

## Paper 74-25

**An Automated Method to Create a Descriptive Index for a Directory of SAS® Programs**

Gary Cunningham, Sanofi-Synthelabo Research, Malvern, PA

**ABSTRACT**

The methods presented in this paper can be used to significantly aid with a programming documentation effort by automating the process of creating a descriptive index for a directory of SAS programs. As referred to in this paper, a descriptive index is simply a listing of the name of each SAS program in a directory and its corresponding description.

Using a pipe to communicate between the SAS System and Windows NT, base SAS code can be used to create a list of all SAS programs in a directory, get each program's description as written in the program's standard header, and output an index that lists each program's name and its accompanying description.

This paper describes a SAS macro that uses the methods above to automate the creation of this index. The SAS macro was developed to run under Windows NT, in Release 6.12 of the SAS System for Windows. This paper is intended for users with an intermediate level of experience with the SAS language.

**INTRODUCTION**

The value of a well-documented program is widely acknowledged and cannot be overstated. Clear and concise comments that describe data manipulation steps, complex mathematical and statistical computations, and not-so-common coding techniques are essential to a well-developed and well-utilized program.

External program documentation is just as important as internal program documentation. One example of external program documentation is the focus of this paper: a descriptive index for a directory of SAS programs. A descriptive index lists the name of each SAS program in a directory and its corresponding function. It serves an important role in that it creates documentation that provides any user the ability to easily and quickly identify and locate programs of interest.

SAS macro PROGINDX was developed to completely automate the creation of this index, thereby saving a tremendous amount of time while significantly contributing to the programming documentation effort.

**MACRO OVERVIEW**

SAS macro PROGINDX uses base SAS code and some features available with the SAS System and Windows NT to automate the creation of a descriptive index for a directory of SAS programs. First, the macro creates a list of all of the SAS programs located in a directory. Second, for each program on the list, the macro scans the header of the program to get the program's description as written by the program developer. Finally, the macro outputs a text file that lists the names of the SAS programs in the directory and their corresponding descriptions.

The call to macro PROGINDX takes the following form,

```
%PROGINDX(dir=, exclude=, outfile=)
```

where

dir = the directory in which the SAS programs reside,

exclude = any SAS programs that should be excluded from the

index (by default, no programs are excluded), and

outfile = the name, including the path, of the external file to which the index should be written (by default, this file is named PROGINDX.TXT and is filed in the directory in which the SAS programs reside).

As an example, the macro call

```
%PROGINDX(dir=u:\drug1\prot1,
           exclude=junk garbage,
           outfile=u:\drug1\doc\progdoc.txt)
```

will generate an index for the SAS programs in directory U:\DRUG1\PROT1, excluding programs JUNK.SAS and GARBAGE.SAS. The index will be filed in directory U:\DRUG1\DOC as a text file named PROGDOC.TXT.

The next three sections describe in detail the primary steps taken by the macro to create the index.

**STEP 1: CREATE A LIST OF SAS PROGRAMS IN A DIRECTORY**

Macro PROGINDX first creates a list of all of the SAS programs in the user-specified directory. This is done by using a pipe.

As defined in the *SAS Companion for the Microsoft Windows Environment, Version 6, Second Edition*, a pipe "is a channel of communication between two processes." "With the SAS System and Windows NT, ... you can use a specialized Windows application to provide information to your SAS session or vice versa."

There are two types of pipes: named pipes and unnamed pipes. Unnamed pipes are also referred to simply as pipes. A discussion of the differences between the two is beyond the scope of this paper. See the *SAS Companion for the Microsoft Windows Environment* for more information.

"Unnamed pipes enable you to run a program outside the SAS System and redirect the program's input, output, and error messages to the SAS System. This capability enables you to capture data from a program external to the SAS System without creating an intermediate data file." Unnamed pipes, therefore, can be used to create a list of files in a directory.

To use an unnamed pipe in SAS, a FILENAME statement of the following form is issued,

```
FILENAME fileref PIPE 'program-name'
           option-list;
```

where

fileref is any valid fileref,

PIPE is the device-type keyword that tells the SAS System you want to use an unnamed pipe,

'program-name' specifies the external Windows application program, and

option-list can be any of the options valid in the FILENAME statement.

The following FILENAME statement in macro PROGINDX uses an unnamed pipe to create a list of all SAS programs in the directory denoted by SAS macro variable DIR:

```
filename getfiles pipe "dir/b &dir.\*.sas";
```

In this statement, 'program-name' is a DOS command, DIR/B, that instructs the system to list all files with the extension SAS in the directory denoted by SAS macro variable DIR. The /B qualifier tells the system to list the file names only, excluding any size, date, or time information.

This list of SAS programs can be put into a SAS data set by using SAS INPUT statements and an INFILE statement that references the FILENAME statement above. The SAS code to do this follows:

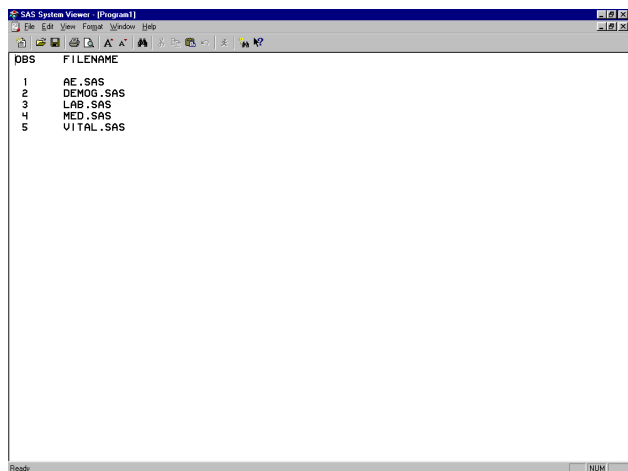
```
data filemams;
  infile getfiles length=1;
  length filename $200;
  input @;
  input @1 filename $varying200. 1;
  filename=uppercase(filename);
run;
```

The SAS code above represents a slight modification of the SAS sample program GETNAMES.SAS that is located in the Technical Support area of the SAS web site.

As an example, assume the following macro call was made:

```
%PROGINDX(dir=u:\drug1\prot1)
```

Assume that five SAS programs reside in directory U:\DRUG1\PROT1: AE.SAS, DEMOG.SAS, LAB.SAS, MED.SAS, and VITAL.SAS. Figure 1 below shows how SAS data set FILEMAMS would look in this example.



DBS	FILENAME
1	AE.SAS
2	DEMOG.SAS
3	LAB.SAS
4	MED.SAS
5	VITAL.SAS

**Figure 1** Data Set Containing List of SAS Programs

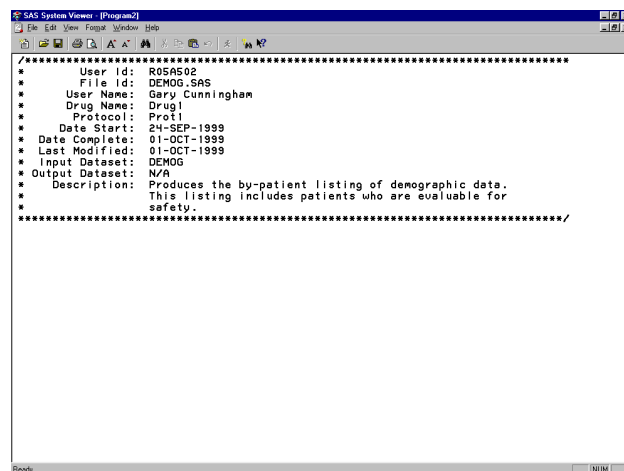
Quoted text in this section has been reprinted with permission of SAS Institute Inc. from the *SAS Companion for the Microsoft Windows Environment, Version 6, Second Edition*. Copyright 1996 by SAS Institute Inc.

Some SAS code presented in this section has been reprinted with permission of SAS Institute Inc. from SAS sample program GETNAMES.SAS located in the Technical Support area of the SAS web site.

## STEP 2: GET THE DESCRIPTION FOR EACH PROGRAM

After creating a list of all of the SAS programs located in the user-specified directory, the macro deletes programs from the list that have been designated by the user as exclusions. Then, for the remaining programs, macro PROGINDX reads in each of the files, searches the program header to find the program description, and retains the text that has been entered in the area designated for the program description. The macro therefore relies on each program having a standard program header and a concise, informative description written in the header by the program developer.

Figure 2 presents an example of the standard header for SAS program DEMOG.SAS. For purposes of this example, the executable SAS code has been removed.



```
*****
* User id: ROSAS02
* File id: DEMOG.SAS
* User Name: Gary Cunningham
* Drug Name: DRUG1
* Protocol: PROT1
* Date Start: 24-SEP-1999
* Date Complete: 01-OCT-1999
* Last Modified: 01-OCT-1999
* Input Dataset: DEMOG
* Output Dataset: N/A
* Description: Produces the by-patient listing of demographic data.
* This listing includes patients who are evaluable for
* safety.
*****
```

**Figure 2** Standard Program Header

Macro PROGINDX simply uses base SAS code – primarily DATA steps, character functions, %DO loops, and macro code – to perform the tasks mentioned above.

Specifically, for each of the SAS programs in the directory, the macro will input and examine the text of the program line-by-line until encountering the keyword "Description:" in the program header. Any and all text entered after the keyword description, but before the final program header line, will be retained by the macro as the description for that program. The text that was retained is placed into an intermediate SAS data set. Each of these data sets is appended in turn to another SAS data set that contains cumulative information from all of the SAS programs that have previously been processed.

Figure 3 below displays the intermediate data set formed after processing the header for program DEMOG.SAS.

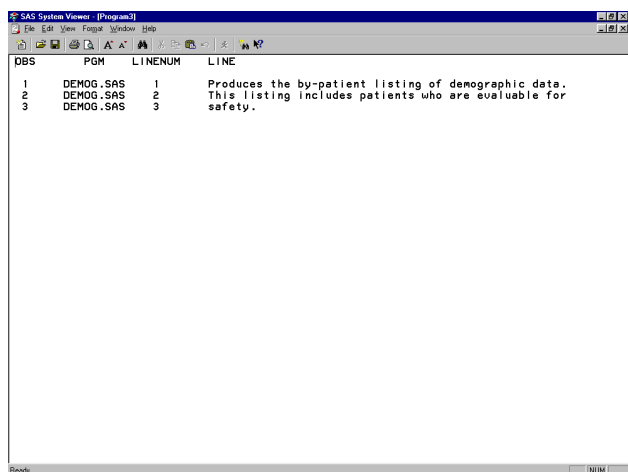


Figure 3 Data Set Containing a Single Program Description

Continuing with the example presented previously, Figure 4 below displays the cumulative data set formed after processing all of the programs in the directory.

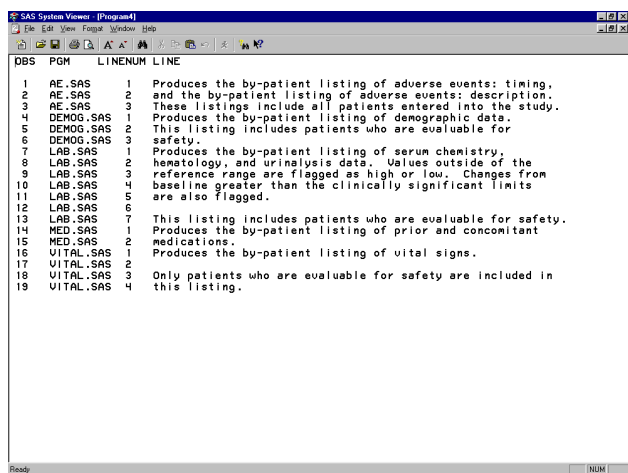


Figure 4 Data Set Containing Multiple Program Descriptions

**STEP 3: CREATE A DESCRIPTIVE INDEX**

After all of the SAS programs have been processed, the macro uses the REPORT procedure to format the listing and the PRINTTO procedure to output the index as a permanent text file. As mentioned previously, the macro allows the user to direct the output file to a specific directory and to name the file as desired.

Figure 5 presents the final output from the example macro call presented earlier. Note that titles on the listing include the directory in which the programs reside and the date and time that the SAS session was initiated. These titles are automatically generated by the macro.

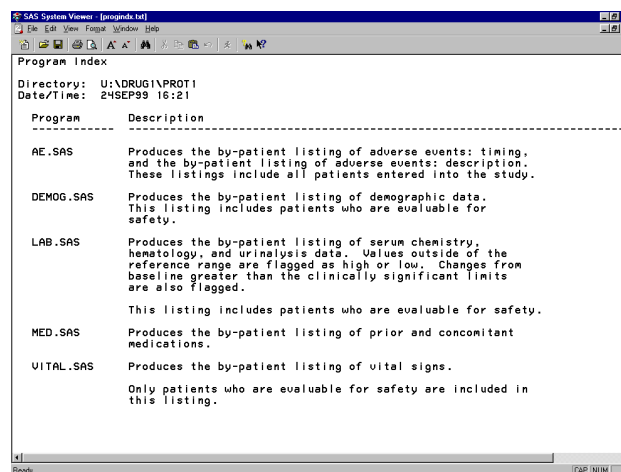


Figure 5 Final Output

**CONCLUSION**

This paper has shown how SAS macro PROGINDX uses some unique features available with the SAS System and Windows NT to completely automate the creation of a descriptive index for a directory of SAS programs. The index serves an important role in that it creates documentation that provides any user the ability to easily and quickly identify and locate programs of interest. This macro, therefore, is a time-saving tool that significantly contributes to the programming documentation effort overall.

Note, however, that while this macro is a valuable, time-saving tool, it will only be of value if the program developer has been diligent in writing a clear and concise description in the standard header for each program. The resulting program index will only be as useful and informative as the individual descriptions within the programs.

**REFERENCES**

SAS Institute Inc. (1996), *SAS® Companion for the Microsoft Windows Environment, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1998), GETNAMES.SAS [Computer program], Cary, NC: SAS Institute Inc.

**TRADEMARKS**

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

**CONTACT INFORMATION**

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

Gary Cunningham  
 Sanofi-Synthelabo Research  
 9 Great Valley Parkway  
 P.O. Box 3026  
 Malvern, PA 19355  
 (610) 889-6451  
 Gary.Cunningham@sanofi-synthelabo.com

## Appendix 1 Macro PROGINDX Code

```

%macro progindx(dir=,exclude=,outfile=);

options nodate nocenter number pageno=1 ps=50 ls=100;

title1 "Program Index";
title3 "Directory:  %upcase(&dir)";
title4 "Date/Time:  &sysdate &systemtime";

*****
*   Create a file reference for the output file.
*****
%if &outfile = %then %do;
    filename outfile "&dir.\progindx.txt";
%end;
%else %do;
    filename outfile "&outfile";
%end;

*****
*   Use a pipe to create a list of all SAS programs located in the user-
*   specified directory.
*****
filename getfiles pipe "dir/b &dir.\*.sas";

data filenams;
    infile getfiles length=1;
    length filename $200;
    input @;
    input @1 filename $varying200. 1;
    filename = upcase(filename);
run;

filename getfiles clear;

*****
*   Exclude any user-specified programs from the list of SAS programs created
*   above.
*****
%if &exclude ^= %then %do;
    %let morepgms = YES;
    %let pgmnum = 1;
    %let pgm = %upcase(%scan(&exclude,&pgmnum, ' '));

    %do %until(&morepgms = NO);
        data filenams;
            set filenams;
            if scan(filename,1, '.') = "&pgm" then delete;
        run;

        %let pgmnum = %eval(&pgmnum+1);
        %let pgm = %upcase(%scan(&exclude,&pgmnum, ' '));
        %if &pgm = %then %let morepgms = NO;
    %end;
%end;

*****
*   Determine the total number of SAS programs on the list.
*   For each of the programs, get the program description by retaining the text
*   entered after the keyword DESCRIPTION but before the final program header
*   line.
*****
data _null_;
    set filenams end=eof;
    if eof then call symput('total',left(_n_));
run;

data saspgms;
run;

%do i = 1 %to &total;
    %let pgm = ;

    data _null_;
        set filenams;
        if _n_ = &i;
        call symput('pgm',trim(filename));

```

```

run;

filename saspgm "&dir.\&pgm";

data saspgm2;
  length pgm $ 200;
  infile saspgm missover lrecl=200 pad;
  input @1 line $200.;
  retain descript colon linenum gotdesc 0;
  drop descript colon gotdesc i;
  pgm = "&pgm";
  if not gotdesc then do;
    if index(upcase(line),'DESCRIPTION:') > 0 then do;
      descript = 1;
      colon = index(line,':');
    end;
    if descript then do;
      if index(line,'***/') = 0 then do;
        line = trim(substr(line,index(line,'*')+1));
        if linenum = 0 then do;
          do i = 1 to colon-1;
            substr(line,i,1) = ' ';
          end;
        end;
        linenum = linenum +1;
        output;
      end;
    else gotdesc = 1;
  end;
end;
run;

data saspgms;
  set saspgms saspgm2;
  if _n_ = 1 and line = ' ' then delete;
  line = left(line);
run;

filename saspgm clear;
%end;

*****
*   Output the index of programs and delete work data sets.
*****;
proc sort data=saspgms;
  by pgm linenum;
run;

proc printto new print=outfile;
run;

proc report data=saspgms nowindows headskip split='?' missing;
  column pgm line;
  define pgm / order width=12 'Program?--';
  define line / display width=80 flow 'Description?--';
  break after pgm / skip;
run;

proc printto;
run;

proc datasets library=work memtype=data;
  delete filenams saspgms saspgm2;
quit;

title;

%mend progindx;

```