to indicate whether the VOLSERS came from the CLIST or the FDR JCL . Note the use of the INPUT statement with the @@. This is useful when each INPUT line contains values for several observations.

```
DATA MERGE;
 MERGE VOL1 VOL2; BY VOLSER;
 IF CLIST=' ';
 PROC PRINT;
 ID VOLSER;
 VAR CLIST AUTO;
```

The MERGE DATA step does a merge by VOLSER of the two previous DATA steps. If the CLIST variable is blank or missing, then that indicates the VOLSER is in one of the FDR jobs but not in the CLIST. A report of the VOLSERS not in the CLIST are printed. If there are none, then there is no report.

CONCLUSION

Many times the data that programmers have to code for does not exist in neatly defined and delineated locations, but the flexibility and power of SAS allows one to find, compare and manipulate strings of data with relative ease and with a minimum amount of code.

DEFINITIONS

CLIST - Command List. It is a high level interpretive language that enables one to work more efficiently with TSO.

JCL - Job Control Language. It is a series of control statements that provide the means of communication between an application program and the operating system and computer hardware.

JOB - A JOB is the basic independent unit of work.

SYSLOG - System Log. This is a record of what has transpired on the system.

TSO - Time Sharing Option. It allows users to interactively share computer time and resources.

ACKNOWLEDGMENTS

This author thanks Russell Ferguson, Deputy Director of Mississippi Dept of ITS for reading this paper and making suggestions.

Randall M Nichols
301 N Lamar St Suite 508
Jackson MS 39201
601-359-2642
nichols@its.state.ms.us