MWSUG 2024 Paper BB-041

Inventory your OS for Programming Information

Brian Varney, Experis

ABSTRACT

Whether you are attempting to figure out what you have when preparing for a migration or you just want to find out which files or directories are taking up all of your space, SAS is a great tool to inventory and report on the files on your desktop or server. This paper intends to present SAS code to inventory and report on the location you want to inventory.

INTRODUCTION

Trying to get a handle on the number of different types of files on a server or network drives can be daunting. Every year a server is in operation, there are opportunities for users to drop programs, data and other types of files across many directories. After a few years there can be thousands of files and many gigabytes of data. At some point the time will come for cleaning up or migrating to a new server. When this time comes, SAS can be a handy tool to inventory your files, summarize files by age, size, directory, etc. SAS can also be used to search through SAS programs for references to servers or statements that interact with Microsoft Excel that need to be changed. Depending on the details of the migration, there may be different results needed from the inventory.

The examples in this paper will be using Microsoft Windows and LINUX operating systems.

CONCEPTS

BUILDING A DATABASE OF FILES

Often, the first step in getting a handle on what files are on your server is to load the files and attributes into a SAS data set. Using pipes and operating system commands, SAS can interrogate a server or network drive and load the files and their attributes into a SAS data set.

The following code is an example of building a SAS data set containing all of the files on the c:junk\example\ directory of a Microsoft Windows desktop.

```
/*-------------------------------------------------------------
  Program: inventory_windows.sas
  Programmer: Brian Varney
  Date: 09/20/2023

  Purpose: To create a database of files from the file system
           recursively based on the base path passed in

-------------------------------------------------------------*/

%macro inventory_windows(basepath=, outds=final_fs_summary);

************************************************************;
** Turn on compression as datasets have large character
** variables.
************************************************************;
options compress=yes;

************************************************************;
** Ensure this macro variable stays local to the macro.
************************************************************;
```

```
%local pc;

*****************************************************************;
** Construct the pipecommand using data step to handle resolving
** macro variables within single quotes.
*****************************************************************;
data _null_;
   bp="&basepath.";
   pipecommand="' dir "||'"'||"&basepath."||'" /S /Q'||"'";
   put pipecommand;
   call symput('pc', pipecommand);
run;
%put &pc.;

*****************************************************************;
** Define pipe command in a filename using the pipe engine.
*****************************************************************;
filename pipedir pipe &pc. lrecl=5000;

*****************************************************************;
** Scan the file system and parse the input.
** Create the directory and date variables. If the date is
**  is missing it means it is a directory and we set it to
**  a very large date for sorting purposes.
*****************************************************************;
data initial_fs_read;
   infile pipedir truncover;
   input line $char1000.;
   length directory $1000;
   length owner $50;
   retain directory;
   if line =' ' or
      index(upcase(line),'<DIR>')    or
      left(upcase(line))=:'VOLUME'   then
     delete;
   if left(upcase(line))=:'DIRECTORY OF' then
   directory=left(substr(line,index(upcase(line),'DIRECTORY OF')+12));
   if left(upcase(line))=:'DIRECTORY OF' then
      delete;
   owner=scan(line, 5, ' ');
   if input(substr(line,1,10),?? mmddyy10.) = . then
      substr(line,1,10)='12/31/2999';
   date=input(substr(line,1,10),?? mmddyy10.);
   format date mmddyy10.;
run;

*****************************************************************;
** Sort by directory name and descending date so the directory
**  data is at the top.
*****************************************************************;
proc sort data=initial_fs_read;
   by directory descending date;
run;

*****************************************************************;
** Restructure the initial_fs data set. Create variables for:
**  date, filename, number of files in directory, directory size,
```

```
**  file size, and file extension.
*************************************************************;
data &outds.(drop=i line);
   set initial_fs_read;
   by directory;
   length filename $75;
   retain number_of_files_in_directory directory_size;
   if first.directory then
   do;
      number_of_files_in_directory=input(scan(line,2,' '),32.);
     directory_size=input(scan(line,4,' '),comma32.);
   end;
   file_size=abs(input(scan(line,4,' '),comma32.));
   filename=' ';
   do i=6 to 100;
     filename=trim(left(filename))||' '||scan(line,i,' ');
     if scan(line,i,' ')=' ' then
        leave;
   end;
   fileext = scan(filename, -1, '.');
   if index(upcase(line),'FILE(S)') then
      delete;
   if date ge '30DEC2999'd then
      delete;
run;

proc sql;
   drop table initial_fs_read;
quit;

%mend inventory_windows;


%inventory_windows(basepath=%str(C:\junk\example));
```

The following code is an example of building a SAS data set containing all of the files on the /opt/myfiles/ directory of a LINUX server.

```
/*-------------------------------------------------------------
  Program: inventory_linux.sas
  Programmer: Brian Varney
  Date: 09/20/2023

  Purpose: To create a database of files from the file system
           recursively based on the base path passed in

-------------------------------------------------------------*/

%macro inventory_linux(basepath=, outds=final_fs_summary);

*************************************************************;
** Ensure this macro variable stays local to the macro.
*************************************************************;
%local pc;

*************************************************************;
** Construct the pipecommand using data step to handle resolving
```

```
** macro variables within single quotes.
*************************************************************;
data _null_;
   bp="&basepath.";
   pipecommand="' ls -laR "||"&basepath."||"'";
   put pipecommand;
   call symput('pc', pipecommand);
run;
%put &pc.;

filename pipedir pipe &pc. lrecl=5000;

*************************************************************************;
** read the results into a SAS data set leveraging the fileref         **;
** defined above.                                                      **;
*************************************************************************;
data &outds.;
   infile pipedir truncover;
   retain directory;
   input line $char1000.;
   if line NE ' ';
   if substr(lowcase(line),1,5) ne 'total';
   if line=:'/' and substr(line,length(line),1)=':' then
   do;
      directory=substr(line,1,length(line)-1);
   end;
   else if substr(line,1,1)='d' then
   do;
   end;
   else do;
      filename=scan(line,-1,' ');
      permissions=scan(line,1,' ');
      mystnum=scan(line,2,' ');
      owner=scan(line,3,' ');
      group=scan(line,4,' ');
      size_bytes=scan(line,5,' ');
      year=scan(line,8,' ');
      if index(year,':') then year=put(year("&sysdate9."d),best12.);
      date=input(trim(left(scan(line,7,' ')))||trim(left(scan(line,6,' ')))||
          trim(left(year)),date9.);
      fileext = scan(filename, -1, '.');
      output;
   end;
   format date mmddyy10.;
run;

%mend inventory_linux;

%inventory_linux(basepath=%str(/opt/myfiles/));
```

Figure 1: Excerpt from final_fs_summary dataset

VIEWTABLE: Work.Final_fs_summary

	directory	owner	date	filename	number_of_files_in_director	directory_size	file_size	fileext
1	C:\junk\example	ZB15-CND1054H3B\User	09/25/2023	registrants.xlsx	16	124984124	102108	xlsx
2	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	Academic Papers Report.sas	16	124984124	22499	sas
3	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	final.zip	16	124984124	123965301	zip
4	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	Final_Submission_Inventory.sas	16	124984124	20181	sas
5	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	notregistered.xlsx	16	124984124	7793	xlsx
6	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	Submission_Information.xlsx	16	124984124	147945	xlsx
7	C:\junk\example	ZB15-CND1054H3B\User	09/21/2023	tutorials.xlsx	16	124984124	6086	xlsx
8	C:\junk\example	ZB15-CND1054H3B\User	09/20/2023	SESUG 2023 Grant Recipient Emails.xlsx	16	124984124	11306	xlsx
9	C:\junk\example	ZB15-CND1054H3B\User	09/11/2023	no_p2p.xlsx	16	124984124	6929	xlsx
10	C:\junk\example	ZB15-CND1054H3B\User	08/29/2023	registrants.csv	16	124984124	109484	csv
11	C:\junk\example	ZB15-CND1054H3B\User	05/31/2023	Academic_Paper_Summmary_30MAY23.ht	16	124984124	108284	html
12	C:\junk\example	ZB15-CND1054H3B\User	05/31/2023	Academic_Paper_Summmary_30MAY23.p	16	124984124	97029	pdf
13	C:\junk\example	ZB15-CND1054H3B\User	05/03/2023	Academic_Paper_Summmary_03MAY23.ht	16	124984124	92921	html
14	C:\junk\example	ZB15-CND1054H3B\User	05/03/2023	Academic_Paper_Summmary_03MAY23.p	16	124984124	96941	pdf
15	C:\junk\example	ZB15-CND1054H3B\User	05/02/2023	Academic_Paper_Summmary_02MAY23.ht	16	124984124	92916	html
16	C:\junk\example	ZB15-CND1054H3B\User	05/02/2023	Academic_Paper_Summmary_02MAY23.p	16	124984124	96401	pdf

LOADING PROGRAMS INTO A SAS DATA SET

Another useful thing to do is to load all of the SAS programs into a data set. This will allow us to search for strings and if brave enough, alter the programs programmatically. The code below will loop over each distinct directory from the directory summary data set created above (see Figure 1) and load the programs into one data set. Each line of the program will be a record in the data set.

```
/*-------------------------------------------------------------
  Program: get_sas_programs.sas
  Programmer: Brian Varney
  Date: 09/20/2023

  Purpose: To create a database of SAS programs from the file system.

-------------------------------------------------------------*/

%macro get_sas_programs(inds=, outds=);

proc sql noprint;
  select distinct directory into :dir1 - :dir9999
  from &inds.
  where lowcase(fileext)='sas';
quit;
%let numdirs=&sqlobs.;
%put &numdirs.;

proc datasets nolist lib=work;
   delete &outds.;
quit;

%do i=1 %to &numdirs.;

data prgs;
  length line sasfile f $300;
  infile "&&dir&i.\*.sas" FILENAME=f;
  input;
  sasfile =f ;
```

```
  linenum+1;
  if lag(sasfile) ne sasfile then
    linenum=1;
  line=_infile_;
run;

proc append base=&outds. data=prgs;
run;

%end;

%mend get_sas_programs;


%get_sas_programs(inds=final_fs_summary, outds=allprgs);
```

Figure 2: Excerpt from allprgs dataset

	line	sasfile	linenum
1		C:\junk\example\Academic Papers Report.sas	1
2	options nodate nonumber;	C:\junk\example\Academic Papers Report.sas	2
3	options orientation=portrait;	C:\junk\example\Academic Papers Report.sas	3
4		C:\junk\example\Academic Papers Report.sas	4
5	%let basepath=C:\Users\User\OneDrive - ManpowerGroup_ExperisStuff\SAS_Stuff\SUGI_Stuff\2023_SESUG;	C:\junk\example\Academic Papers Report.sas	5
6		C:\junk\example\Academic Papers Report.sas	6
7	data _null_;	C:\junk\example\Academic Papers Report.sas	7
8	call symput('currdatetime',trim(left(put(date(),weekdate.)))||' at '||trim(left(put(time(),timeampm.))));	C:\junk\example\Academic Papers Report.sas	8
9	run;	C:\junk\example\Academic Papers Report.sas	9
10	%put &currdatetime.;	C:\junk\example\Academic Papers Report.sas	10
11		C:\junk\example\Academic Papers Report.sas	11
12	proc import file="&basepath.\data_and_reports\submission_information.xlsx"	C:\junk\example\Academic Papers Report.sas	12
13	dbms=xlsx	C:\junk\example\Academic Papers Report.sas	13
14	out=so_initial	C:\junk\example\Academic Papers Report.sas	14
15	replace;	C:\junk\example\Academic Papers Report.sas	15
16	run;	C:\junk\example\Academic Papers Report.sas	16
17		C:\junk\example\Academic Papers Report.sas	17

APPLICATION #1: CREATING A LIST OF UNNEEDED FILES

After running the code for creating the database of files on the location that you passed into the program, you will have a data set with the fields listed below:

A simple SQL query can be specified to find the old (older than January 1st, 2009) and large (approximately larger than 10GB) files:

```
proc sql;
   select *
   from final_fs_summary
   where file_size ge 10000000000 and
         date le "01JAN2009"d;
quit;
```

APPLICATION #2: SEARCHING FOR REFERENCES TO A MACRO CALL.

After running the code for creating the database of SAS programs, you will have a data set with the fields listed below:

A simple SQL query can be specified to find any references to a macro called experis:

```
proc sql;
   select *
   from allprgs
   where lowcase(line) ? '%experis' or
         lowcase(line) ? '%macro experis';
quit;
```

APPLICATION #3: FIND ALL OF THE SAS ENTERPRISE GUIDE PROJECTS

After running the code for creating the database of files on the location that you passed into the program, you will have a data set with the fields listed below:

A simple SQL query can be specified to find the SAS Enterprise Guide projects:

```
proc sql;
   select *
   from final_fs_summary
   where scan(lowcase(filename),-1,'.') = 'egp';
quit;
```

APPLICATION #4: SEARCHING FOR PROGRAMS RELATED TO PROTOCOL X/0001/0001.

After running the code for creating the database of SAS programs, you will have a data set with the fields listed below:

A simple SQL query can be specified to find any references to protocol x/0001/0001.:

```
proc sql;
   select *
   from allprgs
   where lowcase(line) ? 'x/0001/0001' or
         (lowcase(sasfile) ? 'x' and
          lowcase(sasfile) ? '0001' and
          lowcase(sasfile) ? '0001');
quit;
```

CONCLUSION

SAS is a great tool to interrogate and report on data retrieved from your operating system regarding to the files on your disk drives. Whether you are preparing for a migration, want to find older large files not needed any more or need to search through your SAS programs for specific strings, SAS is a powerful tool at your disposal.

CONTACT INFORMATION <HEADING 1>

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

Brian Varney

Experis, a Manpower Company

Portage, Michigan

Work Phone: (269) 365-1755

Email: Brian.Varney@experis.com