Using SAS® Macros to automate external file identification and manipulation- The Power of %sysfunc.

Tim Kenney, Kenney IS Consulting, Inc. , Ojai, CA

ABSTRACT

This presentation discusses details of an approach to automate selective processing of varying data feeds into a moderately complex scheduling system. The context of the general application (developed using SAS® 6.12 on a PC platform) will be briefly described and then we will discuss in detail how features of Base SAS® and the SAS® Macro Language were utilized in building the automation.

Examples will illustrate use of the SAS® Macro function %SYSFUNC, and SAS® functions DOPEN, DCLOSE, DNUM.

INTRODUCTION AND CONTEXT

SAS® provides a variety of features useful in determining properties of various data files (including files not in SAS® format). It is possible to automate the interrogation and appropriate preprocessing of files representing a variety of sources, formats, and content using SAS® tools. Below we define the general context for which we developed these processes. Then we will give a description of some SAS® features which enabled us to implement these tools.

A business opportunity led to the technical requirement of a fully automated, data driven, scheduling system. One critical aspect of the system was the identification and subsequent processing of incoming data files from a variety of different vendors and sources.

The system was developed and implemented using SAS® 6.12 on PCs running Windows NT4.0 or Windows 95.

The design of the system was based on the finite state automaton (FSA) data model, Hopcroft (1969).

IMPLEMENTATION

Developing the SAS® code to implement the core of the scheduling system was relatively easy, basically a large select statement. This will not be discussed here, as it was treated in detail in the SUGI 99 paper "Using Base SAS® tools and the SAS® Macro Language to automate incoming data processing for a complex scheduling system." Instead, we will discuss SAS® features that we used to develop a subsystem [front-end] to efficiently interrogate incoming data for suitability/usability and promote it into the main FSA processing

Overview of processing incoming data:

To quickly implement the business solution, as well as automate the handling of multiple data sources and formats, a set of tools were developed using the SAS® Macro language. These tools rely heavily on a small core of SAS® MACRO functions and regular SAS® functions that are recent additions to the product. We describe some details of their use below.

The identification and processing of incoming data streams:

We imposed strict naming conventions for incoming data. Legal filenames are composed of 8 positionally sensitive characters with extensions of: .csv, .sd2, or .txp [our standard for naming SAS® transport files].

All input data is presented to the system in a single directory, or 'in hopper'. The application polls the input directory, determines the count, names, and types of files, and launches appropriate file specific routines to process the data and capture errors. After processing, the file is removed from the input directory and management reports issued. This cycle continues until no more files are present in the input hopper.

Much of the systems intelligence about existing files and names is realized using the SAS® Macro function %SYSFUNC. This function will populate a macro variable with the results of a SAS® function call in open MACRO code. The macro code (below) demonstrates its use in the identification and processing of files in a directory. To completely demonstrate how %SYSFUNC was used, we must look at some of the SAS® functions that we called. The most complete documentation for these and other related functions is available from the online help in the SAS® product.

Functions that we used with %SYSFUNC:

FILENAME used within a %SYSFUNC function assigns the value of 'fileref' to the macro variable specified in the fileref parameter position. The value of fileref is required by other SAS® functions that work with this directory. This value never needs to be displayed or even known, but has values like _LN00080. A return code is also generated, 0 means that no directory was found, 1 means the directory was found.

DOPEN used within a %SYSFUNC function opens a directory and returns a directory identifier value, which is required for other file access functions. It uses the value of 'fileref' returned by the *FILENAME* function to identify the directory. A return code of 0 or 1 is also available.

DCLOSE closes the directory opened by *DOPEN*.

DREAD returns the name of a member of a directory, given the directory identifier value and the member number in the directory.

DNUM counts the number of members in the open directory. The results are operating system dependent. In NT the parent directory as well as the current directory are counted as two files, and any additional files are added to that. So a value of 3 means 1 file. In OS/2 the value is strictly the count of files, so a value of 3 means 3 files are in the current directory.

PUTN formats a numeric value with a user specified format. This must be used instead of the PUT function.

PUTC formats a character value with a user specified format. This must be used instead of the PUT function.

A detailed discussion of implementation:

To start the process a program called READER is launched by the operator. This could be done using the batch mode of SAS®, or from an interactive session. The first thing READER does is verify that the input directory exists and assign a file reference to it, using %sysfunc and the *FILENAME* function. (see attached code snippet and Figure 1 for details). If the directory is not valid, a message is printed to the operator and the process is terminated.

As multiple files can be in the input hopper, the next step is to determine the number of files present. This is done with a MACRO called %cntdsn (see attachment for entire macro). The function *DNUM* called from the MACRO function %sysfunc is the heart of this capability. However, to use *DNUM*, the directory must first be opened successfully using *DOPEN*, and the operating system must be determined using the automatic MACRO variable SYSSCP. If the open is successful, then the number of files in the directory can be calculated, as well as an index for the starting file. *DCLOSE* called from %sysfunc closes the directory after we are done counting. As with all other critical operations, a return code is set to 1 if the directory cannot be opened.

Once we determine the starting file position as well as the number of files in the directory we can perform file specific operations, one at a time. As noted above, when we complete the processing for a file, we remove the file from the input directory and proceed. Control of individual file processing is accomplished with a MACRO %do %end structure. It is important to note that as files are removed from a directory, the relative index for each file is decreased by one. As a result, we found that we would always open and process the first file on every iteration, until the %do-%end structure completed looping. To prevent endless loops, we maintain a critical error flag which is examined for non zero values, and forces the MACRO to end. Building error trapping at this level was necessary. An important tool for communicating SAS® DATA step or PROC step errors back to the controlling macro is the automatic Macro Variable SYSERR, which contains the return code set by SAS® procedures. Any return code greater than 4 is a fatal error. The value of this Variable must be stored in a specific MACRO variable immediately after the critical DATA step or PROC executes, as it is reset by the next step.

The attached code snippet demonstrates a subset of all the steps required for the automation. We show how we control the loop, and how we interrogate the directory as well as individual files. As an example, we use the %getname macro (based on the SAS® function *DREAD* called from %sysfunc) to capture the file name into a macro variable, which is subsequently parsed to determine the originating vendor. Using the vendor code, a vendor specific password is determined from a user defined format and the *PUTC* function called from %sysfunc. This information is used to open input encrypted SAS® files without operator knowledge or intervention.

The entire system uses the positional fields in the file name and the extension type to selectively include the SAS® statements required for processing, but the complete system is not shown here.

EXAMPLES

```
          %*********************************************************;
          %*  CODE SNIPPET FROM READER, LOOKS AT                   *;
          %* CONTENTS OF DIR AND PROCESS ONE FILE                  *;
          %* AT A TIME                                             *;
          %*********************************************************;
%let rc=%sysfunc(filename(filrf,&indata)) ;          %*RETURN LIBRARY FILE REFERENCE ;
 %let reject=0;                                      %*DEFAULT REJECT CODE TO 0 ;
 %if &rc = 0 %then %do ;                             %*IF THE DIRECTORY EXISTS ;
  %cntdsn(pathin=&indata);                           %*CALL MACRO TO COUNT MEMBERS ;
  %let in_cnt=&memcnt;                               %*GET NUMBER OF FILES IN DIR ;
  %let filenum=&start;                               %*  POINTER TO 1ST FILE IN DIR;
  %let in_id=%sysfunc(dopen(&filrf));                %*GET ID FOR DIRECTORY;
  %if &doscnt > 0 %then %do i=&start %to &in_cnt;    %* PEEL OFF ONE AT A TIME;
   %let filecode=0;                                  %* DEFAULT IS GOOD FILE ;
   %getname(gdid=&in_id,mnum=&filenum);              %* GET ITS NAME, ALWAYS FIRST ;
   %let name1=%scan(&dname,1);                       %* PARSE 8.3 INTO FILE NAME AND ;
   %let name2=%scan(&dname,2);                       %* EXTENSION ;
   %if %length(&name1) = 8 %then %do ;               %* GET SPECIFIC INFO ;
    %let vndr=%substr(&dname,4,1) ;                  %* SOURCE - VENDOR ;
    %let pw=%sysfunc(putc(&vndr,$pwfmt.));           %*GET PASSWORD FOR FILE ACCESS;
   %end ;
   %else %do ;                                       %* FILE IS BAD, NOT 8 BYTES;
    %let filecode=1;                                 %* BAD FILE NAME ;
    data _null_ ;                                    %* NOTE TO OPERATOR ;
     file print ;
     put "BAD file report, &indata.\&dname violates naming convention" ;
    run ;
   %end ;
   %*THE FILE EXTENSION IS USED TO DETERMINE THE FORMAT, ;
   %*AND ROUTINES ARE LAUNCHED TO OPEN THAT PARTICULAR ;
   %*FORMAT AND COME BACK WITH A RETURN CODE AFTER PROCESSING ;
   %*IS COMPLETE... NOT SHOWN HERE ;
  %end ;
```

The preceding code snippet calls two very important SAS® MACROS; %cntdsn and %getname. They also use the %sysfunc function and appear below.

```
 ****************************************************************
 *            MACRO CNTDSN                                      *
 *                                                              *
 *  pathin= input data path                                     *
 *                                                              *
 *  returns  did:              the directory id                 *
 *           start:    an operating specific value first file  (1 or 3)  *
 *           doscnt:   Number of files in directory            *
 *           cnt_rc:   Return code for error trapping (0 or 1) *
 *           memcnt: 'Native' value of member count            *
 ****************************************************************;

%macro cntdsn(pathin=&cd.\data\datain);
 %global memcnt did cnt_rc start doscnt ;
 %let rc=%sysfunc(filename(filrf,&pathin)) ;          %*CHECK DIRECTORY, ASSIGN FILE REFERENCE ;
 %if &rc = 0 %then %do ;
  %let did=%sysfunc(dopen(&filrf)) ;                  %*OPEN DIRECTORY,ASSIGN DIRECTORY ID ;
  %let memcnt=%sysfunc(dnum(&did)) ;                  %* ASSIGN MEMBER COUNT TO &MEMCNT ;
  %if &sysscp=OS2 %then %do ;                         %*&MEMCNT IS FILE COUNT IN OS/2 ;
   %let doscnt=&memcnt;
   %let start=1;                                      %*FIRST FILE IN OS/2 IS INDEXED BY 1 ;
  %end ;
  %else %if &sysscp=WIN %then %do ;
   %let doscnt=%eval(%sysfunc(dnum(&did))-2) ;        %*&MEMCNT IS FILE COUNT PLUS .. AND . ;
   %let start=3;                                      %*IN NT, FIRST FILE IS ALWAYS 3 ;
  %end ;
  %let rc=%sysfunc(dclose(&did));                     %*CLOSE DIRECTORY ;
  %let cnt_rc=0;
```

```
 %end ;
 %else %do ;                                    %*CAPTURE ERROR ;
  %put Bad directory &pathin;
  %let cnt_rc=1;                                %*SET RETURN CODE ;
 %end ;
%mend cntdsn;


 ***************************************************************
 *            MACRO GETNAME                                    *
 *                                                             *
 * Take data set id and member number                          *
 * and return the member name as &dname                        *
 *  Directory must be open for this to work .                  *
 * gdid= Directory id number returned from dopen               *
 * mnum= members number in the directory                       *
 ***************************************************************;
%macro getname(gdid=,
        mnum=);
 %global dname;
 %let tdname=%sysfunc(dread(&gdid,&mnum));      %*USE DIRECTORY ID AND FILE INDEX ;
 %let dname=%upcase(&tdname);                   %*TO CAPTURE DATASET NAME, UPPER ;
%mend getname;
```

REFERENCES

Hopcroft, J.E. and Ullman, J.D. (1969), *Formal Languages and Their Relation to Automata*, Addison-Wesley. Reading, MA.

Kenney, T.A. and Schroeder, D. A. (1999), *Using Base SAS® Macro Language to automate incoming data processing for a complex scheduling system*, Proceedings of the Twenty-Fourth Annual SAS® Users Group International Conference.

SAS Institute Inc. (1997), *SAS® Macro Language: Reference, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS® Language Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Tim Kenney, M.A.
Kenney IS Consulting, Inc.
109 D N. Montgomery St.
Ojai, CA 93023
(805) 646 9982
(805) 646 5111 (fax)
tkenney@west.net