

Using FILEVAR= to read multiple external files in a DATA Step

Reading the filenames from DATALINES

The FILEVAR= option for the INFILE statement provides a simple means of reading multiple external files in a single DATA Step. There are several ways of specifying the list of files that are to be processed. The first is through the use of DATALINES, as in the example below.

```
data one;
infile datalines;
length fil2read $40;
input fil2read $;
infile dummy filevar=fil2read end=done;
do while(not done);
  input var1 var2 var3;
  output;
end;
datalines;
/user/data/input_file_01
/user/data/input_file_02
/user/data/input_file_03
/user/data/input_file_04
;
```

The name of an external file is read from the DATALINES on the first infile statement and stored in a variable for later use. In this example the name of the external file is stored in the variable "fil2read". On each iteration of the DATA Step, a new value of "fil2read" is read from the DATALINES. The FILEVAR= option uses this variable which contains the name of the external file. When the second INFILE statement with the FILEVAR= option is executed, the external file is opened.

In the second INFILE statement, the fileref listed is a dummy argument or place-holder. It does not refer to a fileref assigned by a FILENAME statement. It will appear in the SAS Log for each external file processed.

It is necessary to read the data with a DO loop. Without a DO loop, only one record is read from each external file. Because the DATA Step stops if an INPUT statement attempts to read beyond the end of the file, you must stop inputting records from the external file after encountering the last record. The variable "done", defined by the END= option of the INFILE statement, is set to 1 when the last record is read from the external file. When the value of "done" is 1, the DO loop stops looping and control passes to the next statement following the DO loop.

The DO loop usually takes the form of DO WHILE or DO UNTIL. The choice is up to the programmer. Understanding the differences in the way the two DO loops work can help you avoid some potential problems. The DO WHILE evaluates the test condition at the top of the loop, whereas the DO UNTIL evaluates the test at the bottom of the loop. This means that the DO UNTIL always goes through the code within the loop at least once, but the DO WHILE may not. This means if any of the files in the list is empty, using DO UNTIL will cause the DATA Step to terminate prematurely. Using DO WHILE will not.

If there are no more records in the external file you are processing, the SAS System detects this and sets the END= variable to 1. Using the DO WHILE form, the condition is evaluated and found to be false so the loop does not execute. Using the DO UNTIL form the statements in the loop execute before the condition is tested. The INPUT statement in the loop attempts to read beyond the end of the file, and the DATA Step stops.

NOTE: This paper will show UNIX syntax for external filenames. Host specific differences will be noted where appropriate.

In this example the first INFILE statement points to the DATALINES contained in the program. On each iteration of the DATA Step, a new value of "fil2read" is read from the DATALINES. The second INFILE statement opens the file whose name is retrieved from the DATALINES. The DO WHILE reads the data and writes it to the SAS Data Set.

Reading the filenames from a SAS Data Set

If you have the filenames in a SAS Data Set, you can SET the Data Set instead of reading from the DATALINES. In the following example we have a SAS Data Set called "two" with some number of observations and a variable "fil2read" containing the names of the external files to be read.

```
data one;
set two;
infile dummy filevar=fil2read end=done;
do while(not done);
  input a b c;
  output;
end;
run;
```

Reading all the files from a directory using a Pipe

If you want to read all the files in a directory, first you need to get the list of files in the directory. With UNIX, OS/2, WIN-NT, and WIN95, use an operating system PIPE to obtain a list of files in the directory in a form that the SAS System uses in an INFILE/INPUT. This is done through the FILENAME statement using the PIPE keyword.

When a command is entered at the screen prompt, the operating system processes the command and the information generated is written to your screen. The screen is also known as standard output (STDOUT). When the PIPE keyword is used in a FILENAME statement, the command inside the quotes is passed to the operating system where it is processed and the results are returned to the DATA Step through the PIPE. The data is held in buffers until the INPUT statement reads it.

NOTE: PIPEs are not available on the DOS, Windows 3.1, or mainframe operating systems. Those systems will be addressed later in this paper.

NOTE: You can test the validity of the command that you issue to the operating system, by issuing the command from the command line and viewing the results.

Under UNIX, the command to generate the list looks like this:

```
filename indata pipe 'ls -l /dept/tsd/dataread/files';
```

On the PC operating systems it looks like this:

```
filename indata pipe 'dir C:\dept\tsd\dataread\files /b';
```

NOTE: For further information on the operating system commands above, see the documentation for your operating system.

The information returned from either of these commands is a list of filenames contained in the directory specified. This presents a problem because the INFILE statement requires a fully qualified path and filename. Therefore you must add that information to the "fil2read" variable. See below:

```
filename indata pipe 'ls -l /dept/tsd/dataread/files';

data one;
length fil2read $40 lname $20 fname $20 state $2;
infile indata trunccover;
input f2r $20.;
fil2read='/dept/tsd/dataread/files/'||f2r;
infile dummy filevar=fil2read end=done;
do while(not done);
  input lname : $20. fname : $20. state $ phone;
  output;
end;
run;
```

On the PC platforms use the DOS form of the path and change the assignment of the variable "fil2read" like this:

```
fil2read='C:\dept\tsd\dataread\files\'||f2r;
```

If you want a subset of files in a directory based on character matching in the name, you can change the FILENAME statement like this:

```
filename indata pipe 'ls -l /dept/tsd/dataread/files/inf06*';
```

This is what the UNIX operating system returns:

```
/dept/tsd/dataread/files/inf06.chem.dat
/dept/tsd/dataread/files/inf06.english.dat
/dept/tsd/dataread/files/inf06.history.dat
/dept/tsd/dataread/files/inf06.math.dat
/dept/tsd/dataread/files/inf06.phys_ed.dat
```

On the UNIX platform, this returns the same path that is specified in the command and all the filenames that satisfy the criteria of the search. This behavior is unique to the UNIX environment. Referring to the previous DATA Step example, the variable "f2r" can be used as the FILEVAR= variable instead of "fil2read".

On the PC platforms, you will need to concatenate the drive and path onto the FILEVAR= variable because the PC operating systems do not return the fully qualified path.

Reading the filenames from a file

You can also create an external file that contains the names of the external files that you want to read, as long as the files are logically grouped. On MVS, for instance, these are members in a PDS. On VM/CMS, these are files that share a common FILENAME, FILETYPE, or FILEMODE. On the directory based systems these are files in the same directory. This can be done on all platforms in one way or another. Here are some examples:

On MVS you can use PROC SOURCE to generate a list of members of a PDS. See below:

```
PROC SOURCE NOPRINT NODATA INDD='USERID.LEVEL1.LEVEL2' DIRDD=OUT;
RUN;
```

The INDD option refers to a PDS, and the DIRDD option specifies a file to which 80 byte directory records are written. Details are found in the "SAS Companion for the MVS Environment". The file PROC SOURCE creates looks like this:

```
BATCHTSO      o " o          SASxxx
BATCH1        n  p          SASxxx
BATCH2        n  p "       SASxxx
BATCH3        & p p "       SASxxx
CONCAT        n  p |       SASxxx
DOLOPT        o  o   h h   SASxxx
```

On VM/CMS generate a list of files with a common FILEMODE by issuing the following command from the command line:

```
LISTFILE * * A (EXEC
```

This will generate a file called CMS EXEC A that will look like this:

```
&1 &2 PROFILE EXEC      A2
&1 &2 PROFILE SAVE     A1
&1 &2 PROFILE XEDIT    A1
&1 &2 PROFILE2 XEDIT   A1
&1 &2 REDTAP SAS        A1
&1 &2 REDTAP SASLOG     A1
&1 &2 REDTAPE LOG       A1
```

NOTE: This file is a little different from the rest that are generated in this section. All the rest have the filenames left aligned in the file that is created. In the file created on VM/CMS the filename begins in column 7.

For the PC based systems you can create the list of files like this:

```
DIR C:\dept\tsd\dataread\files\ /B >C:\DATA\LIST.LST
```

The resulting file will look like this:

```
first.dat
second.dat
third.dat
fourth.dat
```

On UNIX, like using the PIPE keyword on the FILENAME statement, the results that you get depend on how the command is structured. Generating a file that contains the fully qualified path and filename is done by issuing the following command:

```
ls -l /dept/tsd/dataread/files/* > v7test.files
```

The resulting file called "v7test.files" will look something like this:

```
/dept/tsd/dataread/files/test.log  
/dept/tsd/dataread/files/test.lst  
/dept/tsd/dataread/files/test.sas
```

With out the wildcard('*') in the command the file will look like this:

```
test.log  
test.lst  
test.sas
```

Now that the file is created, you can use it to read the files in the directory with the following DATA Step as a model. On the directory based systems you need to concatenate the fully qualified path to the front of the filename. This model shows that technique.

NOTE: This sample is based on the assumption that the file containing the list of filenames is created in the same directory where the files reside and contains only the names of the files to be read.

```
data one;  
length pftr $100;  
infile '/dept/tsd/dataread/files/v7test.files' trunccover;  
input ftr $40.;  
pftr = '/dept/tsd/dataread/files/'||ftr;  
infile dummy filevar=pftr trunccover end=done;  
do while(not done);  
  input var1 var2 var3;  
  output;  
end;  
run;
```

NOTE: "FILEVAR=variable" is the only form allowed for this option, therefore you must concatenate the path with the file's name into a variable to be used with FILEVAR=.

Other programming considerations

There are two statements to be careful of when reading multiple files with FILEVAR= because they can produce unexpected results. They are the subsetting IF and DELETE statements.

Both have the same effect. When the subsetting IF condition is not met, or when a DELETE statement is executed, the SAS system stops processing the current iteration of the DATA Step and control returns to the first executable statement following the DATA statement. This means that if the subsetting IF or DELETE statement is in the DO loop, no more records will be read by the DO loop, the current file is closed and the next file is opened for processing.