Paper AD09

Multiple File Processing with SAS®

Kevin McGowan, Constella Group, Durham NC

ABSTRACT

Sometimes there is a need to process multiple files with one program. In some cases the number of files that need processing is not known in advance, just the location of the files is known. By using the SAS macro language along with data steps and SAS functions such as dopen, dnum, dread, and dclose it is possible to process multiple files easily without a lot of complex programming. These techniques are becoming more useful with the switch to data arriving in electronic format rather than paper. By processing multiple files with one program or data step the chance of errors and processing time can be significantly reduced. This paper will provide examples to show different techniques for multiple file processing.

INTRODUCTION

Most of the time SAS users tend to think of data as being processed in term of records in a file. In some situations the data is not contained in just one file – it is spread across many different files. Sometimes these files are exactly the same and sometimes they are not. This paper will describe basic techniques to deal with multiple files by using one program rather than multiple programs.

Advantages of multiple file processing

The combination of built in SAS functions and the SAS macro language give SAS programmers the ability to write programs that efficiently process multiple files with one program. The ability to process multiple files with one flexible SAS program can come in handy in several common situations. Some examples are:

· Multiple work sites sending in new data files to one central location for processing and analysis

· Collection of existing data files from various sources to be processed together for analysis including meta type analyses

· Quality control to compare various versions of data files to make sure they are correct or consistent

· Testing the output of other programs to make sure the data they produce are correct or in the correct format

One issue that needs to be resolved when writing a program to process multiple files is how flexible the program should be. On the one hand writing programs that are very flexible is a

good idea because they can be used in a wide variety of cases. The downside to writing a very flexible program is that is takes longer to write and test and the added complexity may lead to hidden bugs that are not discovered during testing but show up later when the program is being used. A good approach is to design a basic multiple file handling program that can be used to process multiple files that can be plugged into other programs as needed. The basic program can also be modified at a later date to make it more specific if that is needed.

Steps to designing a program for multiple file processing

There are some basic steps that need to be followed before starting the writing of a program to process multiple files. These steps are:

· Determine how much data/files need to be processed – is the amount known ahead of time or can it vary each time the program is run?

· Determine the location of the files – are they in the same place every time the program is run or can they change to various locations?

· Determine the file formats – are they SAS files, text files, spreadsheets or some other format? Are the files all the same format?

· Is there data or information in a separate file or as part of the file name that can be used to determine how to process each file?

Once the questions above are answered then programming can start. As is the case in most programming tasks it is better to start with a very basic program to make sure the essential elements of the program work before moving on to more complex parts of the program.

Basic SAS functions for file processing

Like most programming languages, SAS has a set of functions that are designed to work with files. These functions are used to locate files, open and close files, and count the number of files in a directory. These functions are not well known by many SAS programmers but they are the key building blocks of programs that process multiple files. The most useful file handling functions are:

Dnum – this function returns the number of files in a directory

Dopen – this function opens a directory and returns a directory identifier

Dclose – this function closes a directory that was opened with dopen

Fopen – this function opens an external file and returns a file identifier

Dclose – this function closes a directory, external file, or directory member

Fread – this functions reads a record from an external file into a file data buffer

Mopen – this function opens a file by directory id and member name

Dread – this function returns the name of a directory member

%sysfunc – this macro function allows the use of a standard SAS function in a macro language programming statement.

Examples of Multiple File Processing
The following examples show various ways that SAS functions and the SAS macro language can be combined to write programs to handle multiple files in various ways:

In each example the source code is commented the first time it is used. Subsequent examples only have comments for code that was not used in previous examples.

Example 1: A basic use of multiple file handling

```
/******* **************** BASIC EXAMPLE

 This example shows the basic procedures for opening a directory and reading
the filenames in that directory and then processing those files.

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

options mprint;

%macro basic1;
%let filrf=mydir;
%let rc=%sysfunc(filename(filrf,"C:\testsas\"));  /* assign dir name */
%let did=%sysfunc(dopen(&filrf));   /* open directory */
%let lstname=;  /* clear filename macro var */
%let memcount=%sysfunc(dnum(&did));  /* get # files in directory */

%if &memcount > 0 %then  /* check for blank directory  */

  %do i=1 %to &memcount;  /* start loop for files */

   %let lstname=%sysfunc(dread(&did,&i));  /* get file name to process */

   filename dr "c:\testsas\&lstname";  /* assign file name */

   data a;     /* do whatever processing is needed with file */
    infile dr;
      input b $ 1;
    a=1;
    proc print; title "&lstname"; run;

  %end;

%let rc=%sysfunc(dclose(&did));  /* close directory */
%mend basic1;
```

Example 2: Multiple file handling with decisions on how to handle the files

```
/*  ************************ FILE TYPE EXAMPLE

   This example adds a feature that checks the file extensions as each file
is read.  The extension is then used to determine how to process the file.

  ***********************************/
 options mprint symbolgen;
 %macro filetype;
%let filrf=mydir;
%let rc=%sysfunc(filename(filrf,"C:\testsas\"));
%let did=%sysfunc(dopen(&filrf));
%let fname=;
%let memcount=%sysfunc(dnum(&did));
%if &memcount > 0 %then

  %do i=1 %to &memcount;

   %let fname=%sysfunc(dread(&did,&i));

   %let iw=%index(&fname,.);  /*  find start of extension */
   %let exts=%substr(&fname,&iw);   /* find file extension */

   filename dr "c:\testsas\&fname.";

    %if &exts = .sas %then %do; /*  check file extension */

   data a;     /* process file based on extension */
    infile dr;
      input b $ 1;
    type="SAS file";
    proc print; title "Sas file "; run;
    %end;

       %else %do;  /* the other way to process the file  */

   data a;
    infile dr;
      input b $ 1;
    type="Text file";
    proc print; title "Reg file "; run;

      %end;

    %end;

  %end;

%let rc=%sysfunc(dclose(&did));

 %mend filetype;
```

Example 3: Multiple file handling by using data about the file to determine how to process the file

```
/*  ********************* FILE LOCATION EXAMPLE

   This example uses the name of the file that is being processed to
determine where the output data should be stored.

 *******************************/
 options mprint symbolgen;
 %macro fileloc;
%let filrf=mydir;
%let rc=%sysfunc(filename(filrf,"C:\testsas\"));
%let did=%sysfunc(dopen(&filrf));
%let fname=;
%let memcount=%sysfunc(dnum(&did));
%if &memcount > 0 %then

  %do i=1 %to &memcount;

   %let fname=%sysfunc(dread(&did,&i));

   %let iw=%index(&fname,.);

  %let loc=%substr(&fname,1,%eval(&iw-1));  /* get first part of file name */

   filename dr "c:\testsas\&fname";

   filename outfile "c:\tsas\&loc\&fname";  /* use file name to determine
directory for output file */
   run;

    data a;
    infile dr;
      input b $ 1;

     file outfile;  /*  write output where it needs to go */
     put  b;

     proc print;
        title "Sas file "; run;

    %end;


%let rc=%sysfunc(dclose(&did));

 %mend fileloc;
```

Example 4: Another example using file information to determine how to process the file

```
/*  ******************************CHOOSE PR OC EXAMPLE

   This example uses the file extension to determine which proc to run on the
data.


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

  options mprint symbolgen;
 %macro chproc;
%let filrf=mydir;
%let rc=%sysfunc(filename(filrf,"C:\testsas\"));
%let did=%sysfunc(dopen(&filrf));
%let fname=;
%let memcount=%sysfunc(dnum(&did));
%if &memcount > 0 %then

  %do i=1 %to &memcount;

   %let fname=%sysfunc(dread(&did,&i));

   %let iw=%index(&fname,.);
   %let exts=%substr(&fname,&iw);

   filename dr "c:\testsas\&fname.";

    %if &exts = .txt %then %do;  /* choose proc based on file extension */

   data a;
    infile dr;
      input b $ 1;
    type="text file";
    proc freq;
      title "Sas file &fname &exts"; run;
    %end;

       %else %do;

   data a;
    infile dr;
      input b  1;
    type="Non Text file";
    proc means;
      title "Non text file &fname &exts"; run;

      %end;

    %end;
  %end;
%let rc=%sysfunc(dclose(&did));

 %mend chproc;
```

Example 5: File processing while keeping track of the number of files processed and the names of the files processed.

```
/***************************** COUNT AND STORE NAMES EXAMPLE

   This example keeps a count of each file type it process and then at the
end it prints the filenames for each type.

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

 %macro countn;

%let filrf=mydir;
%let rc=%sysfunc(filename(filrf,"C:\testsas\"));
%let did=%sysfunc(dopen(&filrf));
%let fname=;
%let memcount=%sysfunc(dnum(&did));
%if &memcount > 0 %then

 %let datcount=0;
 %let sascount=0;

  %do i=1 %to &memcount;

   %let fname=%sysfunc(dread(&did,&i));

   %let iw=%index(&fname,.);
   %let exts=%substr(&fname,&iw);

   filename dr "c:\testsas\&fname.";

    %if &exts = .dat %then %do;

   %let datcount=%eval(&datcount+1);   /* this counts files by extension */

    data tdat;   /* this stores file name in a SAS dataset */
         fname="&fname";
         output;
       run;

      proc append base=alldat data=tdat;  /* file name is appended to overall
list */

      run;

   data a;
    infile dr;
      input b $ 1;
    type="dat file";
    proc freq;
      title "Dat file &datcount"; run;
    %end;

       %else %do;
```

```
    %let sascount=%eval(&sascount+1);

   data tsas;
         fname="&fname";
         output;
       run;

      proc append base=allsas data=tsas ;

      run;

   data a;
    infile dr;
      input b $  1;
    type="Non dat file";
    proc freq;
      title "Non dat file &sascount"; run;

      %end;

    %end;

  %end;

%let rc=%sysfunc(dclose(&did));

  proc print data=alldat;  /* print list of dat files  */
   title "List of all dat files"
   run;

  proc print data=allsas;  /* print list of SAS files  */
   title "List of all SAS files"
   run;


 %mend countn;
```

CONCLUSION

More and more data is being sent to various sites in electronic format instead of paper format. This data needs to be processed and analyzed in an efficient manner with minimum errors. Using SAS code that processes multiple files at the same time allow s users to get data in and out of their systems quicker while reducing errors. The use of built in SAS functions and the SAS macro language gives programmers flexibility to write code that can handle a wide range of data in both scientific and business applications.

CONTACT INFORMATION

Kevin McGowan
Constella Group, LLC
2605 Meridian Parkway
Durham, NC 27713
(919) 313-7554
kmcgowan@constellagroup.com
http://www.constellagroup.com