Sharing Your Tips and Tricks with Others.
Give Your Toolbox a Web Presence

John Charles Gober
Bureau of the Census, Demographic Surveys Methodology Division

INTRODUCTION

The purpose of this paper is to re-emphasize the importance of sharing knowledge, techniques, and ideas among your fellow programmers. It will hopefully put to rest the concept of hiding code for the sake of job protection and ego, and that the mutli-programmer office does benefit from shared technologies.

This paper will review three simple to intermediate pieces of **SAS**® code that will allow a user to easily and automatically convert their favorite programs, shortcuts, notes, and examples into html code and allow them to be placed in to an intranet toolbox that can be accessed by the entire office.

All of the code used in this paper can be found either in the **SAS Online Doc® for Version 8**, the **SAS Technical Support** web pages at www.sas.com, **SAS Guide to Proc Report**, and the **SAS Guide to the Output Delivery System®**. The idea itself to create the intranet toolbox came while taking the **SAS Core Concepts** training course. So in a way sharing knowledge has already paid off. The completed code for this application is on the last page of this paper. Please refer to it as needed along with a screen image of the actual application.

THE FILENAME STATEMENT

Everyone is familiar with the basic **FILENAME** statement with the implied **DISK** access method and possibly the technique of concatenating files using a single **FILENAME** statement. But the filename can also be used to read and write to **DUMMY** files used for testing your programs, send **EMAIL**, access files using **FTP** or **URL**, and create self deleting files with the **TEMP** access method.

The access method, though, that was most useful for creating the toolbox was the **PIPE** access method. The **PIPE** command allows the programmer to execute any command native to an operating system and have the results treated as a text file, which can be read directly into **SAS**. By using the **PIPE** method it was possible to read in the contents of the directory(s) containing the program names that were to be placed in the toolbox. Sample code for **Windows**® and **Sun Solaris**® is below.

```
/* SAMPLE FILENAME STATEMENT
   FOR WINDOWS USING PIPE*/
FILENAME IN1 PIPE "DIR
   C:\TEMP\EXAMPLES\*.SAS"
      LRECL=150;

/* SAMPLE FILENAME STATEMENT
   FOR SUN SOLARIS USING
   PIPE*/
FILENAME IN1 PIPE "LS
  /TEMP/EXAMPLES/*.SAS"
      LRECL=150;
```

Depending on the length of your directory paths and program names you may with to increase or decrease the LRECL= option accordingly.

Some of our users asked why the basic **FILENAME** statement using wildcards was not used for this application. The answer that was given out was that it could have been used just as easily, but the technique of using wildcards and concatenation is routine and would limit the possibility of learning something new. By using a different approach it was hoped that a new technique could be learned and programmer knowledge could be increased. The Sample code for a **FILENAME** statement with wildcards and concatenation is below. The same general logic applies for both Windows and Sun operating systems.

```
/* SAMPLE FILENAME WITH
     WILDCARD FOR WINDOWS*/
  FILENAME IN1
  "C:\TEMP\EXAMPLES\*.SAS,
   C:\TEMP\MACROS\*.SAS";
```

CAPTURING THE PROGRAM CODE

While the **FILENAME** statement above will allow the programmer to identify the names of the files containing all of the sample code there still exists the problem of actually reading in the program statements stored within the programs. This is easily accomplished by using two **INPUT** and **INFILE** statements, one which should be familiar and the other one possibly not so familiar. The first **INFILE** and **INPUT** statement reads the names of the program files containing our sample programs and code, which were identified and referenced by the **FILENAME PIPE** statement.

```
INFILE IN1 PAD MISSOVER;
INPUT FIL2READ $CHAR150. @;
```

Note the use of the use of the **PAD** and **MISSOVER** options on the **INFILE** statement. The **PAD** statement will cause the variable called **FIL2READ** to be filled with trailing blanks. On the Windows platform this option is needed because we do not know the actual length of the character string being returned by the **PIPE**. With out the **PAD** option **SAS** might not be able to determine the end of the record and you will receive the familiar "**SAS went to a new line when INPUT statement reached past the end of a line**" message. On other platforms it was found that the **MISSOVER** option was necessary to prevent **SAS** from reading the next line of information.

The next three statements used in the application is also operating system dependent. Since the system commands for listing the files inside of a directory performs differently on various platform the first program statement was needed to filter out any extraneous information, which did not contain the SAS programs, we wanted to process through the rest of the program.

```
IF INDEX(FIL2READ,'.SAS');
PGMNM =
   TRANWRD(
      SCAN(FIL2READ,6,
         ' '),'.SAS','');
PATHFILE =
   'C:\TEMP\EXAMPLES\'||
      TRIM(LEFT(PGMNM))||'.SAS';
```

The second statement was used to strip off the '.sas' suffix from the program name to create a variable for cosmetic and presentation purposes for the html pages. The third statement was needed to create a variable to reflect the complete path and filename of the file containing the sample program code. This was needed in order to pass the locations of the sample program files in to the second **INPUT** statement.

The second **INPUT** statement makes use of the **FILEVAR** option in order to sequentially read multiple files from a list or other aggregate source. There are numerous examples and variations of this technique in past proceedings of many of the **SESUG** and **SUGI** conferences papers as well as the **SAS** Technical Support pages on the web.

```
INFILE DUMMY FILEVAR=PATHFILE
END=DONE PAD MISSOVER;
   DO WHILE(NOT DONE);
     INPUT @001 LINEINFO
                    $CHAR150.;
     OUTPUT;
   END;
```

The use of the file reference of **DUMMY** is purely a matter of preference. The name could as well have been **IN1**, **IN2**, or **MYLIST**, etc. If you remember from above, the variable **PATHFILE** contains the complete path and file name of a program containing one of our sample programs. By allowing the **FILEVAR=** to be equated to this variable then through the use of a do loop the entire sample program can be read into the application.

By using the above method we have create a SAS dataset containing the variables **PGMNMN** containing the name of the sample program we wish to include in the intranet application, and **LINEINFO** which contains the actual code inside the sample. These are the only two variables needed. All others can be dropped.

THE OUTPUT DELIVERY SYSTEM

To the inexperienced or even the seasoned programmer the **ODS** may seem a little ominous and overwhelming and something not to be taken lightly, even to a **SAS** programmer of over 15 years. Fortunately there are only a few statements from the **ODS** that the intranet application needed and these were readily available and explained in the **SAS Guide to the Output Delivery System®** and the **PROC TEMPLATE FAQ** on the SAS Technical Support web pages.

```
ODS HTML
     FILE='BODY.HTM'
     CONTENTS='CONTENTS.HTM'
     FRAME='FRAME.HTM'
     PATH='C:\TEMP\DATA'
        (URL=NONE)
     NEWFILE=PAGE
     STYLE=TEST1;
```

When used in conjunction with the **PROC REPORT** procedure and the **PROC TEMPLATE** procedure, which will be explained in the next sections, the above statements tell SAS to generate an html page (body) for every sample program to be included in the intranet application. The above statement also tells **SAS** to generate a table of contents for all of the body pages and integrate the two into a frame environment. There are a few other **ODS** statements needed for the intranet applications but they will be listed in the complete program listing at the end of the paper.

PROC REPORT

PROC REPORT is another one of those procedures that can be made as simple or as complex as needed. For this application only a few of the basic statements are needed. These statements work in conjunction with the **ODS HTML** statements.

```
PROC REPORT DATA=ONE
     (KEEP=PGMNM LINEINFO) NOWD;
   BY PGMNM;
   TITLE1 "EXAMPLES";
RUN;
```

This straightforward code when executed with **ODS HTML** will create a separate web page for each unique occurrence of the value of the variable **PGMNM**. Again there are a few other statements, which have been added to the final program on the last page for increased aesthetics of the overall application.

PROC TEMPLATE

With each licensed copy of **SAS** there are supplied several basic templates that give the user the ability to choose a predefined look and feel to the html pages and contents. These templates give the users a predefined starting point when it comes to changing the look, feel, and behavior of their pages. To view the code for these templates the user needs only to go to the Results Window, click on View, Templates, sashelp.tmplmst. This application used the DEFAULT template with only a few minor modifications to Again only very minor modifications to one of the templates was needed to make the intranet application functional.

By far the most difficult aspect of this application was changing the default look and feel of the web pages generated by the ODS. Even though the basic changes worked, the appearance of the titles, the fonts, the program names in the table of contents, and the background colors needed to be changed. This took about ten hours of trial and error to complete the changes.

```
PROC TEMPLATE;
   DEFINE STYLE TEST1;
   PARENT=STYLES.DEFAULT;
     REPLACE CONTENTFOLDER /
       LISTENTRYANCHOR = ON;
       REPLACE COLOR_LIST /
         'BGA1' = CXE5C76B;
       REPLACE COLORS /
          'CONTENTBG' =
COLOR_LIST('BGA1')
         'DOCBG' =
COLOR_LIST('BGA1');
     REPLACE CONFOLDERFG /
          BACKGROUND =
COLORS('CONTENTBG');
       REPLACE TEXT /
          'CONTENT TITLE' =
'DSMD INTRANET TOOLBOX';
       REPLACE TABLE /
           CELLPADDING = 0
         CELLSPACING = 0;
   END;
RUN;
```

There were numerous other template statements that could be changed but the above statements were just enough to make the application just a little more pleasing.

CONCLUSION

It does not matter if you are an experienced programmer or a novice, or if you are the only programmer in your office or one of a hundred. There will always be a need for quicker and better ways of programming. By placing all of your samples, techniques, ideas, and standards in a central location then everyone in your office

can have access to a learning and efficiency toolbox that will make programming more productive and enjoyable.

CREDITS AND ACKNOWLEGEMENTS

Questions and comments concerning this paper can be directed to:

John Charles Gober
Bureau of the Census
FOB3 MS8700
Washington DC 20233-8700
John.charles.gober@census.gov

Sample code for the Intranet Toolbox

```
/*************************************************************/
/**Program Desc:   THIS IS AN HTML FRAME EXAMPLE             */
/*************************************************************/
 filename in1    pipe "dir c:\Temp\examples\*.sas" lrecl=150;

 data one;
    length fil2read name $ 150;
    length pgmnm $32;
    infile in1 pad missover;
    input fil2read $char150. @;
    if index(fil2read,'.sas');
    pgmnm = tranwrd(scan(fil2read,6,' '),'.sas','');
    pathfile = 'c:\Temp\examples\'||trim(left(pgmnm))||'.sas';
```

```
    infile dummy filevar=pathfile end=done pad missover;
    do while(not done);
      input @001 lineinfo $char150.;
      output;
    end;
run;

proc sort data=one;
   by pgmnm;
run;

proc template;
   define style test1;
   parent=styles.default;
     replace ContentFolder /
       listentryanchor = on;
         replace color_list /
           'bgA1' = cxE5C76B;
         replace colors /
            'contentbg' = color_list('bgA1')
         'docbg' = color_list('bgA1');
     replace Confolderfg /
            background = colors('contentbg');
         replace text /
            'Content Title' = 'DSMD IntraNet Toolbox';
         replace Table /
             cellpadding = 0
         cellspacing = 0;
   end;
run;

ods listing close;
ods html file='body.htm'
         style=test1
         contents='contents.htm'
         frame='frame.htm'
         path='c:\temp\data'(url=none)
         newfile=page;

options nobyline;
title1;

ods proclabel 'Samples and Ideas';

proc report data=one
              (keep=pgmnm lineinfo) nowd contents='' noheader
              style(column)=[font_face=Courier
                             font_size=2
                             asis=on];
   by pgmnm ;
   column lineinfo;
   define lineinfo /'Program Code' left;
   label pgmnm='Pgm';
   title1 "EXAMPLES";
run;

ods html close;
```

```
ods listing;

***** end of program *****
```

Sample of the DSMD Intranet Toolbox