Paper 86-26

YOU CAN LOOK IT UP: A Robust Macro Using PROC FORMAT with CNTLIN

Myra A. Oltsik, Direct Response Corporation, White Plains, NY

ABSTRACT

The CNTLIN= option in PROC FORMAT lets the programmer create a format from one dataset using one identifying field/ variable, and use that format to search for records in another dataset. This is an especially helpful option when extracting large numbers of specific records from flat files or unindexed datasets. This paper presents a Macro program which includes all the steps necessary to produce such a format. Even if you don't know much about macro programming, you can use this Macro to create CNTLIN Formats for yourself.

Introduction

The programs I write require a number of specific records from a large file or dataset which is not indexed. To do so, I created a macro robust enough to handle character variables of any length.

My Data

Most of the data of the insurance company I work for are placed in VSAM files on an MVS main frame system. I often need to extract data based on ID number or vehicle identification number (VIN). The advantage of using VSAM files is that they are indexed to speed up searches. While most of the VSAM files are indexed on ID number, depending on with which data they are associated, not all files are indexed as such. (The prime key index is nearly always a date-time stamp.) None of the files are indexed on VIN. In addition, some files are "refreshed" monthly. The monthly backup generations are saved in flat files, and not indexed at all. While the company has only been selling for a few years, we already have main databases of about 300,000 of one type and 40,000 of another. In addition, I have had to match our data with data from other sources, for example, a tape of all 8 million cars registered in New York. Using PROC CNTLIN formats makes look ups in these files much faster and more efficient.

The Macro

```
/*
   PROGRAM:        MCNTLIN
   AUTHOR:         Myra A. Oltsik
   ORIGINAL DATE:  10/25/99
   PURPOSE:        AUTOMATICALLY CREATE 'CNTLIN FORMAT' TO FIND
                   RECORDS IN AN UNINDEXED FILE OR DATASET,
                   ALLOWING FOR AN EMPTY INPUT DATASET TO THE MACRO.
   LAST CHANGE:    06/22/00: ADD PARAMENTS SO THAT ANY CHARACTER
                             FIELD CAN CREATE A 'CNTLIN FORMAT'.
                   08/15/00: ADD SORT TO HAVE UNIQUE ID VARIABLE
                             DIRECTLY IN THE MACRO.
*/
```

```
%MACRO MCNTLIN(FORMT,DSN,ID);

   /**************************************************************/
   /* FORMT - THE NAME OF THE FORMAT BEING CREATED.              */
   /* DSN   - THE INPUT DATASET.                                 */
   /* ID    - THE FIELD/VARIABLE FROM WHICH TO CREATE THE FORMAT. */
   /**************************************************************/

   /**************************************************************/
   /* PART 1                                                     */
   /* SORT THE INPUT DATASET ON THE ID USING 'NODUPKEY' OPTION   */
   /* AS THE CNTLIN OPTION CANNOT HAVE DUPLICATE KEYS.           */
   /**************************************************************/

   PROC SORT DATA=&DSN OUT=UNIQID(KEEP=&ID) NODUPKEY;
      BY &ID;
   RUN;

   /**************************************************************/
   /* PART 2                                                     */
   /* FIND THE LENGTH OF THE ID FIELD USING PROC CONTENTS AND    */
   /* CREATE A MACRO VARIABLE WITH THAT LENGTH IN IT.            */
   /**************************************************************/

   PROC CONTENTS DATA=UNIQID OUT=CNTNT NOPRINT;
   RUN;

   DATA _NULL_;
      SET CNTNT;
      IF NAME = UPCASE("&ID");
      CALL SYMPUT('PLEN',TRIM(LEFT(PUT(LENGTH,8.))));
      STOP;
   RUN;

   DATA &FORMT;

      LABEL   = "KEEP";
      FMTNAME = "$&FORMT";
      FORMAT START $CHAR&PLEN..;

      /***********************************************************/
      /* PART 3                                                  */
      /* CREATE A 'DUMMY' FORMAT THE SAME LENGTH AS THE ID FIELD */
      /* WHEN THE INPUT DATASET IS EMPTY.                        */
      /***********************************************************/

      IF RECS EQ 0 THEN DO;
         START = REPEAT('X',&PLEN - 1);
         OUTPUT ;
      END;

      SET UNIQID NOBS=RECS;
      START = &ID;
      OUTPUT ;
   RUN;
```

```
         PROC FORMAT CNTLIN = &FORMT;
         RUN;

         /*****************************************************************/
         /* PART 4                                                        */
         /* DELETE THE CONTENTS AND UNIQUE DATASETS.                      */
         /*****************************************************************/

         PROC DATASETS LIB=WORK NOLIST;
            DELETE CNTNT UNIQID;
         RUN; QUIT;

      %MEND MCNTLIN;
```

The input fields for the macro are FORMT, the name of the format being created; DSN, the name of the input dataset; and, ID, the name of the field or variable from which to create the format, *i.e.*, the field you want to search in another dataset. The names you input are then substituted for the macro variables in %MCNTLIN.

Part 1

The data in the format *must* be unique. The input dataset, however, may have duplicates. Rather than remembering to create a dataset of unique IDs, %MCNTLIN does it for you.

```
PROC SORT DATA=&DSN
      OUT=UNIQID(KEEP=&ID) NODUPKEY;
   BY &ID;
RUN;
```

The output dataset UNIQID is just a list of all the IDs for which you want to search.

Part 2

SAS® has many ways to find the length of a variable. This macro uses PROC CONTENTS.

```
PROC CONTENTS DATA=UNIQID OUT=CNTNT
   NOPRINT;
RUN;
```

```
DATA _NULL_;
   SET CNTNT;
   IF NAME = UPCASE("&ID");
   CALL SYMPUT('PLEN',
      TRIM(LEFT(PUT(LENGTH,8.))));
   STOP;
RUN;
```

The macro variable PLEN then takes on the length of the ID field.

While it may be faster to use the SASHELP.VCOLUMN file in this case, the macro would then need the LIBNAME of the datasets as a *separate* input field. With PROC CONTENTS, you can use databases either in the WORK library or in any specified library.

Part 3

```
DATA &FORMT;

   LABEL   = "KEEP";
   FMTNAME = "$&FORMT";
   FORMAT START $CHAR&PLEN..;

   IF RECS EQ 0 THEN DO;
      START = REPEAT('X',&PLEN - 1);
      OUTPUT ;
   END;

   SET UNIQID NOBS=RECS;
   START = &ID;
   OUTPUT ;
RUN;
```

```
PROC FORMAT CNTLIN = &FORMT;
RUN;
```

I have an MVS job in production which runs every night, obtaining a different number of records each time. There were instances when the first data step created a dataset without any records in it. When that happened, the macro failed, and the rest of the production bombed. I needed a way to keep the program running even with null input datasets. (I owe a great debt to Ian Whitlock who first helped me with this.)

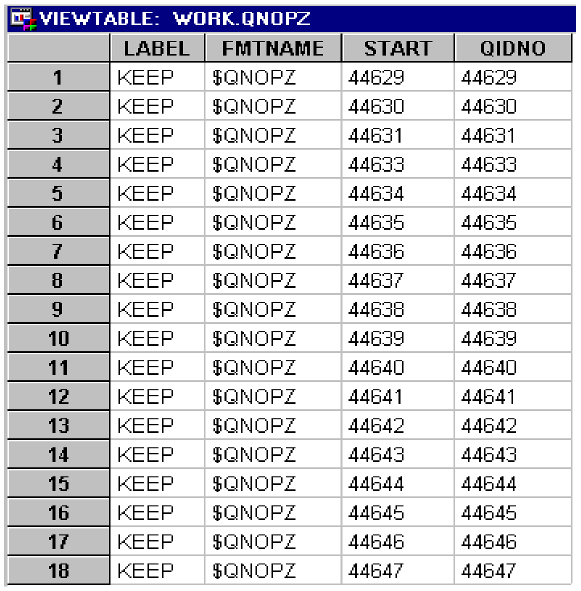
If the input dataset is null – it has no records – the format has only one value that actually doesn't exist in the data file to be searched. It is a string of X's to fill the field to its full length. In this case, the REPEAT() function creates that dummy value. As long as the input dataset isn't empty, the values in that dataset become format values. (For a fuller explanation of the options and statements needed for PROC FORMAT with CNTLIN, see SUGI 23, Coder's Corner, Paper 68, *IN and OUT of CNTL with PROC FORMAT*, by Nancy Patton.)

Part 4

The macro ends by cleaning up the output datasets created by PROC CONTENTS and PROC SORT.

Running The Macro

Once the macro has been created, the two lines underlined in the code below are all that needs to be added to your code. The first passes the values you want to the macro, and the second "keeps" only those records needed for your output dataset.


VIEWTABLE: WORK.QNOPZ

	LABEL	FMTNAME	START	QIDNO
1	KEEP	$QNOPZ	44629	44629
2	KEEP	$QNOPZ	44630	44630
3	KEEP	$QNOPZ	44631	44631
4	KEEP	$QNOPZ	44633	44633
5	KEEP	$QNOPZ	44634	44634
6	KEEP	$QNOPZ	44635	44635
7	KEEP	$QNOPZ	44636	44636
8	KEEP	$QNOPZ	44637	44637
9	KEEP	$QNOPZ	44638	44638
10	KEEP	$QNOPZ	44639	44639
11	KEEP	$QNOPZ	44640	44640
12	KEEP	$QNOPZ	44641	44641
13	KEEP	$QNOPZ	44642	44642
14	KEEP	$QNOPZ	44643	44643
15	KEEP	$QNOPZ	44644	44644
16	KEEP	$QNOPZ	44645	44645
17	KEEP	$QNOPZ	44646	44646
18	KEEP	$QNOPZ	44647	44647

Example of the table the macro creates

```
%MCNTLIN(QNOPZ,QNOP,QIDNO);

DATA INDEXQ;
   INFILE D0152001 END=TRAIL;
   INPUT
      @  6 QIDNO      $CHAR8.
      @;
   IF TRAIL = 1 THEN DELETE;
   IF PUT(QIDNO,$QNOPZ.)='KEEP';
   INPUT
      @ 14 (PK1-PK30) (PD10.)
   ;
   FORMAT PK1-PK30 20.;
RUN;
```

CONCULSIONS

The macro presented in this paper allows you to extract a large number of specific records from an even larger dataset without the latter dataset being indexed. It is robust enough to handle character variables of any length.

Myra A. Oltsik, Data Specialist
Direct Response Corp.
4 Gannett Drive, Suite 2
White Plains, New York 10604-3408
(914)640-6530
myra.oltsik@responseinsurance.com