Paper 057-29

A Macro for Reading Multiple Text Files

Debbie Miller, National Park Service, Denver, CO

ABSTRACT

Most of us have to exchange data at some time or another with others who do not use the SAS® System, and therefore cannot use or send SAS data sets. Importing data files that are not in SAS format into SAS and compiling them for processing can be a time-consuming process. This is particularly so if the files are numerous and are not identical in format. SAS has many powerful tools that can help you to automate this process. This paper presents one simple solution for reading multiple plain text files that have differing numbers of variables using base SAS and the macro facility. This paper is intended for users of all skill levels.

INTRODUCTION

Oftentimes we are handed data in a collection of external files and asked to bring the data into SAS, where we need to assemble the individual files into a single data set for further processing. When the files do not all contain the same variables in the same order, it is a challenge to import them without having to process each file individually. This paper presents one simple solution that uses basic macro coding to read multiple plain text files that do not share a common data order.

PROBLEM: IMPORTING TEXT FILES WITH DIFFERENT VARIABLES

As a data analyst I exchange data with many external agencies that do not use SAS and thus need to send or receive data in non-SAS formats. Not long ago I received a set of over 300 text files containing meteorological data from approximately 60 individual stations located at park units across the country. Each file contains one year's worth of data at a particular station. These data in these files needed to be read into SAS and compiled so that summary statistics could be computed. This presented a problem, because not all of the available parameters are measured at every site, and some sites' sets of measurements have changed over time. As a result the files do not all contain the same variables, and the order of the variables in the files is not constant. Not wanting to type an individual INPUT statement specific to each external text file, I needed to automate the process of reading the files.

Below are samples from two text files that illustrate the problem:

File 1:

DATE	TIME	SWS	VWS	VWD	**SDWD**	TMP	**DTP**	**RH**	**RNF**	**WET**	SOL	STP
980101	0000	4.8	4.8	80	7	3.2	0.9	69	0	1	0	25.6
980101	0100	4.6	4.6	77	8	2.9	0.9	70	0	1	0	25.9
980101	0200	4.5	4.5	78	8	2.6	0.8	70	0	1	0	25.7
980101	0300	4.8	4.7	77	8	2.3	0.8	71	0	1	0	25.6

File 2:

DATE	TIME	SWS	VWS	VWD	TMP	**DPT**	SOL	STP
940101	0000	3.6	3.5	257	-4.6	-7.1	0	24.8
940101	0100	3.4	3.2	269	-4	-7.2	0	24.4
940101	0200	4.5	4.3	245	-3.6	-7.2	0	24.2
940101	0300	3.7	3.6	255	-4.3	-7.3	0	23.9

File 1 contains five variables that are not found in file 2 (SDWD, DTP, RH, RNF, WET). In addition, file 2 contains one variable not found in file 1 (DPT). This means that only the first five variables are found in both files in the same order, making it impossible to write a single INPUT statement that will read both files. Many of the other 300+ files have still other combinations of variables than those shown here. In order to be able to combine these files, I need a program that can accommodate each file's unique combination of variables.

IMPORTING TEXT FILES WITH A MACRO

The first step in the program is to get the names of all of the text files. The program uses these names to read them one at a time. When reading the data from each file, the program first retrieves the names of the variables in the file, and then it uses the names to read the lines of data. Each of these steps is described below.

RETRIEVING THE FILE NAMES WITH A PIPE

The program first obtains the names of the files to be read in and determines how many there are. It brings the names of the data files into a SAS data set by means of a pipe. This technique is described in the SAS Technical Document TS-581, which is available on the SAS Institute web site. A pipe is a very useful device that allows two processes to communicate.

There are two different kinds of pipes, named and unnamed. A named pipe permits two-way communication, while an unnamed pipe can be used only for one-way communication. In this program I use an unnamed pipe to issue commands from SAS to the operating system, which in this case is Windows 2000. The code below, which has been adapted from an example in TS-581, shows how this is done.

The option PIPE is specified in a FILENAME statement followed by the operating system command in quotes. This operating system command produces a file listing under Windows. The filename statement is then referenced in an INFILE statement in a DATA step. This causes the names of the files to be read from the operating system directory into the SAS data set FILE_LIST. The CALL SYMPUT statement stores the current observation number from the data set FILE_LIST to the macro variable 'num_files'. The final value of the macro variable 'num_files' is the number of file names contained in the data set FILE_LIST.

```
filename indata pipe 'dir d:\requests\miller\testfiles /b';

data file_list;
  length fname $20;
  infile indata truncover;  /* infile statement for file names */
  input fname $20.;  /* read the file names from the directory */
  call symput ('num_files',_n_); /* store the record number in a macro variable */
run;
```

RETRIEVING THE VARIABLE NAMES

Once this data set has been created, the macro *fileread* reads each file in turn. This is necessary because the individual files contain different combinations of variables. The macro uses a do-loop with an index that runs from 1 to the number of files to be read. The DATA _NULL_ step is used in each iteration to read the name of the next file to be read from the FILE_LIST data set and store it in the macro variable 'filein'.

After the name of the next file to be read has been retrieved, the names of the variables are read from the first line of the text file. In this macro, it is necessary to know the maximum number of variables possible in any text file. Here the maximum number is 17. The variable names are then stored as variables X1-X17 in the data set VAR_NAMES. Since at this point I only want to read the names of the variables, which are located in the first line of the input text file, I specify the option OBS=1 in the INFILE statement. This option tells SAS the number of the last record in the input file that you want to read. The MISSOVER option prevents SAS from automatically going to the next line if it doesn't find values for all of the variables. If fewer than 17 variables are read, the remaining variables are set to missing. The resulting data set contains only 1 record, with 17 text variables that contain the names of the variables in the input file that will be read next.

```
%macro fileread;

%do j=1 %to &num_files;

data _null_;
  set file_list;
  if _n_=&j;
  call symput ('filein',fname);
run;

data var_names;
  length x1-x17 $12;
  infile "d:\requests\miller\testfiles\&filein" obs=1 missover;
  input (x1-x17) ($) ;
run;
```

A new macro called *varnames* that is nested inside the main macro, *fileread*, is then used to store the variable names to macro variables. The names are stored in macro variables V1 through V17. This is done by using a DATA _NULL_ step inside a do-loop that executes 17 times. A CALL SYMPUT statement is used to store the names to the macro variables. The GLOBAL statement is necessary so that the 17 macro variables will be available for use outside of the *varnames* macro.

```
%macro varnames;

%do i=1 %to 17;

%global v&i;
```

```
data _null_;
  set var_names;
  call symput("v&i",trim(x&i));
run;

%end;
%mend varnames;

%varnames;
```

READING THE DATA LINES

The next step in the macro *fileread* is to read in the data. Each input text file is read using the code below. The data are read each time into the SAS data set called TEMP, which is overwritten on each iteration of the DO-loop. The INFILE statement option FIRSTOBS=2 tells SAS to begin reading at the second line of the file (since the first line contains the variable names). The option MISSOVER is used here also to prevent SAS from reading from the next line in the event that it does not find all 17 variables. The macro variables &v1 through &v17, which contain the names of the variables, are used in the input statement. This ensures that each variable is assigned the correct name in the SAS data set, regardless of the order in which it appears in the input file. Since the type of each variable isn't known in advance, all variables are initially read in as type character; they can be changed later once the individual files have been grouped together.

```
/* read the data lines into a temporary file */

data temp;
  infile "d:\requests\miller\testfiles\&filein" firstobs=2 missover;
  input (&v1 &v2 &v3 &v4 &v5 &v6 &v7 &v8 &v9 &v10 &v11 &v12 &v13 &v14 &v15 &v16 &v17)
($);
run;
```

Finally, the individual data sets need to be grouped together into one large SAS data set. This is accomplished using a macro %IF statement. When the first text file is read, the index variable 'j' will be equal to 1. If this is the case, then the first DATA step is executed and the data set DATA_ALL is created from the temporary data set. If the index variable 'j' is greater than one, then the data set DATA_ALL already exists, and the second DATA step is executed, adding the temporary data set to the DATA_ALL data set.

```
/* assemble the individual files */

%if &j=1 %then %do;
  data data_all;
    set temp;
  run;
%end;

%else %do;
  data data_all;
    set data_all
        temp;
  run;
%end;


%end;  /* end of do-loop with index j */

%mend fileread;
%fileread;
```

CONCLUSION

Getting data into SAS from other applications for processing is a necessary but sometimes time-consuming process. SAS has many tools available that can help you quickly bring data from external files into SAS data sets. The addition of relatively simple macro routines increases the flexibility and power of the your programs so that you can quickly finish the job of importing and get down to the business of using your data.

REFERENCES

SAS Institute Inc. (1999), *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999), *SAS Macro Language: Reference, Version 8,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1998), *Technical Support Document TS-581*: *Using FILEVAR= to Read Multiple External Files in a DATA Step*, Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Debbie Miller
National Park Service, Air Resources Division
12795 W. Alameda Parkway
Lakewood, CO 80129
Work Phone: (303) 987-6947
Fax: (303) 969-2822
Email: debbie_c_miller@nps.gov