Paper CT07

Customizable Smart Search Tool for Project Leads

Bibas Timilsina, Parexel, Sheffield, UK

ABSTRACT

The FILENAME statement has been around a long time. It has been traditionally used to reference an external file on disk. However, there are many “not so well-known” functionalities, for which the FILENAME statement is a boon. This paper discusses, the various tricks to develop the customizable search tool using the power of FILENAME statement together with well-known PIPE option to search the text token in the various file types.

This paper is based on real time usage of FILENAME statement with PIPE option to search the data point from Raw, SDTM or ADaM database, identifying the common possible data issues from database, developing customizable personal standard tools such as log checker, QC automation, derivation count etc.

INTRODUCTION

The SAS FILENAME statement with PIPE option is a powerful utility that provides users with the ability to search and filter through large amounts of files to identify specific information based on specific criteria such as file type or file name patterns.

During the project work as a programming lead, we come up with the various scenarios/requests where we need to scan the information from numerous data source, can be related to programs, datasets, specifications, outputs, logs etc. We must deal with many urgent requests which will not allow the lead programmers to delegate the task to their support teams, such as, the lead may have to extract the information and share with vendors during the ongoing meetings.

Following are the common scenarios where we can use the search logic to extract the information quicker.
Identifying the

- Reference output programs using specific parameters, datasets, and/or analysis.
- Possible updates based on known scenarios for existing programs.
- Common data issues in SDTM level.
- Run status.
- Recent external files and adding automation on SDTM and ADaM programming.
- Log issues.
- QC status by scanning compare output files.
- Good programming practice related issues.
- List of program files for batch processing.

To understand the search logic first we need to understand the basic about the Filename statement. The wide array of data access methods available to the SAS programmer today has resulted in a decrease in the number of new SAS programmers who explore the FILENAME statement. In fact, in one recent survey, only 14% of users reported using the FILENAME statement as a means of accessing their source data (Milum, 2011). Whereas this decline has come about due to an improved toolset that ultimately makes data access more efficient, this increased efficiency has come at the cost of many new SAS users not being exposed to this powerful, flexible tool. Therefore, the current work attempts to close this gap in knowledge by introducing the reader to the FILENAME statement, providing some oft-encountered techniques for manipulating flat files with the INFILE and FILE statements, and demonstrating several popular (but less frequently utilized) techniques in which the FILENAME statement plays a central role.

“To get started on this exploration, consider how the FILENAME statement was described in The SAS Language Guide for Personal Computers (Release 6.03 Edition) (SAS, 1988): The FILENAME statement associates a SAS fileref (a file reference name) with an external file's complete name (directory plus file name). The fileref is then used as a shorthand reference to the file in the SAS programming statements that access external files (INFILE, FILE, and %INCLUDE). Associating a fileref with an external file is also called defining the file. Use the FILENAME statement to define a file before using a fileref in an INFILE, FILE, OR %INCLUDE statement.”

From this definition the FILENAME statement can be used to generate a symbolic link to an external file (or device) that, in turn, can be used to refer to the file elsewhere in the SAS program.

The FILENAME statement can be used to interact with the world outside of SAS in several interesting ways beyond reading and writing text files.

1. Reading data from an external file: fileref to the external file and then use the fileref in subsequent DATA or PROC steps to read the data from that file into SAS.

2. Writing data to an external file: fileref to an external file and then use the fileref in a DATA step to write data from SAS to that file.

3. Accessing system files: fileref to a system file, such as a text file containing configuration settings or parameters. This allows you to read or modify the contents of the system file within your SAS program.

4. Importing and exporting data: fileref to an input or output file. This helps to establish a connection between SAS and the external software for seamless data transfer.

5. Accessing external databases: fileref to an external database, allowing you to read or write data directly from/to that database within your SAS program.

SYNTAX

The syntax for the most basic usage of filename statement is as follows:

***FILENAME** FILEREF <DEVICE-TYPE/ACCESS METHODS> 'EXTERNAL-FILE' <ENCODING='ENCODING-VALUE'> <OPTIONS><OPERATING-ENVIRONMENT- OPTIONS>;*

Fileref is any valid sas name of length 8 for file reference.
External file is the physical name (recognized by the OS), of an external file.
Encoding: It indicates that the external file has a different encoding from the current session's encoding. While reading the data from encoded file, sas transcodes the data.If the file you want to read (inp_fl in the example below) is in UTF-8 encoding and your session is in Wlatin1 encoding, you need to explicitly specify the encoding of the input file, as SAS assumes that an external file is in the same encoding as the session encoding.
For example:

```
filename inp_fl "physical location of file" encoding="utf-8" ;
data xcoding;
   infile inp_fl;
   input patient aeseverity $ ;
run;
```

Device-type/access methods: It specifies the type of device or the access method that is used if the fileref points to an input/output device or location that is not a physical file. Note that there is a distinction to be made between a device and an access method. A device is a physical computer component whereas an access method can be defined as an interface to a storage medium.

DEVICE TYPES: Some of the interesting device types are discussed in detail below:
DISK: By default, the files referenced in the filename statement are considered to be on the DISK. So, you don't have to explicitly mention it in the SAS code.
For example:

```
filename inp_fl "C:\intemp\inp_fl.sas";
filename out_fl "C:\outtemp\out_fl.sas";
data _null_;
```

```
    infile inp_fl;
   file out_fl;
  input;
 put _infile_;
run;
```

Note that in above statement, DISK hasn't been used (though the device type used is DISK)

PIPE: Unnamed pipes enable you to invoke a program outside of SAS and redirect the program's input, output, and error messages to SAS. This capability enables you to capture data from a program external to SAS without creating an intermediate data file. For example, if you want to get the list of all the files in a particular folder, as SAS data set (for further processing), you can use the following:

```
 /*here temp is temporary folder which contains files;*/
filename DIRLIST pipe 'c:\temp';
```

The directory specified by the above location can have any type of file. If you want to output just the SAS files, then use the code below:

```
filename DIRLIST pipe 'c:\temp\*.sas';
data dirlist;
        length filename $200;
        infile dirlist length=reclen;
        input buffer $varying200. reclen;
run;
```

TEMP: Many times, you want to refer to temporary files which are active only for the current SAS session. You don't have to worry about their name, location, deletion etc (as you want a temporary file). The temporary file can be accessed only through the logical name and is available only while the logical name exists.
For example, if you want to create a small code inside a program and then include the same in your code, then you can use the following:

```
filename code temp;
data _null_;
    file code;
   put "proc sort;";
  put "by a;";
 put "run";
run;
data test;
 a=1;
run;
%include code;
```

Or suppose you want to generate a SAS output in both RTF and PDF format, then you can assign a temporary fileref to the output and pass this temp file to respective macros:

```
filename in_fl TEMP;
proc printto new print = in_fl;
run;
proc report
... ... ...;
... ... ... ...;
run;
proc printto ;
run;
%out2rtf(tmpfl=in_fl);  *here this macros generates rtf outputs.
%out2pdf(tmpfl=in_fl); *here this macros generates pdf outputs.
```

DUMMY: While programming, you frequently create dummy data sets and dummy variables. They are mostly used for testing or troubleshooting purposes. Similarly, DUMMY device can be used to provide as input, an empty file or to discard an output file.
For example, suppose you have a macro test.sas, in which a fileref is a mandatory macro parameter. In this case, you can create a dummy fileref as:

filename mand_par dummy;

Now you can pass this to the test macro to avoid unnecessary error.

ACCESS METHODS: As mentioned, access methods acts as an interface to different media. Some of them are CLIPBOARD, CATALOG, URL, EMAIL, FTP, DDE
CLIPBOARD: This access method enables us to read text data from and write text data to the clipboard on the host computer. For example an entire data set can be written to the clipboard and then can be read from there.
This example writes 2 lines to the clipboard and then writes it into the test dataset.

```
filename clip1 clipbrd;
data _null_;
        file clip1;
        put 'test 1';
        put 'test 2';
run;
data test;
        infile clip1;
        input;
        put _infile_;
run;
```

CLEARING THE FILE REFERENCE:
For clearing the filerefs, CLEAR option is used. It disassociates one or more currently assigned filerefs. Specify _ALL_ to disassociate all currently assigned filerefs.
For example:

```
filename test clear; *to dissociate "test" fileref;
filename _all_ clear; * to dissociate all the file references (filerefs);
```

SCRIPT TEMPLATE

The below simple macro script can be used to search the text token 'HISTO' from the .sas files present in the folder 'dummy1/dummy2/dummy3/dummy4/dumy5/share/data_analysis/prod/program/' and creates the sas dataset 'check_histo1' which we can clean and use for further processing as per our requirement.

```
%macro getfilename  ( path              =,
                      Out               =,
                       Keyword          = ,        /*blank for LS option*/
                        Filenameoptions = grep,    /*LS, GREP, DIR...*/
                         filetype       =sas       /* file extension sas, log, lst, txt, out, sas7bdat, ...*/
                      );
  %let primarypath=&path.;
   filename check pipe " &filenameoptions. -i &keyword. &primarypath.*.&filetype.";
   data &out.;
     length name $1000.;
     infile check dlm='??';  /*dlm can be any special character which is not expected in the files that we are scanning*/
     input name;
       /*scenario based conditions*/
     filename=upcase(scan(scan(name,10,'/'),1,'.'));
  run;
%mend getfilename;
```

```
/**** example call****/
%getfilename(path=dummy1/dummy2/dummy3/dummy4/dumy5/share/data_analysis/prod/program/,/*physical path*/
                out=check_HISTO1,      /* output dataset name*/
          keyword= HISTO1 ,       /*blank for ls*/
   filenameoptions= grep,   /*ls, grep, Dir...*/
           filetype=sas          /*file extension sas, log, lst , sas7bdat, out, txt,.....*/
                );
```

The above program can be used for various purpose and add scenario based condition to clean up the final report and pass the final report to excel using below simple SAS export.

```
proc export outfile="out file path/output.xlsx"
    data=final
    dbms=xlsx replace;
    putnames=yes;
run;
```

SAMPLE LOG CHECKER

With the minimal update on above tool, we can make it to multiple token scanner and can be the starting template for log checker handy tool

```
%macro getfilename(path=,
                   out=,
          keywordlist=,
        filenameoptions= grep
                filetype=sas
                );
%let primarypath=&path.;
%let loopmax=%sysfunc(countw(&keywordlist.,'@'));
%do loop=1 %to &loopmax. ;
  %let keyword=%scan(&keywordlist.,&loop.,'@');

          filename check pipe " &filenameoptions. -i &keyword. &primarypath.*.&filetype.";
          data &out._&loop.;
                length name $1000.;
                infile check dlm='??';
                input name;
                dataset1=upcase(scan(scan(name,11,'/'),1,'.'));
               if index(name,"&keyword.");
        run;
%end;
data &out.;
  set &out._:;
run;
proc sort data=&out. nodupkey;
  by dataset1;
run;

/*<add further required process and generate the suitable report >*/

%mend getfilename;
/**** example 1****/
%getfilename(path=dummy1/dummy2/dummy3/dummy4/dummy5/data_analysis/dummy_reporting/prod/log/,
                out=log,
      keywordlist= warning@error@not found,
  filenameoptions= grep,
          filetype=log
                );
```

CONCLUSION

This paper discusses scanning capability of the FILENAME statement (with handy tool's example templates). The FILENAME statement with PIPE option in conjunction with the power of the DATA step provides a very flexible scanning tool to process various file types. The more you explore it, the more you unravel the power of THE FILENAME Statements with PIPE option.

REFERENCES

1) Statements: FILENAME Statement - 9.2 (sas.com)
2) The Power of "The FILENAME" Statement

ACKNOWLEDGMENTS

I would like to acknowledge and thank to the PHUSE Coding Tips & Tricks chairs for their hard work and review comments on papers and presentations.

CONTACT INFORMATION

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

Author Name: Bibas Timilsina
Company: Parexel
Address: Parexel International, Navigation House, 1 South Quay Drive, Sheffield, S2 5SU, United Kingdom
Email: Bibas.timilsina@parexel.com
LinkedIn: linkedin.com/in/bibas-timilsina-2075775b
Web: https://www.parexel.com/