MWSUG 2024 - Paper AD66

Using SAS I/O Functions to Retrieve External Metadata

Jake Reeser, NORC at the University of Chicago, Chicago, IL

ABSTRACT

This paper discusses external file and I/O functions. I cover how to use these functions to retrieve SAS metadata efficiently, external files, as well as what metadata is available to the user. I will show how to access the metadata of all files in an external file directory. I will lastly show how to use this file metadata to allow for easier data processing.

INTRODUCTION

SAS users will at times need to access external files and directories. Through I/O functions, we can access external files directly, whether that be SAS datasets or other file types. I/O functions allow us to do a variety of operations, including reading/writing from/to external files, as well as retrieving their metadata. By using I/O functions with SAS macros, we can access our datasets/files without having to utilize SAS procedures or the data step. This allows us to write more efficient SAS programs.

This paper would be for those of an intermediate skill level. Knowledge of the SAS macro language is required

USING I/O FUNCTIONS

The first step in accessing the metadata of a SAS dataset is to open it through the OPEN function. This function returns a unique numeric dataset identifier, which is then used as the input in other dataset access functions. If the dataset can't be opened, the function returns 0. The OPEN function can be called in the data step:

```
Dsid=OPEN("sashelp.baseball");
```

As well as through macros:

```
%let dsid=% SYSFUNC(OPEN(sashelp.baseball));
```

NOTE: When using the OPEN function in the data step, you need quotes around the data set name. Quotes are not needed when using the OPEN function in a macro.

After finishing with the dataset, the CLOSE function must be used. This is considered best practice, as you don't want dataset unused identifiers floating. The CLOSE function returns 0 on success and a non-zero integer on failure. The CLOSE function can be called in the data step:

```
Dsid=CLOSE(dsid);
```

As well as through macros:

```
%let dsid=% SYSFUNC(CLOSE(&dsid.));
```

Retrieving Metadata

Once the SAS dataset has been opened, we can retrieve the metadata. To retrieve numeric metadata, the ATTRN function is used. This function takes in the dataset ID we created with the OPEN function, as well as the name of the attribute we want to retrieve. For example, we could retrieve the number of variables like so:

```
%let dsid=% SYSFUNC(OPEN(sashelp.baseball));
```

```
%let num_var=% SYSFUNC(ATTRN(&dsid.,nvar));
%put &=num_var.;
%let dsid=% SYSFUNC(CLOSE(&dsid.));
```

The resulting SAS log:

```
NUM_VAR =24
```

All the different attribute types that you can access with the ATTRN function are listed below:

Attribute	Description
ALTERPW	Specifies whether a password is required to alter the data set.
ANOBS	Specifies whether the engine knows the number of observations.
ANY	Specifies whether the data set has observations or variables.
ARAND	Specifies whether the engine supports random access.
ARWU	Specifies whether the engine can manipulate files.
AUDIT	Specifies whether logging to an audit file is enabled.
AUDIT_DATA	Specifies whether after-update record images are stored.
AUDIT_BEFORE	Specifies whether before-update record images are stored.
AUDIT_ERROR	Specifies whether unsuccessful after-update record images are stored.
CRDTE	Specifies the date on which the data set was created. The value that is returned is the internal SAS datetime value for the creation date. TIP Use the DATETIME. format to display this value.
ICONST	Returns information about the existence of integrity constraints for a SAS data set.
INDEX	Specifies whether the data set supports indexing.
ISINDEX	Specifies whether the data set is indexed.
ISSUBSET	Specifies whether the data set is a subset.
LRECL	Specifies the logical record length.
LRID	Specifies the length of the record ID.
MAXGEN	Specifies the maximum number of generations.
MAXRC	Specifies whether an application checks return codes.
MODTE	Specifies the last date and time that the data set was modified. The value returned is the internal SAS datetime value.
NDEL	Specifies the number of observations in the data set that are marked for deletion.

NEXTGEN	Specifies the next generation number to generate.
NLOBS	Specifies the number of logical observations (the observations that are not marked for deletion). An active WHERE clause does not affect this number.
NLOBSF	Specifies the number of logical observations (the observations that are not marked for deletion) by forcing each observation to be read and by taking the FIRSTOBS system option, the OBS system option, and the WHERE clauses into account.
NOBS	Specifies the number of physical observations (including the observations that are marked for deletion). An active WHERE clause does not affect this number.
NVARS	Specifies the number of variables in the data set.
PW	Specifies whether a password is required to access the data set.

Table 1: Attributes for ATTRN function

To retrieve character metadata, the ATTRC function is used. This function also takes in the dataset ID we previously created, as well as the name of the attribute we want to retrieve. For example, we could retrieve information on if a dataset is sorted, and if so, by which variables:

```
PROC SORT data=sashelp.baseball out=work.baseball;

by team name;

run;

%let dsid=%sysfunc(OPEN(sashelp.baseball));

%let ds_sort =% SYSFUNC(ATTRC(&dsid.,sortedby));
%put &= ds_sort.;
%let dsid=% SYSFUNC(CLOSE(&dsid.));
```

The resulting SAS LOG:

```
DS_SORT=Team  Name
```

All the different attribute types that you can access with the ATTRC function are listed below:

Attribute	**Description**
CHARSET	Returns a value for the character set of the computer that created the data set.
COMPRESS	Returns a value that specifies how a data set is compressed.
DATAREP	Returns a value that indicates whether the data set is in a native format.
ENCODING	Specifies the encoding of the data set.
ENCRYPT	Returns 'YES' or 'NO' depending on whether the SAS data set is encrypted.

ENGINE	Returns the name of the engine that is used to access the data set.
LABEL	Returns the label assigned to the data set.
LIB	Returns the libref of the SAS library in which the data set resides.
MEM	Returns the SAS data set name.
MODE	Returns the mode in which the SAS data set was opened.
MTYPE	Returns the SAS library member type.
SORTEDBY	Returns an empty string if the data set is not sorted. Otherwise, it returns the names of the BY variables in the standard BY statement format.
SORTLVL	Returns a value that indicates how a data set was sorted.
SORTSEQ	Returns an empty string if the data set is sorted on the native computer or if the sort collating sequence is the default for the operating environment. Otherwise, it returns the name of the alternate collating sequence used to sort the file.
TYPE	Returns the SAS data set type.

Table 2: Attributes for ATTRC function

It is possible to access similar metadata through other methods, such as through the CONTENTS procedure, dictionary tables and data step techniques. In utilizing I/O functions with macros, we can access our dataset without having to enter a data step or call a SAS procedure, which increases the efficiency of our program.

ACCESSING OTHER EXTERNAL FILES

To access external files of other file types, a FILEREF must be created through the FILENAME function. A FILEREF is a reference that connects the external file to the SAS session. Once the FILEREF is assigned, the FOPEN function can be used to open the external file, so that it can be accessed by other external file functions. The FOPEN function returns a unique file identifier on success and returns 0 on failure. After finishing with the dataset, we must use the FCLOSE function to close the connection to the external file, and deassign the FILEREF.

With the connection to the external file opened, we can retrieve its' metadata through the FINFO function. This function takes a file ID, created via FOPEN, as well as the metadata attribute we want to view. The attributes available to the user differ between operating environments, so it is best to use the FOPTNAME function (determines the names of the available Metadata attributes) and FOPTNUM function (determines the number of system-dependent Metadata attributes that are available to the user) in conjunction with FINFO. We can utilize all these functions to gather all available metadata attributes of a file through this macro function:

```
%macro file_info(fname);
      %let rc=%SYSFUNC(FILENAME(ref, &fname.));
      %let fid=%SYSFUNC(FOPEN(&ref));
      %let infonum=%SYSFUNC(FOPTNUM(&fid));
      %do j=1 %to &infonum;
            %let name=%SYSFUNC(FOPTNAME(&fid, &j));
            %let value=%SYSFUNC(FINFO(&fid, &name));
            %put  Attribute &name is equal to &value;
      %end;
      %let rc=%SYSFUNC(FCLOSE(&fid));
```

```
        %let rc=% SYSFUNC(FILENAME(ref));
%mend;

%file_info(H:\MWSUG\IO_Example1.xlsx);
```

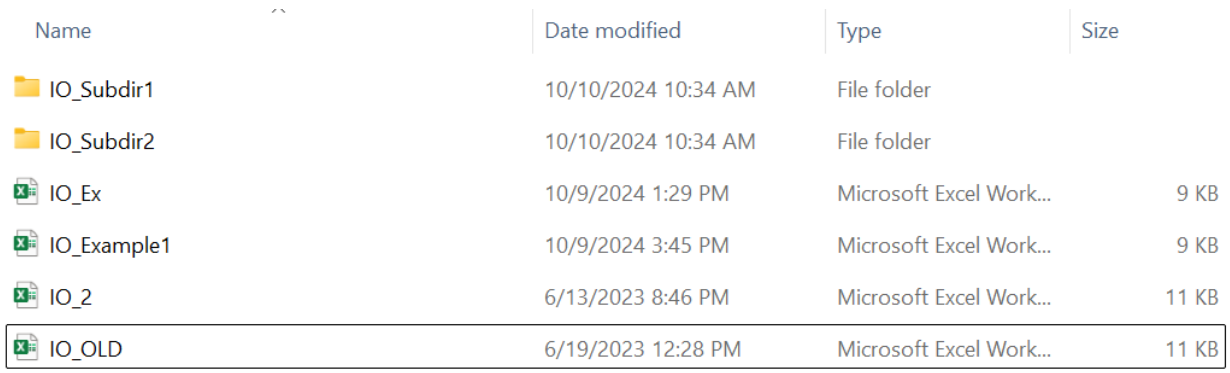
The resulting SAS Log:

```
Attribute Filename is equal to H:\MWSUG\IO_Example1.xlsx
Attribute RECFM is equal to V
Attribute LRECL is equal to 32767
Attribute File Size (bytes) is equal to 8900
Attribute Last Modified is equal to 09Oct2024:15:45:36
Attribute Create Time is equal to 09Oct2024:15:45:37
```

ACCESSING EXTERNAL DIRECTORIES

There are instances where SAS users will need to perform the same operation across all external files in a directory. To perform these operations, we utilize the DOPEN, DNUM, DREAD, and DCLOSE functions. The DOPEN function takes in a FILEREF and returns a unique directory identifier on success and 0 on failure. The DNUM function takes in the directory identifier created by the DOPEN function and returns the number of members in the directory. DREAD is used to retrieve the member names in the directory. When used with DNUM, we can gather the names of all members in a specific directory. Once we are done with our operation, we must use DCLOSE to close the connection to the folder directory, and deassign the FILEREF.

For example, given the below windows folder:



Name	Date modified	Type	Size
IO_Subdir1	10/10/2024 10:34 AM	File folder	
IO_Subdir2	10/10/2024 10:34 AM	File folder	
IO_Ex	10/9/2024 1:29 PM	Microsoft Excel Work...	9 KB
IO_Example1	10/9/2024 3:45 PM	Microsoft Excel Work...	9 KB
IO_2	6/13/2023 8:46 PM	Microsoft Excel Work...	11 KB
IO_OLD	6/19/2023 12:28 PM	Microsoft Excel Work...	11 KB

Figure 1: Windows Directory Contents

We can use the below SAS code to retrieve the contents of the above directory, and bring that into our SAS session:

```
%macro member_names(dir);

      %let rc=%sysfunc(FILENAME(ref, &dir.));
      %let did=%sysfunc(DOPEN(&ref));
      %do j=1 %to %sysfunc(DNUM(&did.));
            %let memname=%sysfunc(DREAD(&did, &j));
            %put Member number &j is &memname;
      %end;
      %let rc=% SYSFUNC(DCLOSE(&did));
      %let rc=% SYSFUNC(FILENAME(ref));
%mend;

%member_names(H:\MWSUG\)
```

The resulting SAS log:

```
Member number 1 is IO_Subdir2
Member number 2 is IO_Subdir1
Member number 3 is IO_Ex.xlsx
Member number 4 is IO_OLD.xlsx
Member number 5 is IO_2.xlsx
Member number 6 is IO_Example1.xlsx
```

IMPORTING ALL FILES IN A DIRECTORY

In some instances, a SAS user will need to import all files in a directory. This can be done through putting PROC IMPORT into a macro function like so:

```
%macro import(dir, fname, fext);
   PROC IMPORT
   datafile="&dir.\&fname."
   dbms=&fext.
   out=%QUPCASE(%QSCAN(&fname,1,.)) replace;
   run;
%mend;
```

Writing out each macro call with the different filenames is one way to go about this, but it is inefficient. Also, if files are added to the directory, we would have to go in and update our program to account for this. By using the external file functions discussed earlier, we can pass in our parameters into our macro function. For example, we can import all XLSX files in the H:\MWSUG directory with the below macro:

```
%macro import_all(dir,fext);
   %local filrf rc did memcnt fname i;
   %let rc=%SYSFUNC (FILENAME(filrf,&dir.));
   %let did=%SYSFUNC (DOPEN(&filrf));
   %do i = 1 %to %SYSFUNC (DNUM(&did));
       %let fname=%qsysfunc(DREAD(&did,&i));
       %if %QUPCASE(%QSCAN(&fname,-1,.)) = %UPCASE(&fext) %then %do;
           %import(&dir.,&fname.,&fext.);
    %end;
   %end;
   %let rc=%SYSFUNC(DCLOSE(&did));
   %let rc=%SYSFUNC(FILENAME(filrf));
%mend;

%import_all(H:\MWSUG,XLSX)
```

Through using external file functions, we have reduced the amount of manual code needed, while also making our program dynamic.

IMPORTING EXTERNAL FILES BASED ON DATE

When working with a program that runs routinely, external files that are used as input may be produced/updated periodically. When this happens, the SAS user may need to go in and update the program to point to the new file. Instead of doing this manual step, we can utilize external file functions to access a files date metadata.

Looking at the same directory from figure 1, let's say we want to access the most recently modified XLSX file. With the below code, we can create a SAS dataset that has the file name as well as the most recent modified date of the file.

```
%let dir=H:\MWSUG\;
DATA file_date;
   rc=FILENAME("mydir", "&dir.");
   did=DOPEN("mydir");
   do i=1 to DNUM(did);
       fname=DREAD(did,i);
          if upcase(SCAN(fname,-1,.)) = "XLSX" then do;
                rc=filename("fref", cats('"',"&dir",fname,'"'));
                fid=FOPEN("fref");
                moddate=INPUT(FINFO(fid, 'Last Modified'),datetime.);
                rc=FCLOSE(fid);
                rc=FILENAME("fref");
                OUTPUT;
          end;
   end;
   did=dclose(did);
format moddate datetime.;
drop rc did i fid;
RUN;
```

This code creates the following dataset:

Table: WORK.FILE_DATE | View: Column n

Total rows: 4 Total columns: 2

	fname	moddate
1	IO_OLD.xlsx	19JUN23:12:28:25
2	IO_Example1.xlsx	09OCT24:15:45:36
3	IO_Ex.xlsx	09OCT24:13:29:23
4	IO_2.xlsx	13JUN23:20:46:17

Figure 2: SAS dataset FILE_DATE in the WORK library

In this instance, we are only interested in the Last Modified attribute, but we can add other file attributes if needed.

Once we have each file's last modified date, we can apply additional logic to determine which file has the most recent modified date. This file name can then be passed into the import function:

```
DATA _NULL_;
      length max_fname $255 max_date 8;
      retain max_fname max_date ;
      set file_date end=eof;
      if max_date<moddate then do;
            max_date=moddate; max_fname=fname;
      end;
      if eof then output;
      CALL SYMPUTX('imp_file',max_fname);
run;
PROC IMPORT
      datafile="&dir.\&imp_file."
      dbms=xlsx
      out=%qupcase(%qscan(&imp_file,1,.)) replace;
RUN;
```

CONCLUSION

I/O functions allow the SAS user to gain useful metadata about their external files. They also allow the user to access SAS datasets efficiently, improving the runtime of SAS programs. We can also write dynamic programs that allow macro functions to apply to all files in specific directories, while also utilizing file metadata in data processing.

CONTACT INFORMATION

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

Jake Reeser

NORC at the University of Chicago

55 E Monroe

Chicago, IL 60603

Reeser-jake@norc.org