PharmaSUG 2019 - Paper AP-001

Get Smart! Eliminate Kaos and Stay in Control – Creating a Complex Directory Structure with the DLCREATEDIR Statement, SAS® Macro Language, and Control Tables

Louise S. Hadden, Abt Associates Inc.

ABSTRACT

An organized directory structure is an essential cornerstone of data analytic development. Those programmers who are involved in repetitive processing of any sort control their software and data quality with directory structures that can be easily replicated for different time periods, different drug trials, etc. Practitioners (including the author) often use folder and subfolder templates or shells to create identical complex folder structures for new data spans of data or projects, or use manual processing or external code submitted from within a SAS® process to run a series of MKDIR and CHDIR commands from a command prompt to create logical folders. Desired changes have to be made manually, offering opportunities for human error. Since the advent of the DLCREATEDIR system option in SAS version 9.3, practitioners can create single folders if they do not exist from within a SAS process. Troy Hughes describes a process using SAS macro language, the DLCREATEDIR option, and control tables to facilitate and document the logical folder creation process. This paper describes a technique wrapping another layer of macro processing which isolates and expands the recursive logical folder assignment process to create a complex, hierarchical folder structure used by the author for a project requiring monthly data intake, processing, quality control and delivery of thousands of files. Analysis of the prior month’s folder structure to inform development of control tables and build executable code is discussed.

INTRODUCTION

Since 1998, the U.S. Centers for Medicare and Medicaid Services (CMS) has maintained a website, Nursing Home Compare, which provides detailed quality information about every certified nursing home in the country. In December 2008, CMS greatly enhanced the usability of the website by adding an easy-to-understand 5-star rating. Each nursing home receives one to five stars based on performance in each of three key quality domains (health inspections, reported staffing levels, and quality measures derived from mandated assessments of resident health and well-being) plus an overall quality rating. Calculation of ratings requires integration of information from both facility and resident-level data sources. SAS® was used extensively in analysis to support the development of the rating system, and it is currently used to process data to refresh the ratings each month, based on newly collected data in each domain.

Data from two organizations is received monthly, data from two additional organizations is received quarterly, and data from an additional organization is received biannually. Processing is classified into four separate categories: reading in original data, calculating ratings, “other” processing (reading in data not used in ratings, redacting text fields, preparing for various types of output), and production of various types of output. A standard set of programs run in a standard, complex folder structure each month, with additional programs for quarterly and biannual processing.

This project is representative of the adage attributed to Heraclitus “The only constant is change”. The incoming data changes every month. Sometimes the underlying structure of the data changes, and there needs to be flexibility in our system. Accordingly, we employ control tables in a number of instances, including date macro include files, hard code include files, self-writing batch files to run sequential programs, etc. This has worked to an extent - our processes have worked well for a long time, over 9 years, perhaps with and/or because of extraordinary effort by the programmers and technicians. As we look at our system with an eye toward software quality, though, it is clear we have a long way to go.

One time-consuming aspect of this project is recreating the complex directory structure every month for the next month’s processing. We had been using an empty “skeleton” file structure, copying it over each month, then copying the most recent programs used in the prior month’s processing by hand. The files in the “skeleton” and even the structure itself had a habit of changing on a regular basis, and we started to have a clutter of file skeletons, something inevitably would not get copied over, and chaos reigned. The DLCREATEDIR option as described by Troy Hughes offered an alternative to this clumsy and error prone manual process.
COLLECTING INTELLIGENCE

In order to construct the desired directory structure for each prospective month, it is necessary to inventory the previous month’s directory structure, and the contents thereof, as it is possible that the directory structure has changed in the prior month. There are a number of ways to accomplish this, and several excellent papers describing those ways, some referenced in this paper. I chose to repurpose a data management tool that I originally designed for data storage management. It is also advisable to fully familiarize oneself with the task at hand – in my case, monthly processing, in terms of knowing the data being processed, the environment in which your data is housed and processed, and the software used to process it. SAS provides a number of tools to assist in this process including many ways to utilize SAS metadata and a wide variety of data step functions and macro functions.

As noted above, our monthly processing routine utilizes control tables containing date macros, including macros describing dates for the previous month’s processing. Utilizing these control tables, we run a program “readdirlist.sas” on the previous month’s processing folder. This program redirects piped input into a temporary SAS data set, which is massaged to parse out file type variables, etc., including directories and subdirectories. Note that running the program “in situ” allows us to define file paths as they exist in the current system, which increases portability of the techniques shown in this paper. The program outputs an MS Excel® workbook and a SAS data set named with a date stamp. We use the data set, filtered for specific purposes such as creating directories and copying files, as input.

Note: date macros are part of a separate include file. You can also use %WINDOW and %DISPLAY to collect this information. This technique works either interactively or in batch (in either case you need to fill in the information in order for the program to run.)

/* set the dates for the current round of processing */
/* prompts the user for the date and year each time batch mode is ran */

/** %WINDOW defines the prompt **/
%window info
  #4 @5 'Please enter the year files are received(YYYY):'
  #4 @53 fileyear 4 attr=underline display=yes auto=no color = blue

  #8 @5 'Please enter the date files are received(MMDD):'
  #8 @53 filedate 4 attr=underline display=yes auto=no color = red;

/** %DISPLAY invokes the prompt **/
%display info;

OR

/* Either in a control file referenced by the program or located in the program as shown below */

%let prevyear=2017;
%let prevdate=0701;

%let archout=mp&prevyear.&prevdate.;

libname dd '.';
filename yy '.\reviseddirlist.txt';
filename dirlist pipe "dir ""S:\Projects\NH-Compare\Data_from_CMS\Monthly Processing\Files&prevyear.&prevdate."" /s'';
run;

data temp1000 ;
  infile dirlist lrecl=1000 missover pad;
  input foo1 $char1000.;
if foo1='' then delete;
foo1=left(foo1);
if index(foo1,'bytes')>0 then delete;
seqnum=put(_n_,z8.);
retain sumflag 0;
dirflag=(substr(foo1,1,9)='Directory');
if substr(foo1,25,5)='<DIR>' then recatype='sdirname';
if substr(foo1,40,1)='.' then delete;
if substr(foo1,38,1) ne ' ' then recatype='filename';
if substr(foo1,1,9)='Directory' and substr(foo1,25,5) ne '<DIR>' then
   recatype='mdirname';
sumflag=sumflag+dirflag;
run;
...
proc export data=temp5 dbms = excel
   outfile = "\&archout._listing.xlsx" replace;
sheet="&archout.";
run;
data dd.mp&prevyear.&prevdate.;
set temp5;
run;

Bonus tip: Use %WINDOW to create a prompt to collect information to be used as a macro variable.

Careful review of the Excel workbook output is absolutely essential. Automation of any sort inherently requires constant vigilance in order to achieve quality outcomes – and this time-saving technique is no exception. “Ad Hoc” changes may have occurred in a prior month’s processing that were not intended to go into production. There’s no such thing as being too careful! Luckily SAS provides many opportunities to collect metrics in this effort. Comparison of READDIRLIST output from one month to another allows review of changes to the directory structure, programs, etc. and can be done programmatically using PROC COMPARE.
**Bonus tip:** Use the SAS supplied SIZEKMG format to convert a numeric size variable into kilobytes, megabytes, and gigabytes. For example

```sas
sizefmtd=put(size,sizekmg.);
```

results in a character string such as 118KB.

**Taking Control**

Creating a directory with the DLCREATEDIR option is as easy as entering an OPTIONS DLCREATEDIR statement at the top of your program, and then specifying the desired library name in a libname statement. However, our directory structure is both complex and voluminous, and since the DLCREATEDIR system option only creates one subdirectory at a time, requiring changes in the “base” directory, the program would have been inordinately long, and would have required a lot of upkeep. In our programs the date macros are contained in an external control file, shown here for explanatory purposes. Note that to avoid any potential catastrophic overwriting of a production library the new “base” folder has a suffix of “T”.

```sas
OPTIONS dlcreatedir;

%let fileyear=2017;
%let filedate=0801;
%let prevyear=2017;
%let prevdate=0701;

%let basedir=S:\Projects\NH-COMPARE\Data_From_CMS\Monthly Processing\;

/* create a top level libname */
libname monthly "&basedir.Files&fileyear.&filedate.T";

/* now create the new month's directory */
libname base "&basedir.Files&fileyear.&filedate.T";
```

We modified two macros, pioneered by Troy Hughes in “SAS® Spontaneous Combustion: Securing Software Portability through Self – Extracting Code”. One of these macros “crawls” through a list of folders (seen below in control table dirconfig.txt), and the primary macro which reads in the control table, and calls the crawler macro. The primary macro uses two parameters specifying the directory name and the level (seen at the far left below). Data from the directory listing above is used to self-generate the macro call.
As you can see from the control table above, a level may have more than 1 subfolder, but the DLCREATEDIR system option requires that directories be built one at a time. The crawler macro allows the creation of multiple subdirectories with a single macro invocation. The control table is also easily editable if the directory structure changes (which it has and does.)
%let modir=&basedir.&modirsuff.;

data temp&level;
   length category $ 8;
   set control;

     if upcase(tab)="<LIBRARY>" then category="lib";
   else do;
   if category="lib" then do;
   call execute('%crawler(dir='||strip(lib)||',loc='||strip(loc)||');');
   end;
   end;
   retain category;

run;

This portion of the combined directory creation and file copying routine runs quickly and efficiently, yielding a perfectly formed, empty directory structure for succeeding months.

ELIMINATING KAOS

Along with a complex directory structure for our project, there are a multitude of other files required for processing the data: programs, include files, control files, metadata files, XML maps, format and macro catalogs, etc. As with the directory structure itself, other files change over time, and the same information we collected about the previous month’s directory can help us populate our new monthly folder. Copying over all of the processing from a prior month would be counter-productive and confusing. We can use our metadata from the prior month (and some substantive knowledge about the processing) to choose the files we need, by subdirectory.

As with creating directories and subdirectories, there are many ways to copy files within SAS. An in-depth discussion is not within the scope of this paper. Suffice it to say that the choice of which method to use will depend on a number of factors, including your operating system, environment, security concerns, etc.
We wanted to be able to use macro processing, and control file types that were copied by subdirectory. We created several specific macros for different purposes – one for copying just SAS programs, one for copying SAS programs and files with .txt extensions (include and control files), one for copying SAS data sets and MS Excel workbooks, and so on.

Each macro performs 3 primary tasks: identifies matching directories for the prior and upcoming month, creates a list of curated files to be copied, and executes copying the files. Two macros, %gocopy and (variants of) %copyfiles, drive the processes. The macro used for copying files in the top level of the monthly processing folder serves as a template for the other more specific file copying macros.

/* copies previous mo files to current mo directories based on order number */

%macro copyfiles(num);
proc sql noprint;
select filename into: name
from filenames
where ord=&num;
quit;

options noxwait;
%local rc fileref ;
%let rc = %sysfunc(filename(fileref,%qcmpres(&currpath))) ;

%if %sysfunc(fexist(&fileref)) %then %do;
%sysexec copy "%qcmpres(&prevpath\&name)"
"%qcmpres(%qcmpres(&currpath)\&name)" ;
%end;
%let rc=%sysfunc(filename(fileref)) ;
%mend;

%macro gocopy;
%do i=1 %to &maxord;
%copyfiles(&i);
%end;
%mend;

/* macro to copy top level files */

%macro copytop;
%let prevpath=&basedir.Files&prevyear.&prevdate./;
%let currpath=&basedir.Files&fileyear.&filedate.T/;

/* make a list of everything in a directory */
%let indir=&prevpath;
%let nfiles=0;
filename indir "&indir";
data _filenames1(where=(
index(upcase(filename),'.' gt 0 )));
format filename $45. ;
did=dopen('indir');
do i=1 to dnum(did);
  filename=dread(did,i);
  output;
end;
rc=dclose(did);
keep filename;
run;

data filenames;
set _filenames1;
ord=_n_;        
run;

proc sql noprint;
select max(ord) into: maxord
from filenames;
quit;
%gocopy;

%mend copytop;

%copytop;

Selective copies are done on files in subdirectories, using a macro parameter to run through
subdirectories, and using the index function to select certain file types by extension. The macro below
selects SAS PROGRAMS but excludes SAS data sets, and selects files with the .txt extension. Other
macros select only SAS programs and files, and all files. As always, you should consult your metadata
(directory listing) to assess whether or not all needed files are being copied, and that ONLY needed files
are being copied. In our case, processing yields some massive, and numerous, files that should not be
copied. Specific macros are assigned to subdirectories based on a careful review and knowledge of our
project metadata.

%macro copysasp(subdir=1OriginalData/Processing);
...
data _filenames1(where=(
  (index(upcase(filename),'.SAS') gt 0
  and index(upcase(filename),'BDAT') eq 0) OR
  index(upcase(filename),'TXT') gt 0
)); ...
WOULD YOU BELIEVE? AND NEXT STEPS

The DLCREATEDIR macro process and copy files suite of macros are contained in the same program, and take just seconds to execute.

NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:

- real time 29.23 seconds
- cpu time 8.06 seconds

The concept of control tables and metadata as program drivers, in conjunction with SAS macros, makes a difficult and onerous monthly chore a pleasure.

In order to more fully automate the process, we could use the metadata worksheet created above, and filter a list of the base directory and subdirectories from the prior month into specialized control tables the series of macros can read. However, it is difficult to ascertain what folders should be copied from which directories mechanically. As long as change is a constant, control tables will need to be rigorously created and reviewed.

REFERENCES


Usage Note 56710: The DLCREATEDIR system option creates a single directory

ACKNOWLEDGMENTS

The author gratefully acknowledges the helpful work of Troy Martin Hughes, Justin Jia, Amanda Lin, Laura Lawrence, Art Carpenter, Mary Rosenbloom, Richann Watson and Kirk Paul Lafler, among others.

RECOMMENDED READING

• Base SAS® Procedures Guide

CONTACT INFORMATION

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

Louise Hadden: Louise_Hadden@abtassoc.com

Scan me for sample code and additional papers!

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 trademarks of their respective companies.