PP14

# An Animated Guide: A program to Read information from Program Headers and Produce Reports for Programming Managers

Russ Lavery, Contractor, Bryn Mawr, U.S.A.

## ABSTRACT

This paper describes, and provides to the public, a program that can be used to help manage a department of SAS programmers. The program will, if a user specifies one or more, top level directories on a windows machine, recursively search under those top level directories and find every SAS program and SAS table under those directories. It will then create data sets containing information on every file and program. It will also create pdf documentation on programs using the program headers and comments. Compare this functionality to that of the dictionary tables which only contain data about tables in *established* librefs and no information about programs.

## INTRODUCTION

I propose to start the paper with some context that suggests why this program might be useful. A few years ago, I was transferred to a position where I was the "babysitter" for a departmental server. Soon after starting, I was told we would be migrating to a new server and a new OS. To be an effective babysitter and/or migrator, I needed to know what was going on in the current server and there was no documentation on the scheduled production programs, the files they used, or the files they created.

It was a fast-paced, often ad-hoc, environment filled with both one-of requests and scheduled production programs. As one manager told me "We're not big on documentation or control here" and he was correct. At that client, tables were "lost" and programs were "lost" – by lost I mean created, used and forgotten. Programs and tables would later be "found" (really noticed by someone new) and no one would know: who owned that file, why it had been created or if it could be deleted to free up space. When a source file from a vendor was changed, or discontinued, management did not have any record of who (actually, what production programs) used that source file. Management could not give advance notice of "problems to come" because of source file changes. The experience was that source files were changed – scheduled programs stopped running – and people scrambled to fix things up.

As part of the migration, the migration team needed to know what files were used by the heavily macro-ized (and completely lacking in comments) production programs – programs that had NO program headers at all. It took four man-weeks of reading SAS code to find all the tables that were used and created by the production programs. There was a great desire, among the migration team, that there should be some way to manage all the tables and programs. We had several discussions about how great it would be to use a management console and have production programs, input tables and output tables "registered" in one spot. We wanted metadata and did not have it.

The projects done, for that client, used too much brute force and too little automation. The pain of that work still lingers and motivated this paper. I was talking, with Frank DiIorio, one day and he mentioned that he had a program that used a DOS command to search under a directory and find programs. His program pulled off the program headers and put them into a pdf file. He agreed to share his program and that was the start for this paper and this program.

This program only works on a windows machine because it uses a DOS command to search under a list of top-level directories to find all SAS programs and tables. It then creates several types of output that ca be used to monitor, and manage, programming activity. The program could be converted to a unix version by creating a unix command that recursively searches under directories.

The program creates four output files and the four files can be used, by a manager, to answer questions like:

What programs are finished, and what are under development, for some client or study
What tables does a program use and what output does a program create
What internal client needs to be notified about a problem with a source file
What programs has a particular programmer worked on in a time period
What tables were created in a certain time period
and many more questions

I addition, the program creates pdf documentation for each program, using headers and comments. This is similar to Javadoc.

## AN ABBREVIATED HEADER – AND EXAMPLES OF STANDARD COMMENTS:

```
/*************************************************************************
** Start entries after the *. Start in col 31 or 32
Program: 1* X_Required_45CharMax_X
Dev/Active/Inactive/AH: 2* X_Required_45CharMax_X
Much standard header omitted
Maintenance: this is not currently parsed
   Much of the header not shown
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX

*************************************************************************/

/*********************************************************
** section 1 : reads raw data files
*********************************************;
** comment 1 Business rule;
some SAS code
/*This comment will not end up in the documentation */
** comment 2 Business rule;
some more code /*This comment will not end up in the documentation */

/*********************************************************
** section 2 : merge files
*********************************************;
** comment 3 Merge files and do cool stuff;
Additional SAS code

/*********************************************************
** section 3 : Create output files
*********************************************;
** comment 4 Explain the use of the following export;
export code
** comment 5 Explain the use of the following export;
export code
```

The program in the appendix uses the full standard header and standard comments (see above for examples) and will create documentation on the programs.

## DETAILS OF THE STANDARD HEADER:

The program header is quite lengthy and is shown below. Creating such a header, from scratch, would be cumbersome and likely not maintained. However, the shell of the header can be pasted into a program using an abbreviation file that can be found on my web site (russ-lavery.com) and then installed, in your SAS environment in less than thirty seconds. When the abbreviation file is installed, you can type DOCBLOCK<ret> and SAS will paste in the shell of a standard header – a header that the program in the appendix of this paper will parse.

```
/*************************************************************************
** Start entries after the *. Start in col 31 or 32
Program: 1* X_Required_45CharMax_X
Dev/Active/Inactive/AH: 2* X_Required_45CharMax_X
Programmer: 3* X_Required_45CharMax_X
Program Owner: 4* X_Required_45CharMax_X
Business Owner: 5* X_Required_45CharMax_X
Date Pgm. Started: 6* X_Required_ISO8601_EG1997-07-16_X
Date First Into Production: 7* X_Required_ISO8601_or DEV_X
Most Recent Into Production:8* X_Required_ISO8601_EG1997-07-16_X
Client: 9* X_Required_45CharMax_X
Study: 10* X_Required_45CharMax_X
Purpose:11** X___Required_60CharMax___X
***********
Preceeding: 12 (prog name or NONE) Programs that must be run before this can be run (prog
name or
NONE)
X_Optional_45CharMax_X
X_Optional_45CharMax_X X_Optional_45CharMax_X
Concurrent: 13 (prog name or NONE) Programs that can be run concurrent with this
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Following: 14 (prog name or NONE) Programs that Can only be run after this finishes
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Driver files: 15 (file name or NONE) External files containg metadata to control program
execution
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Infiles: 16(table name or NONE) perm files USED by this program
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Outfiles: 17(table name or NONE) perm files CREATED by this program
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Macros Called: 18(macro name or NONE) perm macros called by this program
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Formats Called: 19(format name or NONE) perm formats called by this program
X_Name/NONE_45CharMax_X
X_Name/NONE_45CharMax_X X_Name/NONE_45CharMax_X
Maintenance: this is not currently poarsed
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
*************************************************************************/
```

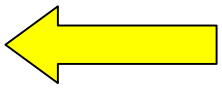
The bits of information in the header are data about the program that follows it. The program in the appendix turns these bits of information into data in a SAS table. A SAS file is created that holds all the variables in the header and allows a manager to run queries against that information. Important pieces of information in the header are:

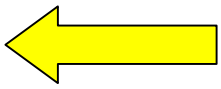
| | |
|---|---|
| Program: | Program name |
| Dev/Active/Inactive/AH: | Program status:<br>**Dev**elopment: being written or modified<br>**Active**: Debugged and able to be run and currently being run – a production program<br>**Inactive**: Debugged and able to be run. Was prod. and removed from prod. for some reason. |

| | |
|---|---|
| | **AH**: Ad-hoc program. Debugged and able to be run. Run at request of client |
| Programmer: | Name of person who wrote program |
| Program Owner: | Programming Manager. Approver of changes to program. |
| Business Owner: | Internal client for output. Provider of business rules. Approver of changes to program. |
| Date Pgm. | Started: Date Development started |
| Date First Into Production: | Date first into production |
| Most Recent Into Production | Date when most recent mod put into production (often the production directory) |
| Client: | Internal/external client name |
| Study: | Id for project |
| Preceding: (Short for preceding programs) | |
| | Programs that must be run before this can be run (prog name or NONE or NA) |
| Concurrent: | Programs that can be run concurrent with this |
| Following: | Programs that can only be run after this finishes |
| Driver files: | External files containing metadata to control program execution |
| Infiles: | Perm files USED by this program |
| Outfiles: | Perm files CREATED by this program |
| Macros Called: | Perm macros called by this program |

When the above information is entered into a program header, it can be accessed, by another program as used as metadata.

To run the program in this paper you must tell the program: 1) where to put the file that the program creates containing information about variables in tables 2) where to put the pdf that holds documentation – headers and comments 3) what directories to search. This is done by modifying the code below that is taken from the program.





```
/*destination for PDF of info about variables*/
%let SendPDFTo =E:\SASHeaders\ReportOnSASFiles.pdf;
%put &SendPDFTo;

/*where to put the PDF of documentation- the Headers and comments*/
%let SendDocTo =E:\SASHeaders\DocumentationOnProgs.pdf;
%put &SendDocTo;
data a_Dirs2Search;

/*where to search*/
 infile datalines truncover firstobs=3;
input @1 Dir2Search $char30.;
call symput("DirNo"||left(Put(_n_,4.)) , strip(Dir2Search));
call symput("NDirs" , left(put(_n_,4.0)));
datalines;
12345678901234567890123456789O
Note: one row (E:\SASHeaders) would produce the same results as these three
rows
F:\SASHeaders\Prod
F:\SASHeaders\Dev
F:\SASHeaders\Users
;run;
```

Of course, since you have the program, you can modify the header and then modify program to meet your needs.

It should be pointed out that the reports that are shown below are created by querying one master table. It was thought that having reports, that a manager might use frequently, created by the program would be useful. However, it is expected that many users would modify the program to produce the one master table and run their own queries as desired.

Below is one of the tables produced (named Final_all_sas_progs). It contains metadata about the all programs ***no matter if their headers have issues (missing values) or not***.

"No Of Items Missing" tells how many header items were missing (or incorrect) and "Abbrevs_of_header_sections" give information about what specific items are missing. These fields can be used by a manger to insure that headers are being filled out by all programmers. Additionally, if the company has created a prod directory , the Fullpath and date fields can be used in queries to find what programs were put into prod in a certain period of time.
It is important that the manager enforce discipline in header creation and maintenance and this table facilitates that task. If a program has any items missing (No Of Items Missing GT 0) it should trigger a conversation with the responsible programmer so that the header is fixed.

| No Of Header Items Missing | Date Created _DOS | LastSave Date_DOS | LastAccess Date_DOS | Owner_ DOS | Active_ Inactive _AdHoc | Programmer From Header | Abbrevs_of_ MISSING header_sections. These abbreviations are presented in the same order that they appear in the header, as an aid to interpretation | FullPath | Start Date From Header |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 1/10/2011 | Mon, Jan 10, 2011 | Sun, Jun 12, 2011 | Everyone | | | _Pr_Dv_Pr_PO_BO_DS_DP_MP_Cl_ St_____ Missing Program name, Dev/prod/adhoc flag. Programmer, progr am Owner etc. | F:\SASHeaders\Users \FrankD\Air\FranksF Ooling.sas | |
| 10 | 1/10/2011 | Mon, Jan 10, 2011 | Sun, Jun 12, 2011 | Everyone | | | _Pr_Dv_Pr_PO_BO_DS_DP_MP_Cl_ St_____ | F:\SASHeaders\Users \RussL\Shoes\Progra Ms\RussShoes.sas | |
| 3 | 1/8/2011 | Thu, Feb 10, 2011 | Sun, Jun 12, 2011 | Everyone | Active | RI | _____DS_____ ___Pr_____FC | F:\SASHeaders\Prod\ W_Print\Programs\W _Print.sas | |
| 2 | 1/10/2011 | Mon, Feb 7, 2011 | Sun, Jun 12, 2011 | Everyone | Dev | RI | _____DP_____ _____MC___ | F:\SASHeaders\Dev\ W_PrintD\Programs\ W_PrintD.sas | 7/17/2007 |
| 0 | 1/8/2011 | Mon, Feb 7, 2011 | Sun, Jun 12, 2011 | Everyone | Active | Elmer Fudd | _____ _____ The fact that there are no abbreviations in his string means that this program has all parts of the header filled in | F:\SASHeaders\Prod\I _Summary\Programs\ I_Summary.sas | 7/16/2002 |
| 0 | 1/10/201 | Mon, Feb 7, 2011 | Sun, Jun 12, 2011 | Everyone | Dev | RI | _____ _____ | F:\SASHeaders\Dev\ M_UnivariateD\Progra ms\M_UnivariateD.sa s | 7/16/1997 |
| 0 | 1/8/2011 | Mon, Feb 7, 2011 | Sun, Jun 12, 2011 | Everyone | Active | RI | _____ _____ | F:\SASHeaders\Prod\ M_Univariate\Progra ms\M_Univariate.sas | 7/16/2005 |

I leave it to you, as a SAS programmer to determine what other queries you could run against this table that would be helpful to a manager. The use of this table is not limited to the one task mentioned above.

Below is the table Final_c_all_progs_all_attributes. This contains parsed header information arranged in a format that facilitates querying. Please concentrate on the columns titled attribute and value. Since a program can have several source files (and several output files) this table was has many rows of data for each program (and each program header). Some people might describe this as "normalized data". Some of the data, like program name, repeats over several rows and some data (like attribute and value) are unique. By querying on the attribute and value fields, this table allows a manager to answer questions like: "What programs use this file as input?" and "What file creates this table?"

| Program | #_ of__ sections | Active | Programmer | Program Owner | Business Owner | Date Started | Date1st Into Prod | Recent Into Prod | Client | Study | Attribute | Value | Purpose | FullPath |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| _Summary.sas | 8 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | Preceding | SfirstProg.wsas | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 7 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | CONCURRENT | S_AllInOne.sas | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 6 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | CONCURRENT | S_together.sas | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 6 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | FOLLOWING | S_PostProcessing.sas | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 5 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | DRIVER FILES | S_SomeFile.xls | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 4 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | INFILES | sashelp.citimon | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 3 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | INFILES | sashelp.shoes | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 3 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | INFILES | Sashelp.air | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 3 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | OUTFILES | Sreports.SDSN_M_Univariate | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 2 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122 | OUTFILES | Sbogus.bogus | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 2 | Active | Elmer Fudd | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122122 | MACROS CALLED | S_BD2Age | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |
| _Summary.sas | 1 | Active | RI | Ping Programmer | Sam Summary | 7/16/2002 | 7/16/2002 | 7/16/2002 | X2 | 122122 | FORMATS CALLED | $S_Gndr. | a simple summary | F:\SASHeaders\Prod\I_Summary\Programs\I_Summary.sas |

The table below, named Final_progswmissinginfo, is an exception report on headers with problems. If a manager runs the header parser program daily, s/he can quickly find problems I headers and train the programmers to maintain the headers. It is a subset of Final_all_sas_progs, the first table described. Every program in this table has a header problem and the manager should contact the responsible programmer to have the header fixed.

| #_of __sections | Date Created _DOS | Last Save Date _DOS | Last Access Date _DOS | Owner _DOS | Active _Inactive _AdHoc | Programmer From Header | Abbrevs_of_header_sections | FullPath | Start Date From Header |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 1/10/2011 | Sun, Jun 12, 2011 | Sun, Jun 12, 2011 | Everyone | | | _Pr_Dv_Pr_PO_BO_DS_ DP_MP _Cl_St_____ _____ _____ | F:\SASHeaders\Users\FrankD\Air\FranksFooling.sas | |
| 10 | 1/10/2011 | Sun, Jun 12, 2011 | Sun, Jun 12, 2011 | Everyone | | | _Pr_Dv_Pr_PO_BO_DS_ DP_MP _Cl_St_____ _____ _____ | F:\SASHeaders\Users\RussL\Shoes\Programs\RussShoes.sas | |
| 3 | 1/8/2011 | Sun, Jun 12, 2011 | Sun, Jun 12, 2011 | Everyone | Active | RI | _____DS__ _____ _____Pr_____ _____ _FC | F:\SASHeaders\Prod\W_Print\Programs\W_Print.sas | |
| 2 | 1/10/2011 | Sun, Jun 12, 2011 | Sun, Jun 12, 2011 | Everyone | Dev | RI | _____D P_____ _____ ____M C___ | F:\SASHeaders\Dev\W_PrintD\Programs\W_PrintD.sas | 7/17/2007 |

The table below, Final_all_sas_tables_w_vars, contain much the same information as is in the dictionary tables but is for ***ALL*** files under a top level directory and those files will be found regardless of whether a libref points to the directory holding the file. This is useful for finding "lost tables".

| ib_Member_Name | Data_Set_Label | Obs_in_Data_Set | Create_Date | Last_Mod_Date |
|---|---|---|---|---|
| CITIMON | | 14 | 08JAN11:14:42:01 | 08JAN11:14:42:01 |
| DSN_WPRINT1 | | 19 | 08JAN11:14:41:46 | 08JAN11:14:41:46 |
| DSN_WPRINT2 | | 19 | 08JAN11:14:41:46 | 08JAN11:14:41:46 |
| DSN_WPRINT3 | | 19 | 08JAN11:14:41:46 | 08JAN11:14:41:46 |
| DSN_WPRINT4 | | 0 | 08JAN11:14:41:46 | 08JAN11:14:41:46 |

The file Final_all_all_contents_w_vars is too wide to allow a few sample obs to be shown in this proceedings paper– even in landscape format. The table is simply too wide to fit on this page and so will be described (see right).

It is much like the dictionary table SASHelp.vcolumn but, like the output above, this table is not dependent on having a libref that points to the directory.

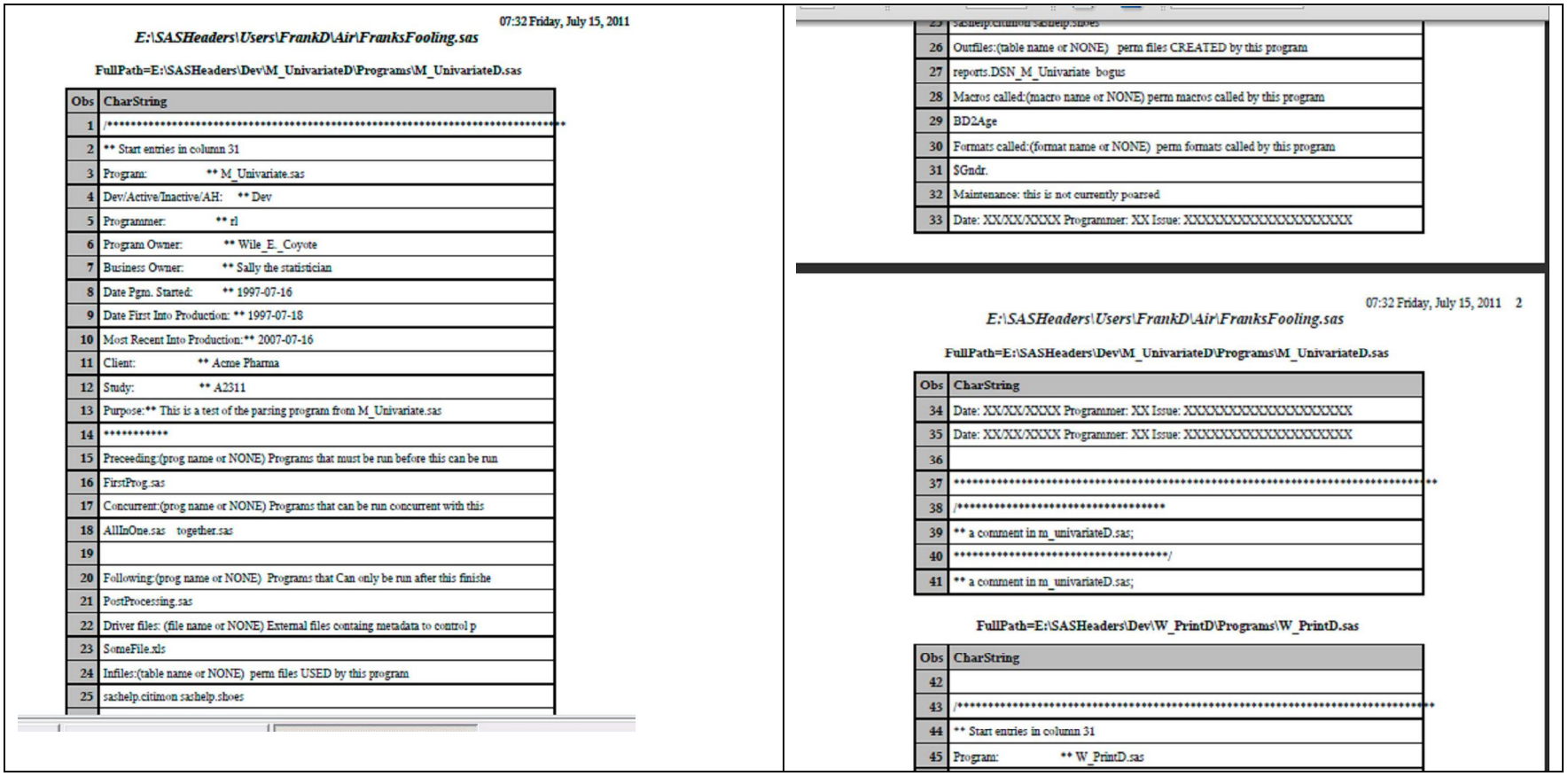
The table shows every variable in every file under the "top level directories that the user specified. It can be used to determine who needs to learn about proposed changes to variable attributes or to check that variable attributes are consistent across all files.

If one knew of a proposed variable change, one could search this file and find the names of tables that contain the variable. Then one could search in Final_c_all_progs_all_attributes and find all programs that used those files – and send out, to responsible programmers, a warning of a variable change and a request to check a list of programs

| # | Variable | Type | Len | Format | Label |
|---|---|---|---|---|---|
| 1 | MEMNAME | Char | 32 | | Lib_Member_Name |
| 2 | MEMLABEL | Char | 256 | | Data_Set_Label |
| 3 | TYPEMEM | Char | 8 | | Spec_Data_Set_Type |
| 4 | NAME | Char | 32 | | Var_Name |
| 5 | TYPE | Num | 8 | | Variable_Type |
| 6 | LENGTH | Num | 8 | | Var_Length |
| 7 | LABEL | Char | 256 | | Var_Label |
| 8 | FORMAT | Char | 32 | | Var_Format |
| 9 | INFORMAT | Char | 32 | | Var_Informat |
| 10 | NPOS | Num | 8 | | Position_in_Buffer |
| 11 | NOBS | Num | 8 | | Obs_in_Data_Set |
| 12 | CRDATE | Num | 8 | DATETIME16. | Create_Date |
| 13 | MODATE | Num | 8 | DATETIME16. | Last_Mod_Date |

Finally, the program will create documentation on the program logic using the header and comments in the program. I

f the comments are useful and describe the steps in the program this can be effective documentation – a logic overview that is available without reading the program details. The output produced by the program can be seen below, lines 1 to 37 are the program header. A “break for a section of code” comment is shown in lines 38 to 40. A one-line comment is shown in line 41. Using “break for a section of code” type comments to show the overall structure and one-line comments to explain details will allow this program to automatically create useful documentation. Please see the early section titled “AN ABBREVIATED HEADER – WITH EXAMPLES OF STANDARD COMMENTS:” for an illustration of how to use “break for a section of code” comments and one-line comments to make the logic of a program easy to understand.



07:32 Friday, July 15, 2011

*E:\SASHeaders\Users\FrankD\Air\FranksFooling.sas*

**FullPath=E:\SASHeaders\Dev\M_UnivariateD\Programs\M_UnivariateD.sas**

| Obs | CharString |
|---|---|
| 1 | /********** |
| 2 | ** Start entries in column 31 |
| 3 | Program: ** M_Univariate.sas |
| 4 | Dev/Active/Inactive/AH: ** Dev |
| 5 | Programmer: ** rl |
| 6 | Program Owner: ** Wile_E._Coyote |
| 7 | Business Owner: ** Sally the statistician |
| 8 | Date Pgm. Started: ** 1997-07-16 |
| 9 | Date First Into Production: ** 1997-07-18 |
| 10 | Most Recent Into Production:** 2007-07-16 |
| 11 | Client: ** Acme Pharma |
| 12 | Study: ** A2311 |
| 13 | Purpose:** This is a test of the parsing program from M_Univariate.sas |
| 14 | ********** |
| 15 | Preceeding:(prog name or NONE) Programs that must be run before this can be run |
| 16 | FirstProg.sas |
| 17 | Concurrent:(prog name or NONE) Programs that can be run concurrent with this |
| 18 | AllInOne.sas together.sas |
| 19 | |
| 20 | Following:(prog name or NONE) Programs that Can only be run after this finishe |
| 21 | PostProcessing.sas |
| 22 | Driver files: (file name or NONE) External files containg metadata to control p |
| 23 | SomeFile.xls |
| 24 | Infiles:(table name or NONE) perm files USED by this program |
| 25 | sashelp.citimon sashelp.shoes |
| 26 | Outfiles:(table name or NONE) perm files CREATED by this program |
| 27 | reports.DSN_M_Univariate bogus |
| 28 | Macros called:(macro name or NONE) perm macros called by this program |
| 29 | BD2Age |
| 30 | Formats called:(format name or NONE) perm formats called by this program |
| 31 | $Gndr. |
| 32 | Maintenance: this is not currently poarsed |
| 33 | Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX |

07:32 Friday, July 15, 2011 2

*E:\SASHeaders\Users\FrankD\Air\FranksFooling.sas*

**FullPath=E:\SASHeaders\Dev\M_UnivariateD\Programs\M_UnivariateD.sas**

| Obs | CharString |
|---|---|
| 34 | Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX |
| 35 | Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX |
| 36 | |
| 37 | ********** |
| 38 | /********** |
| 39 | ** a comment in m_univariateD.sas; |
| 40 | **********/ |
| 41 | ** a comment in m_univariateD.sas; |

**FullPath=E:\SASHeaders\Dev\W_PrintD\Programs\W_PrintD.sas**

| Obs | CharString |
|---|---|
| 42 | |
| 43 | /********** |
| 44 | ** Start entries in column 31 |
| 45 | Program: ** W_PrintD.sas |

## SUMMARY:

Some companies might already have a tracking system where programmers log data against projects. The program in this paper might
be an adjunct to such systems. It will be a useful tool for those companies who do not have a tracking system.

## REFERENCES:

Thanks to Frank Dilorio for sharing his ideas and code – on this and many other occasions – with me an with
others in the SAS community.

## CONTACT INFORMATION:

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

Russ Lavery
Bryn Mawr, PA
Email: Russ.Lavery@verizon.net
Web page for all papers: Russ lavery.com



---------- APPENDIX: Contains program code ----------

```
/************************ ******************************************************
** Start entries after the **
Program: ** headerreaderparserv4.sas
Dev/Active/Inactive/AH: ** Dev
Programmer: ** rl
Program Owner: ** rl, fd & The SAS community
Business Owner: ** rl, fd & The SAS community
Date Pgm. Started: ** Jan 12 2011
Date First Into Production: ** Dev
Most Recent Into Production:** Dev
Client: ** The SAS community
Study: ** n/A
Purpose: Allow a manager to collect info. about programming activity
***********
Preceeding:(prog name or NONE) Programs that must be run before this can be run (prog name
or NONE)
None
Concurrent:(prog name or NONE) Programs that can be run concurrent with this
None
Following:(prog name or NONE) Programs that Can only be run after this finishes
None
Driver files: (file name or NONE) External files containing metadata to control program
execution
internal
Infiles:(table name or NONE) perm files USED by this program
None
Outfiles:(table name or NONE) perm files CREATED by this program
None
Macros called:(macro name or NONE) perm macros called by this program
None
Formats called:(format name or NONE) perm formats called by this program
None
Modifications: this is not currently parsed
```

```
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Date: XX/XX/XXXX Programmer: XX Issue: XXXXXXXXXXXXXXXXXXX
Instructions:
check the following files. Export to Excel use Excel Filters.
WORK.Final_ProgsWMissingInfo
- look here to find files that have missing header info - any program here needs fixing
WORK.Final_All_SAS_Progs
- Query dates in this file to see programs that should have modification note
WORK.Final_All_SAS_Tables_W_Vars
- Query this if you are looking for a table that you have "lost"
options nocenter mprint mlogic source source2 mcompilenote=all;
*******************************************************************************/

/* Table of Contents
Section A__: Manual data entry is done in this section.
Enter the dirs to search and where to send the PDF file
Section B_: Loop over the dirs defined in section A.
Find the SAS programs and SAS Tables under directories entered in Sct. A
Create Macro arrays for looping in Section C
Create a format to control where changes are made to the variable LookFor
Section C_: Check the SAS Programs for Missing header information
Section D_: Check the SAS Files/Tables
Section E_: Merge files from different sections to make final version of the files -
*/

/*******************************************************************************
Section A__: Manual data entry in this section - Enter directories to search
: Create a file of top level dirs.
the program will search these dirs and dirs under them for .sas programs
*******************************************************************************/
/*destination for PDF of info about variables*/
%let SendPDFTo =E:\SASHeaders\ReportOnSASFiles.pdf;
%put &SendPDFTo;
/*where to put the PDF of documentatoin- the Headers and comments*/
%let SendDocTo =E:\SASHeaders\DocumentationOnProgs.pdf;
%put &SendDocTo;

data a_Dirs2Search;
  infile datalines truncover firstobs=3;
  input @1 Dir2Search $char30.;
  call symput("DirNo"||left(Put(_n_,4.)) , strip(Dir2Search));
  call symput("NDirs" , left(put(_n_,4.0)));
datalines;
12345678901234567890123456789O
Note: one row (E:\SASHeaders) would produce the same results as these three rows
E:\SASHeaders\Prod
E:\SASHeaders\Dev
E:\SASHeaders\Users
;
run;

proc print data=sashelp.vmacro;
  title "Check the macro variables we just created";
  where upcase(name)in: ("PROGNO","NPROGS","DIRNO","NDIRS" );
run;
  title "";

/*******************************************************************************
Section B_: Search the dirs defined in section A. Assemble files from different dirs into
a table
*******************************************************************************/
options mcompilenote=all symbolgen MPRINT SOURCE SOURCE2 sPOOL;
%macro LoopOverDirs(DOSDateSwitch=TC, DebugYN=Y); /*Loop and read the program headers*/
  /*Loop over all the dirs and find paths to .sas files under the top level dir*/
  /*After each loop, append the files from this loop into b_AllProgs2Check */
  %local LoopN DbgFlg;
    %if %upcase(&DebugYN) = Y %then
      %do;
          %let DbgFlg= ;
      %end;
    %else
      %do;
          %let DbgFlg=*;
```

```
      %end;
  %put DbgFlg = &DbgFlg;
  /*%let i=1; %LET DOSDateSwitch=tc; %let DbgFlg= ; */
  %local LoopN ;

  %do LoopN = 1 %to &NDirs;
     /*below is the key bit of code - it does a recursive search for files*/
    filename macs pipe "dir ""&&DirNo&LoopN...\*.sas"" /S /Q /&DOSDateSwitch.";

    data b_&DOSDateSwitch._ProgsInDir&LoopN (drop= path BigString ProgOrTable TableName)
    b_&DOSDateSwitch._Tables&LoopN (drop= path BigString ProgOrTable ProgName);
    length Fullpath path $ 120 ProgOrTable $32;
    retain path ;
    format &DOSDateSwitch._DateSAS Weekdate17.;

    infile macs truncover;
    input @1 BigString $120.;

   if bigstring =: "Directory of" then
      do;
       path=strip(substr(BigString,14,95));
       &DbgFlg put BigString=;
       &DbgFlg put path=;
      end;
    if substr(bigstring,3,1)="/" then
      do;
       &DOSDateSwitch._DateCH =substr(bigstring,1,10);
       &DOSDateSwitch._DateSAS =input(&DOSDateSwitch._DateCH,ANYDTDTE10.);
       Owner =substr(bigstring,41,15);
       ProgOrTable =substr(bigstring,63,42);
      end;
   If Reverse(Upcase(substr(left(reverse(ProgOrTable)),1,3)))="SAS"
       then ProgName=ProgOrTable;
    Else If Reverse(Upcase(substr(left(reverse(ProgOrTable)),1,8)))="SAS7BDAT"
       then TableName=ProgOrTable;

    FullPath=Strip(path)||"\"||strip(ProgName)||Strip(TableName);
    If upcase(substr(left(reverse(ProgOrTable)),1,3))="SAS"
       then output b_&DOSDateSwitch._ProgsInDir&LoopN;
    Else If Reverse(Upcase(substr(left(reverse(ProgOrTable)),1,8)))="SAS7BDAT"
       then output b_&DOSDateSwitch._Tables&LoopN;
   run;


   Proc Append base=b_&DOSDateSwitch._AllProgs2Check
     data=b_&DOSDateSwitch._ProgsInDir&LoopN;
   quit;


    Proc Append base=b_&DOSDateSwitch._AllTables2Check
     data=b_&DOSDateSwitch._Tables&LoopN;
    quit;
 %end;
%mend LoopOverDirs;

%LoopOverDirs(DOSDateSwitch=TC); /*date of Creation*/
%LoopOverDirs(DOSDateSwitch=TW); /*date of Modification*/
%LoopOverDirs(DOSDateSwitch=TA); /*date of Last Access*/

/*merge the three types of date info from the three files*/
Proc SQL;
Create table b_AllProgs2Check as
  select TC.ProgName
  ,TC_DateCH as DateCreated_Char ,TC_DateSAS as DateCreated_SAS
  ,TW_DateCH as DateSaved_Char ,TW_DateSAS as DateSaved_SAS
  ,TA_DateCH as DateAccessed_Char ,TA_DateSAS as DateAccessed_SAS
  ,TC.fullpath
  ,TC.owner
  from (b_TC_AllProgs2Check as TC
        inner join
        b_TW_AllProgs2Check as TW
        on tc.fullpath=tW.fullpath
        )
  inner join
```

```
        b_TA_AllProgs2Check as TA
        on tc.fullpath=ta.fullpath;
  quit;


  Proc SQL;
    Create table b_AllTables2Check as
    select TC.TableName
    ,TC_DateCH as DateCreated_Char ,TC_DateSAS as DateCreated_SAS
    ,TW_DateCH as DateSaved_Char ,TW_DateSAS as DateSaved_SAS
    ,TA_DateCH as DateAccessed_Char ,TA_DateSAS as DateAccessed_SAS
    ,TC.fullpath
    ,TC.owner
    from (b_TC_AllTables2Check as TC
         inner join
         b_TW_AllTables2Check as TW
         on tc.fullpath=tW.fullpath
         )
        inner join
        b_TA_AllTables2Check as TA
       on tc.fullpath=ta.fullpath;
  quit;

 /*Put programs, full paths and prog names, into macro array*/
data _null_;
  set b_AllProgs2Check Nobs=obs end=eof;
  call symput("ProgNo"||left(Put(_n_,4.)) , left(FullPath));
  if EOF then call symput("NProgs" , left(put(obs,4.0)));
 run;

/*%put _user_;*/
/*Put Tables, full paths and Table names, into macro array*/
data _null_;
  set b_AllTables2Check Nobs=obs end=eof;
  call symput("FileNo"||left(Put(_n_,4.)) , left(FullPath));
  if EOF then call symput("NFiles" , left(put(obs,4.0)));
run;


/*As QC, Lets print what we are about to check*/
proc sort data=sashelp.vmacro
  out=b_OurMacroVars;
  by name;
  where upcase(name)in: ("PROGNO","NPROGS","DIRNO","NDIRS","FILENO","NFILES" );
 run;

proc print data=b_OurMacroVars;
  title "These macro variables show the programs and files we will check";
run;
  title "";

proc format ;
  /*We Will use (modify) a long string to montor if the data was entered or not*/
  /*This format tells us where in the string "LookFor" we should make the modification*/
  /* 1 2 3 4 5 6 */
  /* 12345678901234567890123456789012345678901234567890123456789012345678901234567890 */
  /* |----- hardcode these changes---|--change w format-------| */
  /*Lookfor = "_Pr_Dv_Pr_PO_BO_DS_DP_MP_Cl_St_Pu_Pr_Co_Fo_DF_In_Of_MC_FC"; */
  /*Use an INformat to convert a string to its corresponding number position in in the
Lookfor string*/
  InValue ChrNbr
  "PRECEEDING"    =34
  "CONCURRENT"    =37
  "FOLLOWING"     =40
  "DRIVER FILES"  =43
  "INFILES"       =46
  "OUTFILES"      =49
  "MACROS CALLED" =52
  "FORMATS CALLED"=55;
run;

/*************************************************************************
Section C_: Loop over the PROGRAMS found and check for header characteristics
We also bring in some info from the DOS DIR command to reduce the chance of a programmer
```

```
scamming the header
reports
*******************************************************************************/
%macro ProgLoop(DebugYN=Y); /*Loop and read the program headers*/
 %local LoopN DbgFlg;
   %if %upcase(&DebugYN) = Y %then
        %do;
          %let DbgFlg= ;
        %end;
  %else
        %do;
          %let DbgFlg=*;
        %end;
%put DbgFlg = &DbgFlg;



%do LoopN = 1 %to &NProgs;
  data c_Parsed_Prog&LoopN(drop=CharString SourceString Counter )
    c_MissingInfo&LoopN(keep=FullPath Lookfor Active Programmer DateStarted MissingInfo)
    c_HeaderNComments&LoopN(keep=FullPath charstring );

   length program $45  MissingInfo 8.   Active $45   Programmer $45   ProgramOwner $45
          BusinessOwner $45   DateStarted $10   Date1stIntoProd $10   RecentIntoProd $10
          Client $45   study $45   Attribute $20   Value $70   Purpose $60  Lookfor $57
          FullPath $70  ;

   Retain program  Active  Programmer  ProgramOwner  BusinessOwner  DateStarted
          Date1stIntoProd  RecentIntoProd Study Client Purpose FullPath MissingInfo
          /*Number of values we should find - subtract 1 for each found*/
          Lookfor
         /*ID number of Header sections - convert to ___ when the section is processed*/
          InHeaderYN /*A flag that helps us decide if we are in the header or not*/
          ;

   *File "&SendTxtTo";
   FullPath="&&ProgNo&LoopN"; /*Put the program path in the output file*/

  if _n_=1 then InHeaderYN="Y";
   /*Lookfor holds an abbreviation for each of the Header items for which we must check*/
  if _n_=1 then Lookfor = "_Pr_Dv_Pr_PO_BO_DS_DP_MP_Cl_St_Pu_Pr_Co_Fo_DF_In_Of_MC_FC";
  label Lookfor = "Abbrevs_of_header_sections";
  /*Above are the First letters of Header sections -All lower case- convert to uppercase
  when the  section is found*/

  If _n_=1 then MissingInfo = 19 ;
  label MissingInfo = "#_of__sections";
  infile "&&ProgNo&LoopN" truncover;

  input @1 CharString $char79.;

  /*output any line in the header*/
  If InHeaderYN="Y" then output c_HeaderNComments&LoopN;

 /*This information (the var and associated value) are on ONE line*/
 /*Read and retain these values - they do not trigger an output statement*/
 If upcase(substr(charstring,1,8))= "PROGRAM:" then
   do; /*Criteria 1*/
     program=strip(substr(charstring,31,45 ));
     if length(program) GT 1 then
       do;/*We found a correctly entered header section- remove the abbrev from Lookfor*/
         MissingInfo = MissingInfo-1 ; /*reduce the count of "missing items" by one*/
         Substr(Lookfor,1,3)="___" ; /*Remove the abbrev from LookFor*/
       end;
   end;

  Else If upcase(substr(charstring,1,23))= "DEV/ACTIVE/INACTIVE/AH:" then
    do; /*Criteria 2*/
      Active=strip(substr(charstring,31,45 ));
       if length(Active) GT 1 then
         do;/*We found a correctly entered header section*/
           MissingInfo = MissingInfo-1 ;
           Substr(Lookfor,4,3)="___" ;
```

```
         end;
    end;
   Else If upcase(substr(charstring,1,11))= "PROGRAMMER:" then
     do; /*Criteria 3*/
       Programmer=strip(substr(charstring,31,45 ));
       if length(Programmer) GT 1 then
          do;/*We found a correctly entered header section*/
             MissingInfo = MissingInfo-1;
             Substr(Lookfor,7,3)="___" ;
          end;
    end;
  Else If upcase(substr(charstring,1,14))= "PROGRAM OWNER:" then
    do; /*Criteria 4*/
       ProgramOwner=strip(substr(charstring,31,45 ));
        if length(ProgramOwner) GT 1 then
           do; /*We found a correctly entered header section*/
              MissingInfo = MissingInfo-1;
              Substr(Lookfor,10,3)="___" ;
           end;
    end;
  Else If upcase(substr(charstring,1,15))= "BUSINESS OWNER:" then
    do; /*Criteria 5*/
       BusinessOwner=strip(substr(charstring,31,45 ));
       if length(BusinessOwner) GT 1 then
          do; /*We found a correctly entered header section*/
             MissingInfo = MissingInfo-1;
             Substr(Lookfor,13,3)="___" ;
          end;
    end;
  Else If upcase(substr(charstring,1,18))= "DATE PGM. STARTED:" then
    do; /*Criteria 6*/
        DateStarted=strip(substr(charstring,31,11 ));
        if length(DateStarted) GT 1 then
           do; /*We found a correctly entered header section*/
             MissingInfo = MissingInfo-1 ;
             Substr(Lookfor,16,3)="___" ;
           end;
    end;
Else If upcase(substr(charstring,1,27))= "DATE FIRST INTO PRODUCTION:" then
     do; /*Criteria 7*/
       Date1stIntoProd=strip(substr(charstring,31,11 ));
       if length(Date1stIntoProd) GT 1 then
          do; /*We found a correctly entered header section*/
            MissingInfo = MissingInfo-1 ;
            Substr(Lookfor,19,3)="___" ;
          end;
     end;
 Else If upcase(substr(charstring,1,28))= "MOST RECENT INTO PRODUCTION:" then
     do; /*Criteria 8*/
        RecentIntoProd=strip(substr(charstring,31,11 ));
        if length(RecentIntoProd) GT 1 then
          do; /*We found a correctly entered header section*/
            MissingInfo = MissingInfo-1;
            Substr(Lookfor,22,3)="___" ;
          end;
     end;
Else If upcase(substr(charstring,1,7))= "CLIENT:" then
     do; /*Criteria 9*/
       Client=strip(substr(charstring,31,45 ));
       if length(Client) GT 1 then
         Do; /*We found a correctly entered header section*/
           MissingInfo = MissingInfo-1 ;
           Substr(Lookfor,25,3)="___" ;
         End;
     end;
Else If upcase(substr(charstring,1,6))= "STUDY:" then
     do; /*Criteria 10*/
       Study=strip(substr(charstring,31,45 ));
       if length(Study) GT 1 then
          Do; /*We found a correctly entered header section*/
            MissingInfo = MissingInfo-1 ;
            Substr(Lookfor,28,3)="___" ;
          End;
```

```
     End;
Else If upcase(substr(charstring,1,8))= "PURPOSE:" then
     do; /*Criteria 11*/
        Purpose=strip(substr(charstring,13,60 ));
        if length(Purpose) GT 1 then
        Do; /*We found a correctly entered header section*/
          MissingInfo = MissingInfo-1 ;
          Substr(Lookfor,31,3)="___" ;
        End;
     End;

/*Lines below also parse the header but will trigger an output statement*/
/*Below we read the information that can have multiple entries per line AND Mult lines*/
/*IN the header, a blank line must separate attribute sections */
If upcase(scan(charstring,1,":" ))
in: ("PRECEEDING","CONCURRENT","FOLLOWING","DRIVER FILES","INFILES","OUTFILES"
,"MACROS CALLED","FORMATS CALLED") then
  do; /*We need to parse when an attribute can have multiple lines and mult entries on a
       line*/
    Attribute = upcase(scan(charstring,1,":" ));
    put "Attribute is: " attribute=;
    FirstTimeFlag="Initialized ";

    input @1 SourceString $char79.; /*read next line */
    If InHeaderYN="Y" then
      do; /*output any line in the header*/
        CharString=SourceString;
        output c_HeaderNComments&LoopN;
      end;

    Do While (substr(SourceString,1,8) NE " "); /*Break on a blank line*/
         Counter=1;
         Do while(scan(SourceString,counter," ") NE "");
           Value =scan(SourceString,counter," ");
           &DbgFlg put value=;
           if length(value) GT 1 then output c_Parsed_Prog&LoopN;
           if length(value) GT 1 and FirstTimeFlag = "Initialized " then
              do; /*We found the FIRST entry in a correctly entered header section */
                MissingInfo = MissingInfo-1 ; /*We have one less missing piece of info*/
            /*determine what is the starting char for removing the abbreV from lookfor*/
            StartingCharNo = Input(Attribute,ChrNbr.);
            &DbgFlg put StartingCharNo=;
            *Char2Use = put(Attribute,$ChrNbr.);
            &DbgFlg put LookFor= ;
            substr(Lookfor,StartingCharNo, 3 )="___";
            &DbgFlg put LookFor=;
         End; /*end of do; We found the FIRST entry*/
      FirstTimeFlag="no longer first";
      counter+1;
    End; /*end of the Do while(scan(SourceString,counter," ") NE ""); */

  input @1 SourceString $char79.;
  End; /*end of Do While (substr(SourceString,1,8) NE " ");*/
End; /*end of We need to parse*/

 AfterPrinting="N";
 if InHeaderYN="N" and substr(charstring,1,2) IN ("**","/*") then
    do;
       OUTPUT C_headerncomments&LoopN;
       AfterPrinting="Y";
    end;

  if &LoopN=4 then
     do;
        put _n_= " ----------------------------";
        put InHeaderYN= afterprinting=;
        put charstring=;
     end;

/*The test for the end of the header is eight asterisks*/
   if substr(charstring,1,8)="********" then
     DO;
        OUTPUT c_MissingInfo&LoopN;
        InHeaderYN="N";
```

```
     END;
run;



  Proc Append Base=C_All_Progs
    Data=c_Parsed_Prog&LoopN;
  Quit;

  Proc Append Base=C_All_MissingInfo
    Data=c_MissingInfo&LoopN;;

  Proc append Base=C_All_HeaderNComments
   data= c_HeaderNComments&LoopN;
Quit;
%end; /*%do LoopN = 1 %to &NProgs;*/
%mend ProgLoop;

%ProgLoop(DebugYN=N);


/*************************************************************************
Section D_: Loop over the SAS TABLES found and collect information
I have worked for clients who lost track of SAS tables. This is for them.
Above, we had to check the DOS dates to prevent scamming data entry in the header
SAS remembers when a table was modified, so we do not need to use DOS to check this
*************************************************************************/
options symbolgen mprint;
%macro FileLoop; /*Create a file showing all files/tables and all their variables*/
        ods pdf file ="&SendPDFTo" StartPage=YES;
        run;

        %local LoopN Thisfile;
        %do LoopN = 1 %to &NFiles;
          %let ThisFile=&&FileNo&LoopN; /*trimming*/
          ods ProcLabel " contents for &&FileNo&LoopN";
          run;

          Title "&ThisFile";
          proc contents data= "&ThisFile" varnum
          out = D_Contents&LoopN(Keep=MemName MemLabel TypeMem Name Type length label
          format informat NPos Nobs CRDate MoDate );
        run;

        ods ProcLabel " 10 obs from &&FileNo&LoopN";
        proc print data= "&ThisFile" (obs=10);
        RUN;

        Proc Append Base=D_All_Contents_W_Vars
          Data=D_Contents&LoopN;
        run;
        Quit;
        %end;
   ods pdf Close;
   run;
%mend FileLoop;
%FileLoop;

%macro FileLoop2; /*Create a file showing all files/tables and all their variables*/
        ods pdf file ="&SendDocTo" StartPage=YES;
        run;

        %local LoopN Thisfile;
        %do LoopN = 1 %to &NProgs;
                %let ThisFile=&&ProgNo&LoopN; /*trimming*/
                ods ProcLabel " Documentation for &&ProgNo&LoopN";
                run;
                Title "&&ProgNo&LoopN";

                proc print data=C_headerncomments&LoopN ;
                RUN;
        %end;
        ods pdf Close;
        run;
```

```
%mend FileLoop2;
%FileLoop2;


proc sort data=C_all_headerncomments;
  by FullPath;
run;

/*print the documentation*/
ods pdf file ="&SendDocTo" StartPage=YES;
run;

proc print data=C_all_headerncomments;
  by FullPath;
run;
ods pdf Close;
run;

PROC DATASETS lib=work; /*Change so view in excel has one row for the var name*/
  modify D_All_Contents_W_Vars (label="labels are good for documentation" sortedby=name);
  label name ="Name of student";
  label CRDATE ="Create_Date";
  label FORMAT ="Var_Format";
  label INFORMAT ="Var_Informat";
  label LABEL ="Var_Label";
  label LENGTH ="Var_Length";
  label MEMLABEL ="Data_Set_Label";
  label MEMNAME ="Lib_Member_Name";
  label MODATE ="Last_Mod_Date";
  label NAME ="Var_Name";
  label NOBS ="Obs_in_Data_Set";
  label NPOS ="Position_in_Buffer";
  label TYPE ="Variable_Type";
  label TYPEMEM ="Spec_Data_Set_Type";
  quit;




/*****************************************************************
Section __E: Make Final tables in the section below
*****************************************************************/
/** reports on SAS programs ***********************************/
Proc SQL; /*Create a file of problem programs for manager to check*/
  /*Use this to check on all programs missing data or not*/
  Create table Final_ProgsWMissingInfo as
  select _sas.MissingInfo as NoOfItemsMissing
  ,DateCreated_Char as DateCreated_DOS
  ,DateSaved_SAS as LastSaveDate_DOS
  ,DateAccessed_SAS as LastAccessDate_DOS
  ,Owner as Owner_DOS
  ,_sas.Active as Active_Inactive_AdHoc
  ,_sas.Programmer as ProgrammerFromHeader
  ,_sas.Lookfor as LookHereforProblems
  ,_sas.FullPath
  ,_sas.DateStarted as StartDateFromHeader
   from C_All_MissingInfo as _sas
    left join
        B_allprogs2check as _dos /*Bring in info from the DOS DIR Command*/
    on _DOS.fullpath=_SAS.fullpath
    where NoOfItemsMissing GT 0
    order by MissingInfo desc ;
quit;

Proc SQL;
  /*Use the dates in the file below to see if a modification note should be in a header*/
  /*This has all programs- Query on DOS dates to see if a mod should be in the header*/
  Create table Final_All_SAS_Progs as
  select
  Max(0,_sas.MissingInfo) as NoOfItemsMissing
  ,DateCreated_Char as DateCreated_DOS
  ,DateSaved_SAS as LastSaveDate_DOS
```

```
  ,DateAccessed_SAS as LastAccessDate_DOS
  ,Owner as Owner_DOS
  ,_sas.Active as Active_Inactive_AdHoc
  ,_sas.Programmer as ProgrammerFromHeader
  ,_sas.Lookfor as LookHereforProblems
  ,_sas.FullPath
  ,_sas.DateStarted as StartDateFromHeader
  from C_All_MissingInfo as _sas
  Full join
       B_allprogs2check as _dos
   on _DOS.fullpath=_SAS.fullpath
  order by MissingInfo desc ;
quit;

Proc SQL;/*Create a table showing Every attribute in any headers*/
  Create table Final_C_All_Progs_All_Attributes
  as select * from C_All_Progs(drop= FirstTimeFlag StartingCharNo Lookfor);
  quit;


/** reports on SAS tables ***********************************/
/*I've been in places where tables / variables were lost*/
/* Final_All_SAS_Tables_W_Vars contains info on all tables and all variables*/
Proc SQL;
  /*Create a summary file showing SAS tables and variables in them - one row per table */
  Create table Final_All_SAS_Tables_W_Vars as
  select distinct MemName, MemLabel, Nobs, CRDate, MoDate
  from D_All_Contents_W_Vars;
  quit;

Proc SQL;
  /*This is different from the Dictionary tables.
  It shows all SAS tables - not just the ones with established "standard libraries" */
  create table Final_All_All_Contents_W_Vars as
  select * from D_all_contents_w_vars;
QUIT;

PROC DATASETS lib=work;
  modify Final_ProgsWMissingInfo (label="Good for Finding missed Sections");
  modify Final_All_SAS_Progs (label="Use DOS date 4 finding missed Mod. Notes");
  modify Final_C_All_Progs_All_Attributes (label="All attributes found in any SAS program
header");
  modify Final_All_SAS_Tables_W_Vars
     (label="Basic info on SAS tables one row per table");
  modify Final_All_All_Contents_W_Vars
     (label="Basic info on SAS tables one row per table-  variable");
  /*This program also produces a pdf showing proc contents and 10 obs for all SAS tables*/
quit;
```