PharmSUG2011 - Paper AD09

The Path, The Whole Path, And Nothing But the Path, So Help Me Windows

Arthur L. Carpenter, California Occidental Consultants, Anchorage, AK

ABSTRACT

Have you ever needed to determine a directory path dynamically? Have you needed to know the name of the currently executing program and where it resides? There are a number of tools available to the SAS programmer that can be used to determine path and location information. These tools range from common to obscure and from simple to complex and they utilize SAS Language functions, System Options, SASHELP views, DICTIONARY tables, and OS Environmental Variables.

This presentation will take a survey of a number of these techniques for finding and determining path and location information. This information can be important for all levels of programmers, and you should be aware of these techniques even if you think that you will never need them.

KEY WORDS

Path, FILENAME, DDL, FMTSEARCH, CALL METHOD, SASHELP, PATHNAME, UNC Path

INTRODUCTION

It is not all that unusual to have a need to determine a physical path. In a dynamic application or even a macro program the programmer cannot always know what a path is going to be when the macro is executed. This means that the macro itself will have to gather this information at the time of execution.

Fortunately there are several ways to gather this path information depending on the situation and what needs to be accomplished.

USING THE PATHNAME FUNCTION

DATA step functions provide power and utility that transcends the DATA step. Very often these functions will be utilized through the macro language.

The PATHNAME function returns the physical path for a given *fileref* or *libref*.

```
filename saspgm "F:\Primary\TheWholePath\abc.sas";

%let pgmpath = %sysfunc(pathname(saspgm));
%put &pgmpath;
```

The LOG shows:

```
F:\Primary\TheWholePath\abc.sas
```

Using PATHNAME on a *libref* returns the path to that directory.

```
libname whole "F:\Primary\TheWholePath";

%let path2whole =  %sysfunc(pathname(whole));
%put &path2whole;
```

The LOG shows:

```
F:\Primary\TheWholePath
```

You can even use it on concatenated *fileref*s such as the autocall library. To gather the current location of all of the locations in the SASAUTOS *fileref* you could specify:

```
%sysfunc(pathname(sasautos))
```

SASHELP VIEWS and DICTIONARY Tables

The path information for existing *libref*s and *fileref*s can be gathered by examining the SASHELP views and a SQL DICTIONARY tables. Here you can find not only the path that would be returned by the PATHNAME function, but other things such as ENGINE as well.

SASHELP.VLIBNAM
DICTIONARY.LIBNAMES

Each row in the view SASHELP.VLIBNAM (note the spelling) contains the *libref* and path information for each *libref* (more than one row for concatenated *libref*s). A portion of a listing of this view shows the primary variables.

```
                         The View SASHELP.VLIBNAM

Obs   libname   engine   path

  1   WHOLE       V9     F:\Primary\TheWholePath
  2   SASHELP     V9     C:\Program Files\SAS\SAS 9.1\nls\en\SASCFG
  3   SASHELP     V9     C:\Program Files\SAS\SAS 9.1\core\sashelp
  4   SASHELP     V9     C:\Program Files\SAS\SAS 9.1\reporter\sashelp
```

The following DATA step extracts the appropriate row and saves the path in a macro variable.

```
data _null_;
   set sashelp.vlibnam(keep=libname path
                       where=(libname='WHOLE'));
   call symputx('wholepath',path,'l');
   run;
```

The macro variable &WHOLEPATH now contains the path associated with the *libref* WHOLE.

Similar information can be extracted from the LIBNAMES DICTIONARY table through an SQL step.

```
proc sql noprint;
   describe table dictionary.libnames;
   select path into :sqllibpath
      from dictionary.libnames
         where libname='WHOLE';
   quit;
%put &sqllibpath;
```

In this SQL step the path associated with the *libref* WHOLE has been placed into the macro variable &SQLLIBPATH. The DESCRIBE statement used here is not really needed. It only writes the names of the columns of the table to the LOG.

SASHELP.VEXTFL and DICTIONARY.EXTFILES

Path information for external files (*fileref*s) can be determined using either the SASHELP.VEXTFL view or through the use of the SQL table DICTIONARY.EXTFILES.

Except for different file and variable names, the DATA step is almost identical to the one used to retrieve the path for the *libref*.

```
data _null_;
   set sashelp.vextfl(keep=fileref xpath
                      where=(fileref='SASPGM'));
   call symputx('pgmpath',xpath,'l');
   run;
%put &abcpath;
```

The SASHELP.VEXTFL view could also be used in an SQL step, however, when SQL is used and a DICTIONARY table is available, the DICTIONARY table will generally be your better alternative. The SQL step could be something like:

```
proc sql noprint;
select xpath into :pgmsqlpath
   from dictionary.extfiles
      where fileref='SASPGM';
quit;
%put &pgmsqlpath;
```

FINDING FORMATS

Much like finding the location of a data table, we may also be faced with locating format catalogs. Fortunately catalogs are stored in libraries and we have already determined how to find the path associated with a library. However since format catalogs can be concatenated across multiple libraries we must first determine the libraries for which we need to determine the paths.

As is so often the case there are multiple techniques for determining the *libref* that contains the catalog that contains the format of interest.

The FMTSEARCH system option is helpful because it lists the library (or libraries) that SAS will search when looking for a format. We can access this option by using the GETOPTION function.

```
%macro fmtlibs;
   %sysfunc(getoption(fmtsearch))
%mend fmtlibs;
%put %fmtlibs;
```

This first version of %FMTLIBS returns the list of search locations enclosed in parentheses.

```
(WORK LIBRARY)
```

Since the parentheses can cause parsing problems a slightly more sophisticated version of %FMTLIBS removes them using the TRANSLATE function.

```
%macro fmtlibs;
   %sysfunc(translate(%sysfunc(getoption(fmtsearch)),%str(   ),%str(%( %))))
%mend fmtlibs;
```

The resulting list of *libref*s `WORK LIBRARY` can now be passed to the PATHNAME function to return the location.

Now all we have to do is parse the list and write the path. The macro %PATHINFO returns the paths associated with a list of one or more *libref*s.

```
%macro pathinfo(liblist);
%local lnum libref path;
%put &liblist;
%let lnum = 0;
%do %while(%qscan(&liblist,&lnum+1,%str(%( %))) ne );
   %let lnum = %eval(&lnum+1);
   %put &lnum;
   %let libref = %qscan(%bquote(&liblist),&lnum,%str(%( %)));
   %put &libref;
   %let path = %sysfunc(pathname(&libref));
   %put &libref &path;
%end;
%mend pathinfo;
%pathinfo(%fmtlibs)
```

You can also gather path information for formats and format catalogs through the use of the view SASHELP.VFORMAT. This view contains one row per format (SAS supplied and user defined), and for user defined formats it also contains the *libref* and catalog name. Similar information can be determined using the SQL table DICTIONARY.FORMATS.

DETERMINING THE EXECUTING PROGRAM NAME AND PATH

Sometimes we need to be able to automatically detect the name or location of an executing program. This can be especially helpful when we write applications that need to self document, perhaps by placing the name and location of the executing in a footnote of the generated table.

This is fairly straight forward when the executing program is running in batch mode. In batch mode the name of the executing program is stored in the system option SYSIN, and the value of system options can be retrieved using the GETOPTION function

Under the windows operating system, the name of the executing program and its path is stored in the environmental variables SAS_EXECFILENAME and SAS_EXECFILEPATH. Environmental variables are maintained by the OS, however SAS can both populate and access their values. Whenever a SAS program is executed, this includes when it is executed through the Display Manager from the Enhanced Editor, these environmental variables are updated.

The values of environmental variables are accessed through the use of the %SYSGET macro function. The following function call retrieves the name of the executing program:

```
%sysget(SAS_EXECFILENAME)
```

The %QSUBSTR function can be used to remove the .SAS extension from the filename.

```
%qsubstr(%sysget(SAS_EXECFILENAME),1,%length(%sysget(SAS_EXECFILENAME))-4)
```

When we need to know the location of the SAS program (when executing from the Enhanced Editor this is the location from which the executing program was retrieved), we can use the SAS_EXECFILEPATH environmental variable.

```
%macro grabpathname;
   %sysget(SAS_EXECFILEPATH)
%mend grabpathname;

%put %grabpathname;
```

The %GRABPATHNAME macro returns both the path and the name of the executing program.

```
F:\Primary\TheWholePath\grabpathname.sas
```

We can eliminate the program name from the path by using a combination of these two environmental variables. In the %GRABPATH macro the %QSUBSTR function is used to remove the name of the executing program from its path.

```
%macro grabpath;
   %qsubstr(%sysget(SAS_EXECFILEPATH),
      1,
      %length(%sysget(SAS_EXECFILEPATH))-%length(%sysget(SAS_EXECFILEname))
      )
%mend grabpath;
```

The macro %GRABPATH resolves to:

```
F:\Primary\TheWholePath\
```

RETRIEVING THE UNC PATH

When a program resides on a network server, the server name is generally mapped to a drive letter. Since this drive letter can be user specific, knowing that a program resides on the F:\ drive for one user is not necessarily helpful to someone else. As was shown using all of the previous methods, it is always the mapped drive that is returned, therefore a different approach is needed to retrieve the actual or UNC path. Although the UNC path information is not stored in a location that is directly available to SAS through the use of a function, it is still possible to get his information - the process is just a bit more challenging.

Certainly we know that the OS has to know the relationship between the mapped drive letter and the actual UNC location. Under Windows this information is stored in a Dynamic Link Library, DLL. Windows has internal tools for accessing the information contained in a DLL and these tools can be accessed from within SAS using the CALL MODULE routine. To make the tools available to the CALL MODULE routine we must first create a CATALOG SOURCE entry for it to operate against. This entry contains the arguments that are passed to and from the Windows routine WNETGETCONNECTIONA.

```
FILENAME SASCBTBL CATALOG "work.temp.attrfile.source";

DATA _NULL_;
    FILE SASCBTBL;

    PUT "ROUTINE WNetGetConnectionA MODULE=MPR MINARG=3 MAXARG=3 STACKPOP=CALLED
RETURNS=LONG;";
    PUT "  ARG 1 CHAR INPUT BYADDR FORMAT=$CSTR200.;";
    PUT "  ARG 2 CHAR UPDATE BYADDR FORMAT=$CSTR200.;";
    PUT "  ARG 3 NUM UPDATE BYADDR FORMAT=PIB4.;";
RUN;
```

The arguments themselves are specific to each routine. The WNetGetConnectionA routine expects three arguments. Here the SOURCE entry has been written to a catalog in the WORK directory, however you will generally make this permanent so that you only have to run this step once.

The CALL MODULE routine can then be used to access the WNetGetConnectionA routine and to retrieve the location.

```
%MACRO getUNC;
   %* Determine the UNC path for the SAS program being executed.;
   DATA _NULL_;
      length input_dir $200 output_dir $200;

      * The input directory can only be a drive letter + colon ONLY e.g. j: ;
      input_dir = "%qsubstr(%grabpath,1,2)"; ❶
      output_dir = ' ';
      output_len = 200;
      call module("WNetGetConnectionA", input_dir, output_dir, output_len); ❷

      call symputx('dir',input_dir,'l'); ❸
      call symputx('path',output_dir,'l'); ❹
      RUN;

   %* Get the name for the program of execution.;
   %let name = %grabname;
   %put drive letter is &dir; ❺
   %put path is &path; ❻
   %put name is &name; ❼
   &path ❽
%MEND getunc;
```

❶ Using the %GRABPATH macro the path which contains the mapped drive letter is obtained. The %QSUBSTR function is then used to strip off everything except the drive letter e.g. F:.

❷ Call the MODULE routine passing it the mapped drive letter contained in the variable INPUT_DIR.

❸ For fun save the mapped drive letter to the macro variable &DIR.

❹ Save the UNC path to the macro variable &PATH.

❺ For fun write the mapped drive letter to the LOG.

❻ For fun write the UNC path to the LOG.

❼ For fun write the program name to the LOG.

❸ The UNC path is returned.

The LOG, which includes the fun output (to help us see what is going on) includes:

```
drive letter is F:
path is \\CALOXYDELL\PrimaryDell
name is Use_SASCBTbl
```

For this example the program USE_SCBTBL.SAS resides on the server which has been mapped to the F: drive. The UNC path shows that the F: drive letter has been mapped to the \primarydell directory on the \\caloxydell server.

Programs that are running on the C: drive are local and will not return a UNC path.

SUMMARY

For dynamic applications and for programs that can reside on servers it may sometimes be necessary to have the program itself determine path information. Fortunately this information is available through a variety of sources, and most of these sources are fairly straightforward in their use.

Although some of the methods discussed here tend to be a bit complex and will most likely only rarely be needed, it remains the programmer's responsibility to know, at the very least, what is possible.

ABOUT THE AUTHOR

Art Carpenter's publications list includes four books, and numerous papers and posters presented at SUGI, SAS Global Forum, and other user group conferences. Art has been using SAS® since 1977 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Advanced Professional™ programmer, and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

REFERENCES

Carpenter, Arthur L., 2004, *Carpenter's Complete Guide to the SAS® Macro Language, 2nd Edition*, SAS Institute Inc., Cary NC.

A search of support.sas.com using "SAS_EXECFILENAME" surfaces a number of references to this general topic.

TRADEMARK INFORMATION