Paper 029-31

Gracefully Terminate a DATA step If the Input Data File is Not Available

Erik W. Tilanus, independent consultant

ABSTRACT

The standard method to read external data into a SAS® data set consists of the following steps:
- Allocate the data file using a FILENAME statement
- Create a DATA step to read the data using INFILE and INPUT statements.

The allocation creates a link between the SAS program and the data file. Normally SAS will access the data file during the compilation phase of the DATA step to determine the availability and the attributes of the file.
If the data file is not available, the DATA step will fail during the compilation and you are left without control about what to do next.

By using the FILEVAR option of the INFILE statement you can move the process of allocation from the compilation to the execution phase of the DATA step and regain control over what is happening with the input file. This allows for corrective actions in case that the input data file is not available.

INTRODUCTION

The standard method to read data into a SAS data set consists of the following steps:
- Allocate the data file using a FILENAME statement
- Create a DATA step to read the data using INFILE and INPUT statements.

In that DATA step you have full control over the read process and you may set indicators for subsequent steps to signal successful or unsuccessful completion of the process, if so desired.

However if the input data file is not available, the DATA step will fail already in the compilation phase and you do not have control over the situation. Consequently you cannot flag an "unsuccessful" completion.

This paper describes a method to circumvent this problem and give control back to the DATA step programmer and signal the missing input.

THE PROBLEM

Following code is a typical example of how a text file can be read into a SAS data set:

```
FILENAME MyInput 'e:\sugi31\MyData.tsv';
DATA MyData;
RETAIN ErrorCount 0;
INFILE MyInput DELIMITER='09'x MISSOVER DSD LRECL=32767 FIRSTOBS=2 END=eof;
INFORMAT product $8. country $2. january february march april may june july
         august september october november december best32.;
INPUT product $ country $ january february march april may june july
      august september october november december;
IF _ERROR_ THEN ErrorCount +1;
IF eof THEN CALL SYMPUTX('Status', ErrorCount);
RUN;
```

When you submit this code and everything is all right, the DATA step will start reading the input data from "mydata.tsv" and builds the SAS data set "MyData". If SAS encounters errors while reading the data, it will set the _ERROR_ automatic variable and adds 1 to the variable ErrorCount. After the last record from the input has been read, the number of encountered errors is stored in the macro variable Status. Based on the value of Status you can determine what to do with subsequent steps. Note that this code assumes that the first record of the file contains header fields, rather than data.

When you look at it in more detail, you will see that SAS accesses the data file during the compilation of the DATA step, when it encounters the INFILE statement. It does so to determine the attributes of the data file in order to call in the right reading routines.

WHAT IF THE INPUT DATA FILE IS NOT AVAILABLE?

If the input data file does not exist or is not accessible for some reason, the situation is different. In that situation the DATA step fails already in the compilation phase. So the step is never executed, which means that you don't have control over it and cannot set any flags.

THE SOLUTION

We will present two solutions to the problem of the missing input data file.
In the first one we simply set a macro variable to flag that the DATA step has executed. In the second one we claim control during the execution of the DATA step, regardless whether the input file is available or not.

SOLUTION 1: A SIMPLE TEST ON A MACRO VARIABLE

In this solution we simply add a %LET statement before the DATA step:

```
   %LET Status=-1;
```

Now the value of the macro variable Status will be –1 if the DATA step did not execute. If it was executed the value of Status will be 0 or positive, if data errors were encountered. But the SAS log does not look very friendly:

```
ERROR: Physical file does not exist, e:\sugi31\mydatax.tsv.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.MYDATA may be incomplete.  When this step was stopped there
         were 0 observations and 15 variables.
```

In background processing SAS normally takes one more action: it sets the system option OBS=0 and goes into syntax checking mode:

```
NOTE: SAS set option OBS=0 and will continue to check statements.
      This may cause NOTE: No observations in data set.
```

From there on no steps are executed. That would mean that you cannot perform any "damage control" actions. Well, in fact you can. In syntax check mode you still can execute macro's and you can reset the options that SAS sets by inserting an OPTIONS statement after the failing DATA step:

```
   OPTIONS OBS=MAX NOSYNTAXCHECK;
```

Most likely you just want to signal the fact in the SAS log and stop completely. That is what the next macro does:

```
   %MACRO ErrorTest;
     %GLOBAL Status;
     %IF &Status = -1 %THEN %DO;
       %PUT Failure to read input data;
       ENDSAS;
     %END;
   %MEND;

   %ErrorTest
```

Although we now successfully trapped the fact that the input data file was not available, you can hardly call the way the DATA step is terminated "gracefully".

SOLUTION 2: CLAIM CONTROL BACK

In this solution we claim control back. The DATA step execution is started regardless the availability of the input data file. This is accomplished by changing the FILENAME statement to a dummy:

```
   FILENAME MyInput Dummy;
```

Now SAS cannot determine during the compilation phase that the data file is not available and it will start the execution. We have control! But now we have to create the link with the data file and test whether it is available in the DATA step ourselves:

```
   FILENAME myinput dummy;
   %LET Filename =e:\sugi31\mydata.tsv;

   DATA mydata;
   RETAIN InputFile "&Filename" ErrorCount 0;
   IF FILEEXIST("&filename") THEN DO;
      INFILE myinput DELIMITER='09'x MISSOVER DSD LRECL=32767 FIRSTOBS=2
             FILEVAR=InputFile END=eof;
      INFORMAT product $8. country $2. january february march april may june july
               august september october november december best32.;
      INPUT product $ country $ january february march april may june july
            august september october november december;
      IF _ERROR_ THEN ErrorCount +1;
      IF eof THEN CALL SYMPUTX('Status',ErrorCount);
   END;
   ELSE DO;
      PUT "Input file &filename not available";
      CALL SYMPUTX('Status',-1);
      STOP;
   END;
   RUN;
```

The FILEVAR option in the INFILE statement allows you to specify the input file at execution time. So what we do is we store the path and file name in the macro variable Filename and assign that value to the variable InputFile. Next we test whether the file is there with the FILEEXIST function and if it exists we use it in the FILEVAR option of the INFILE statement to read the input. If it does not exist, we issue the error message and stop the DATA step gracefully with the STOP statement. No error messages or warnings in the SAS log.

A nice bonus of this approach is, that you can identify the missing data set in the error message. So if you have more than one input data set, you can indicate which one fails. You can also decide to continue with the other input files and leave the missing file out. It is all under your control.

FINE-TUNING THE SOLUTION

The code as shown above checks whether the data set is available with every path through the DATA step. This takes a lot of extra time and processing. When it is sufficient to check the availability only at the start of the step you can change the first lines of the DATA step as follows:

```
   DATA mydata;
   RETAIN inputfile "&filename" ErrorCount 0 fileOK;
   IF _N_=1 and FILEEXIST("&filename") THEN fileOK=1;
   IF fileOK THEN DO;
      INFILE ...
```

You can also decide to check the availability periodically. Next lines check the availability of the file after each 10 records:

```
   DATA mydata;
   RETAIN inputfile "&filename" ErrorCount 0 fileOK;
   IF INT(_N_/10)=_N_/10 OR _N_=1 THEN DO;
      fileOK=0;
      IF FILEEXIST("&filename") THEN fileOK=1;
   END;
   IF fileOK THEN DO;
      INFILE ...
```

On my computer system (AMD Athlon XP2700+, 512Mb internal memory) I clocked the following times, with an input data file of 100,000 records on a local hard disk:

Test frequency	Elapsed time	CPU time
Only at first record	0.53 sec	0.51 sec
Every 10 records	1.31 sec	1.29 sec
At every record	7.53 sec	7.43 sec

Next I moved the input data file to another computer in the network. The network consisted of a 100 Mbps cable connection between my computer and a router and a 54 Mbps wireless connection from the router to the other system. Both systems were running Windows XP. The measured timings are dramatically different:

Test frequency	Elapsed time	CPU time
Only at first record	5.34 sec	0.60 sec
Every 10 records	98.20 sec	4.29 sec
At every record	980.15 sec	35.71 sec

Apparently Windows takes a lot of time to handle the FILEEXIST function call over the network.

RECOMMENDATION

When the data file resides on a local hard disk, it is unlikely that it will disappear once you have started the DATA step. Therefore it makes sense to test only during the first pass through the step.

Given the timing effects when checking via a network connection, you probably want to do the same, in case you access the data file via the network. But if the network fails, you get a nasty error messages. This is what was reported in the SAS log when I pulled the network plug half way through:

```
ERROR: Undetermined I/O failure.
FATAL: Unrecoverable I/O error detected in the execution of the data step program.
       Aborted during the EXECUTION phase.
NOTE: 67089 records were read from the infile MYINPUT.
      The minimum record length was 44.
      The maximum record length was 69.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.MYDATA may be incomplete.  When this step was stopped
there were 67088 observations and 16 variables.
```

When testing at every record, the chances of a graceful termination are much better. Only in case the connection is broken between the execution of the FILEEXIST function and the INPUT statement you would have an error as above. Otherwise the SAS log will contain the following lines:

```
Input file \\Notebook\DATA (E)\SUGI31\mydata.tsv not available
NOTE: 2042 records were read from the infile MYINPUT.
      The minimum record length was 44.
      The maximum record length was 56.
NOTE: The data set WORK.MYDATA has 2042 observations and 16 variables.
NOTE: DATA statement used (Total process time):
      real time           30.03 seconds
      cpu time            0.62 seconds
```

Note that the first line is written by the PUT statement at the end of the DATA step.

CONCLUSION

By moving the allocation of the input file to the execution phase of the DATA step it is possible to monitor the availability of the input data. Given the extra processing to check the availability of the input data file, it is recommended to check only once at the first pass through the DATA step.

The presented code shows an example with one input data file. It is left to the reader to expand it to more than one input file.

REFERENCES

For more information about the FILEEXIST function and the CALL SYMPUTX routine and any other statements used in the sample code, please refer to:

SAS Institute Inc. (2004), *SAS 9.1 Language Reference: Dictionary.* Cary NC: SAS Institute Inc.

This language reference is also available in OnlineDoc.

CONTACT INFORMATION

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

Erik W. Tilanus
Horstlaan 51
3941 LB Driebergen-Rijsenburg
the Netherlands
Phone and fax: +31 343 517007
Email: erik_tilanus@compuserve.com