Paper 074-2009

The Perils of End-of-File Processing when Subsetting Data

Leonard Landry, Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

End-of-file processing in a SAS® DATA step occurs when an entire file is read and some specific processing takes place only after the last record is read from the dataset being processed. When selecting a subset of the data from the dataset using a subsetting IF statement, problems can arise due to improper placement of the end-of-file processing code. This paper will explain why the problem occurs and offer several suggestions on how to avoid it.

INTRODUCTION

SAS offers a simple way of detecting end-of-file by using the END=*variable* option on a SET or MERGE statement. This option will define a variable whose value is set to 1 when the last record is read. We can then perform some end-of-file processing by using a simple IF statement to test for a value of 1 for this variable.

The placement of this IF statement is critical when using a subsetting IF statement to process only a subset of the dataset. There are some situations when the code will not be executed if the IF statement is not properly placed. However, the problem may be difficult to detect as there are some situations where it will work properly and also when the code is not executed there will be no error message. Similarly, a problem may occur when executing a match/merge and attempting to perform some specific processing after all records are read.

The purpose of this paper is to show when the problem will occur, explain why it occurs, and offer several suggestions as to how it can be avoided.

EXAMPLE 1 – THE PROBLEM

The first example will demonstrate the problem when reading a SAS dataset and processing a subset of the dataset using a subsetting IF statement. We will use the following data as input.

Obs	Sex	VarA
1	M	10
2	F	5
3	M	20
4	F	15
5	M	15
6	F	5
7	M	12
8	F	20

The program below will read the entire dataset. As it reads the dataset it will accumulate the sum of VarA and after all records are read it will print to the log the accumulated sum of the variable VarA. Also shown below is the log which shows that the program worked successfully.

```
Data _null_;
   set test end=end1;
   retain total 0;
   total = total + VarA;
   if end1 then put total=; /* end-of-file processing */
run;

total=102
NOTE: There were 8 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Now we will try subsetting out only the records where sex = 'F' by using a subsetting IF statement. Here is the modified program and the log showing that it also works.

```
Data _null_;
   set test end=end1;
   if sex = 'F';  /* subsetting if */
   retain total 0;
   total = total + varA;
   if end1 then put total=; /* end-of-file processing */
run;

total=45
NOTE: There were 8 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Now we will change the subsetting IF statement to subset out the records where sex = 'M'. The program is shown below with the log which, this time, does not contain the total, indicating that the end-of-file processing code did not execute.

```
Data _null_;
   set test end=end1;
   if sex = 'M';   /* subsetting if */
   retain total 0;
   total = total + varA;
   if end1 then put total=; /* end-of-file processing */
run;

NOTE: There were 8 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Although it may not be obvious why one of these programs works and not the other, the problem occurs only when the last record in the dataset is deleted by the subsetting IF. The reason this happens is that when a record is deleted by a subsetting IF, SAS stops processing and returns to the next iteration of the DATA step. Thus, any executable statements placed after the subsetting IF do not get executed.

EXAMPLE 1 – THE SOLUTION

There are many ways to avoid this problem. Some programmers make it a practice to never use a subsetting IF. The program could easily be rewritten without the subsetting IF such as the following and would give the correct results as shown here.

```
Data _null_;
   set test end=end1;
   retain total 0;
   if sex = 'M' then total = total + varA;
   if end1 then put total=; /* end-of-file processing */
run;

total=57
NOTE: There were 8 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

Another option would be to use a WHERE statement instead of the subsetting IF. This option may not always be available depending on the criteria on which the subsetting is based but when it is available it will work. The program and log below show that this approach works.

```
Data _null_;
   set test end=end1;
   where sex = 'M';
   retain total 0;
   total = total + varA;
   if end1 then put total=; /* end-of-file processing */
run;

total=57
NOTE: There were 4 observations read from the data set WORK.TEST.
      WHERE sex='M';
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.00 seconds
```

Another option is to use the subsetting IF statement and place the end-of-file processing code before the SET statement. This will work because it is the SET statement that triggers the end of the implicit looping. After the last record is read and processed (or not processed as in the case where it is deleted by the subsetting IF) control returns to the next iteration of the DATA step and then stops when it reaches the SET statement and there are no more records to read. This is illustrated in Figure 1, the flow chart shown below from SAS documentation "Step-by-Step Programming with Base SAS® Software, How the DATA Step Works: A Basic Introduction". The following code shows this technique along with the log which verifies that it works.

```
Data _null_;
   if end1 then put total=; /* end-of-file processing */
   set test end=end1;
   if sex = 'M';   /* subsetting if */
   retain total 0;
   total = total + varA;
run;

total=57
NOTE: There were 8 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

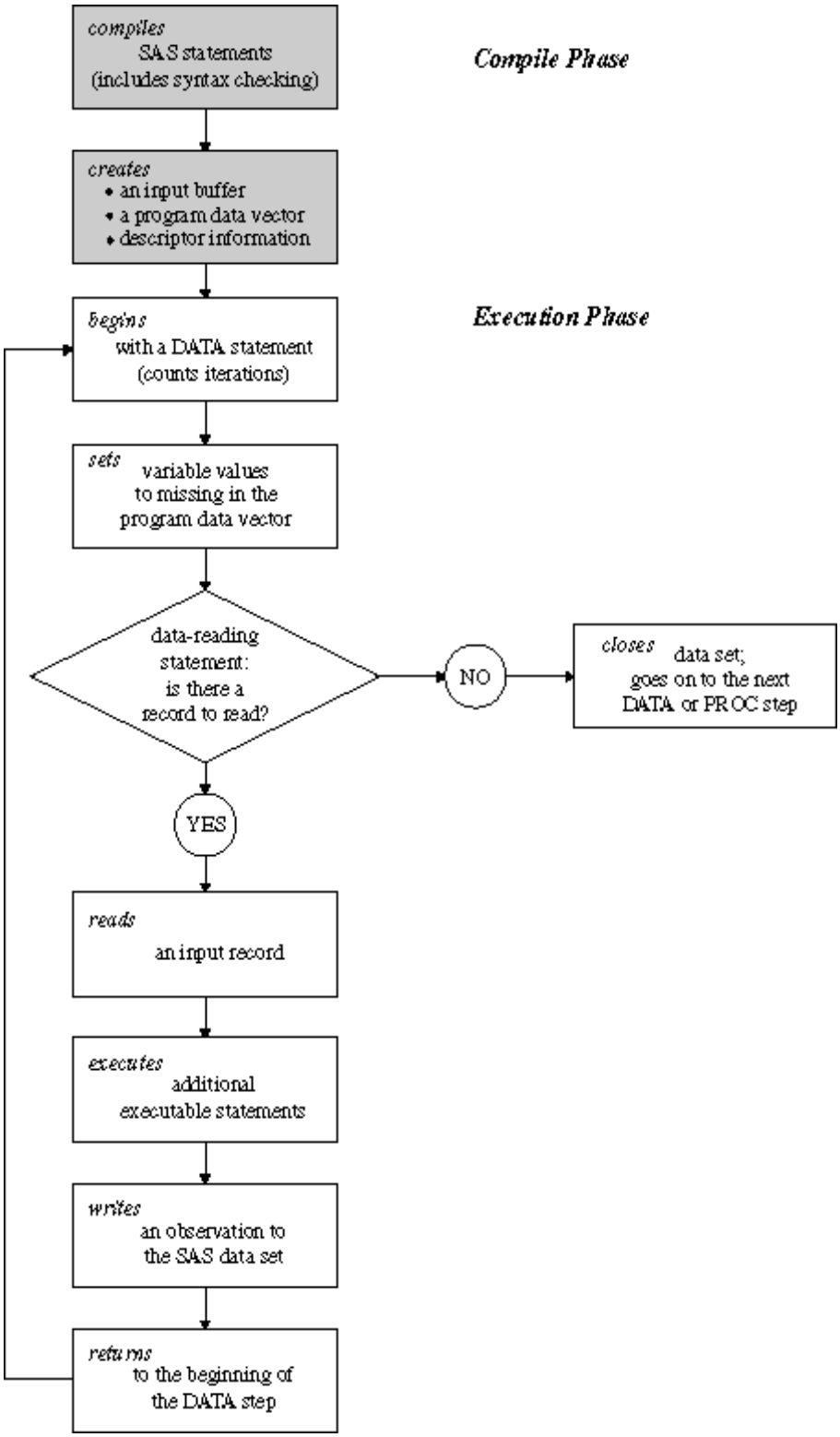

Figure 1

Flow of Action in a Typical DATA Step

EXAMPLE 2 – THE PROBLEM

The second example will demonstrate the problem when performing a match/merge and selecting only the records common to both files. We will use the following data as input.

	FileA		FileB
ID	**VarA**	**ID**	**VarB**
1	**10**	**1**	**5**
2	**15**	**2**	**8**
3	**12**	**4**	**25**
5	**20**	**5**	**11**

The program below will perform a match/merge keeping only the records where the value of ID is the same in both FileA and FileB. For these records, sums will be accumulated for VarA and VarB. After all records are read these two sums will be written to a SAS dataset called sums. The log shows that there were 3 records retained where the value of ID matched and 1 record was written to the dataset "sums".

```
Data FileAB(keep=ID varA varB) sums(keep=SumvarA SumvarB);
   retain SumvarA SumvarB 0;
   merge FileA(in=inA) FileB(in=inB) end=end1;
   by ID;
   if inA & inB;   /* subsetting if */
   SumvarA = SumvarA + VarA;
   SumvarB = SumvarB + VarB;
   output FileAB;
   if end1 then output sums; /* end-of-file processing */
run;

NOTE: There were 4 observations read from the data set WORK.FILEA.
NOTE: There were 4 observations read from the data set WORK.FILEB.
NOTE: The data set WORK.FILEAB has 3 observations and 3 variables.
NOTE: The data set WORK.SUMS has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

To illustrate how the problem can occur we will add one additional record to FileB having an ID of 6 and rerun the same code. The data now looks like this.

	FileA		FileB
ID	**VarA**	**ID**	**VarB**
4	**10**	**1**	**5**
5	**15**	**2**	**8**
6	**12**	**4**	**25**
5	**20**	**5**	**11**
		6	**20**

Below is the log from this run. Again there are the same three records that are retained having the common ID but this time the "sums" dataset has 0 observations indicating that the end-of-file processing code did not execute.

```
NOTE: There were 4 observations read from the data set WORK.FILEA.
NOTE: There were 5 observations read from the data set WORK.FILEB.
NOTE: The data set WORK.FILEAB has 3 observations and 3 variables.
NOTE: The data set WORK.SUMS has 0 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

The problem in this case is the same as in Example 1. The statement "IF inA & inB;" is a subsetting IF statement. When the last record is read from FileB and the ID is not common to both files the subsetting IF causes SAS to stop processing and return to the top of the DATA step so the end-of-file processing code is not executed. When the MERGE statement is executed there are no more records to read from either FileA or FileB and the DATA step closes.

EXAMPLE 2 – THE SOLUTION

In this situation a WHERE statement cannot be used as the criteria in the subsetting IF are not fields in the SAS datasets. Similar to the solutions shown above in Example 1 we can avoid the use of the subsetting IF or we can place the end-of-file processing code before the MERGE statement. Both techniques are shown below with the logs showing that 1 record was written to the "sums" dataset.

```
Data FileAB(keep=ID varA varB) sums(keep=SumvarA SumvarB);
   retain SumvarA SumvarB 0;
   merge FileA(in=inA) FileB(in=inB) end=end1;
   by ID;
   if inA & inB
      then do;
               SumvarA = SumvarA + VarA;
               SumvarB = SumvarB + VarB;
               output FileAB;
        end;
   if end1 then output sums; /* end-of-file processing */
run;

NOTE: There were 4 observations read from the data set WORK.FILEA.
NOTE: There were 5 observations read from the data set WORK.FILEB.
NOTE: The data set WORK.FILEAB has 3 observations and 3 variables.
NOTE: The data set WORK.SUMS has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
Data FileAB(keep=ID varA varB) sums(keep=SumvarA SumvarB);
   retain SumvarA SumvarB 0;
   if end1 then output sums; /* end-of-file processing */
   merge FileA(in=inA) FileB(in=inB) end=end1;
   by ID;
   if inA & inB;
   SumvarA = SumvarA + VarA;
   SumvarB = SumvarB + VarB;
   output FileAB;
run;

NOTE: There were 4 observations read from the data set WORK.FILEA.
NOTE: There were 5 observations read from the data set WORK.FILEB.
NOTE: The data set WORK.FILEAB has 3 observations and 3 variables.
NOTE: The data set WORK.SUMS has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

CONCLUSIONS

The examples presented in this paper show that when using a subsetting IF statement improper placement of the end-of-file processing code can result in the code not being executed. The problem may go undetected as it will only occur when the last record read is deleted by the subsetting IF and no error messages will be written to the log. The problem does not occur when subsetting using a WHERE statement. Placing the end-of-file processing code before the SET or MERGE statement will avoid the problem.

REFERENCES

Step-by-Step Programming with Base SAS® Software – SAS Institute

CONTACT INFORMATION

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

Leonard Landry
Database Manager,
Business and Labour Market Analysis Division,
Statistics Canada
leonard.landry@statcan.gc.ca