

SYMPUT Routine: Painlessly Putting Parts and Pieces Together

Imelda C. Go, Richland County School District One, Columbia, SC

ABSTRACT

The macro SYMPUT routine was useful in consolidating the same type of information from different sources. This particular use of the SYMPUT routine evolved from the need to combine the data (in the form of 50 different files) from 50 different schools in a large school district. Each school has a record in an external text file that contains, as fields, all values of parameters that are predictably needed for processing the data file from each school. The SYMPUT routine was used to access specific macro parameter values (from the external text file) for each school. This eliminated the need to explicitly type/retype these specific values in SAS® program code for each set of files that needed to be consolidated.

INTRODUCTION

We have several files (with the same layout) that need to be combined. If it is feasible, one might choose to combine all raw data files into one file before any processing. If the source of the data file is important, it would be good if each file had a field which identifies the source of the records. If it is not important, then the processing is even simpler. In this paper, it is important to identify the source of each data file.

The approach will be to process each individual file first before the final consolidation. This was realistic in our case because the only way to identify the source of each file was by its file name. There was unfortunately no field in the data file that consistently identified the source of the file. The following abbreviated example illustrates the SYMPUT routine application using only 12 (instead of the original 50) schools.

DATA FILE WITH MACRO PARAMETERS

The first step was to create a data file that contains, as fields, all parameters that might be useful for processing each file. The following table shows a sample layout:

Columns	Description
1 - 2	index
3 - 4	school level (E=elementary, M=middle, H=high)
5 - 29	school name
30 - 36	unique numeric ID for schools (BEDS code)
37 - 44	unique character ID for schools

This data file for 12 schools would look like:

1 E ARDEN ELEM.	4001020ARDEN
2 E BRADLEY ELEM.	4001024BRADLEY
3 E BRENNEN ELEM.	4001025BRENNEN
4 E BURNSIDE ELEM.	4001027BURNSID
5 E A. C. MOORE ELEM.	4001048ACMOORE
6 M ALCORN MIDDLE	4001019ALCORN
7 M W. A. PERRY MIDDLE	4001067WAPERRY
8 M SANDERS MIDDLE	4001034SANDERS
9 H EAU CLAIRE HIGH	4001005EAUCLAIR
10H A. C. FLORA HIGH	4001007ACFLORA
11H C. A. JOHNSON HIGH	4001011CAJOHNS
12H KEENAN HIGH	4001012KEENAN

CREATING INDIVIDUAL DATA SETS

The next step is to create the individual SAS data sets that will be combined later. Using an iterative loop in a macro, we can read all 12 files and create their corresponding data set.

SEGMENT	PROGRAM CODE
1	%LET PERIOD=C979845; %LET EXT=45; LIBNAME DT "T:\C&PERIOD";
2	%MACRO READDATA;
3	%DO INDEX=1 %TO 12; DATA TEMP;
4	INFILE "T:\&PERIOD.\SCHOOLS.CON" PAD; INPUT INDEX 2. LEVEL \$ 2. SCHOOL \$ 25. BEDSCODE \$ 7. CODE \$ 7.;
5	IF INDEX=&INDEX; CALL SYMPUT ('CODE',CODE); CALL SYMPUT ('INDEX',INDEX); CALL SYMPUT ('LEVEL',LEVEL); CALL SYMPUT ('SCHOOL',SCHOOL); CALL SYMPUT ('BEDSCODE',BEDSCODE);
6	
7	DATA DT.&CODE; INFILE "T:\C&PERIOD.\C&BEDSCODE..&EXT" LRECL=343 PAD;
8	***SAMPLE INPUT STATEMENT FOLLOWS: INPUT SSN 1-9 GRADE 10-11 NAME \$ 13-40;
9	INDEX=&INDEX; LEVEL="&LEVEL"; LENGTH SCHOOL \$ 25.;
10	SCHOOL="&SCHOOL"; CODE="&CODE"; ***YOU CAN INSERT OTHER PROCESSING STATEMENTS IN THIS SEGMENT;
11	%END; %MEND READDATA; %READDATA;

Segment 1. These are some macro variable specifications that are useful in processing this particular type of twelve raw data files. This set of 12 files is what our office refers to as the 45th-day consolidated data files for the 1997-98 school year (i.e., C979845).

Segment 2. This is the beginning of the iterative macro.

Segment 3. The macro variable INDEX iterates from 1 till 12. Each iteration corresponds to the 1st through 12th schools.

Segment 4. The input statement reads the external data file with macro parameters. In this case, the external file is T:\C979845\SCHOOLS.CON.

Segment 5. For each iteration, the only observation that is kept is the record that has the same index value as that of the current value of the iterating variable INDEX. When INDEX=1, the only record kept for that data step is the record for Arden Elem.

Segment 6. The SYMPUT routine is used to assign the values of the variables in the data set (created by segments 4 and 5) to some macro variables. When INDEX=1, the following values are assigned to macro variables:

Macro Variable	Assigned Value
code	ARDEN
index	1
level	E
school	ARDEN ELEM.
bedscode	4001020

Segment 7. This uses the macro variables from segment 6 to process the file for one school. When INDEX=1, the data set ARDEN is created by reading the file named C4001020.45. The C refers to the consolidated data files. The 4001020 is the unique numeric ID for Arden Elem. The 45 refers to the 45th day of the school year.

Segment 8. This segment would contain statements that are specifically for the files you are dealing with.

Segment 9. This uses the macro variables (from segment 6) to assign values to ordinary SAS variables. When macro variable INDEX=1, data set ARDEN will have ordinary SAS variables with the following values: INDEX=1, LEVEL="E" SCHOOL="ARDEN ELEM.", and CODE="ARDEN".

Segment 10. This segment would contain statements that are specifically for the files you are dealing with.

Segment 11. This contains the end of the macro and the statement that calls for the execution of the macro.

The idea is to use the SYMPUT routine to supply the macro variable values required to process the files. This eliminates the need to explicitly type long lists of statements that call macro routines. For example, if we did not use INDEX to iterate through the 12 schools, we would need 12 macro statements to execute macro READDATTA that would involve various parameters. We would get a series of 12 statements such as:

```
%readdata(ARDEN,ARDEN ELEM.,4001020,E,1)
%readdata(BRADLEY,BRADLEY ELEM.,4001024,E,2)
%readdata(BRENNEN,BRENNEN ELEM.,4001025,E,3)
%readdata(BURNSID,BURNSIDE ELEM.,4001027,E,4)
%readdata(ACMOORE,A. C. MOORE ELEM.,4001048,E,5)
%readdata(ALCORN,ALCORN MIDDLE,4001019,M,6)
%readdata(WAPERRY,W. A. PERRY MIDDLE,4001067,M,7)
%readdata(SANDERS,SANDERS MIDDLE,4001034,M,8)
%readdata(EAUCALAI,EAU CLAIRE HIGH,4001005,H,9)
%readdata(ACFLORA,A. C. FLORA HIGH,4001007,H,10)
%readdata(CAJOHNS,C. A. JOHNSON HIGH,4001011,H,11)
%readdata(KEENAN,KEENAN HIGH,4001012,H,12)
```

This series of statements is not very readable as the number of statements gets larger and as we deal with more and more macro parameters. One advantage of using an external data file with the macro parameter values is that it can be used in other similar programs that can use at least some of the fields in this file as macro parameter values. It is very convenient when a change has to be made to the data file with macro parameter values. You do not have to revise several programs to make changes since your various programs use the same external file.

Suppose one of the 12 data files was not available. The macro should still go through all 12 iterations in spite of a missing data file. The SAS log will put an error message that the file cannot be found and create an empty data set that corresponds to the missing data file. This data set is, of course, accessible even if it has no records. In reality,

you may need to collect the data files over time, and it takes a while before you complete the set of files. Hence, you may need to consolidate a partial set of files while you wait for the rest of the files. Empty data sets for missing files are to your advantage because you do not need to change any SAS code when you combine the different data sets. You may have a SAS statement that uses all 12 data sets and it will still execute even if some of the data sets are empty.

The INDEX variable also served another purpose. Its increasing numeric order reflects the order in which school reports were required to appear on paper. There are times when having such a simple variable is more convenient than making sure another variable or a combination of variables can produce the desired sort order.

RELATED IDEAS

- Now that you've created all the SAS data sets that you need, it is a simple matter to combine them into a single master data set.
- You can write a PROC DATASETS statement to delete all the small data sets once you have consolidated them.
- If you ran the macro in spite of not having all the 12 data files, you can write some statements to tell you which ones are missing. You can use the external data file to check against the members of the incomplete consolidated data set.
- You can add two macro parameters to the macro readdata and revise the macro DO statement to iterate based on the values supplied for its two macro parameters. For example, if the desire is to process the files for the 3rd through 7th schools, then the command of %readdata(3,7) would be used.

CONCLUSION

This paper proposes the use of the SYMPUT routine to handle the task of consolidating different files from various sources. Its accompanying macro routine can be tailored to suit your computing needs and can be revised to deal with other sets of files that can be consolidated by using the same basic strategy.

The paper illustrates an application of the SYMPUT routine. It does not claim that the illustrations provide the most ideal or only way to deal with similar situations.

REFERENCES

SAS Institute Inc. (1997), *SAS Macro Language Reference, First Edition*, Cary, NC: SAS Institute Inc.

TRADEMARK NOTICE

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Imelda C. Go
Office of Research and Evaluation
Richland County School District One
1616 Richland St.
Columbia, SC 29201

Tel.: (803) 733-6079
Fax: (803) 929-3873
icgo@juno.com