

Which program use that dataset ?

Helen Nicholson, Cmed Clinical Services



1. Introduction

You have just been told that the data has changed for one dataset and you need to know which program use it so they can be re run. You have finished all your programs and would like to do a final check on the log files to make sure that there is nothing in them that should not be there.

Both these are situations where you need to be able to quickly search some files—you may have a search facility to do this easily but if not then how? You already have SAS so is there not a way you can use that ?

The following is one way of doing it— written as a macro, using the file handling functions within SAS, you just pass in the directory to be searched, the file type to be looked at and optionally what to look for and the type of report you want produced at the end.

2. Check the path

The first thing to do is to check the path—there is no point continuing if you have typed in the path name wrong

```
%if %sysfunc(fileexist(&dir))=0 %then %do ;
  %put ;
  %put ERROR: The directory &dir does not exist.;
  %put ;
  %abort ;
%end ;
```



3. Read the directory

Open the directory and work out how many files are in there. Loop through the list of files, checking if they are of type you want to check. If the file extension matches the type that you want to search then output the filename.

```
data list_dir ;
  rc=filename("filrf",&dir);
  if rc =0 then do ;
    did=dopen("filrf") ;
    memcnt=dnum(did);
    do i = 1 to memcnt;
      name=scan(dread(did,i),-1,'. ');
      if upcase(dread(did,i)) ne upcase("&ext") then do;
        if ("%superq(ext)" ne "" and upcase(name)=upcase("&ext")) or ("%superq(ext)" = "" ) then
          do;
            file_n = dread(did,i);
            output ;
          end;
        end;
      end;
    end ;
  else do ; sysmes=sysmsg() ; put sysmes= ; end ;
  rc_d=dclose(did);
run ;
```



4. Read in the files

Using the list of files you created above read in the contents of the files to a dataset. Each line is read in as an observation. You will need to attach the directory name to the file, and check the return codes to make sure that it is all going to plan.

```
data read_file ;
  set list_dir ;
  length file_n2 $200 ;
  file_n2=ccompress("&dir./"||file_n) ;
  length f_line $200. ;
  rc_fn=filename('f_ref',file_n2) ;
  if rc_fn =0 then do ;
    rc_o=fopen('f_ref') ;
    if rc_o gt 0 then do while (fread(rc_o)=0) ;
      rc_g=fget(rc_o, f_line,200) ;
      output ;
    end ;
  end ;
  rc_c=fclose(rc_o) ;
run ;
```



5. What to search for

In order to decide if the text you are looking for is in the file or not each line of the file needs to be checked to see if the text is in the line. The index function is used on the variable that contains the line of the file, if the text is found then the line is output. Before you can do this you need to know what text you are searching for. If no text has been passed in then the code will look for the default text, in the example given here it is things that you do not want in the log files.

If you have passed in what you are going to search for then you need to know how many bits of text you are looking for. This is found by, if the parameter string is greater than one, replacing the ' ' with '.', then using a do while loop and the scan function to add one to the count every time a new word is found. The final count is put into a macro variable which is then used in the next dataset.

```
%if %length(&srch_for)> 1 %then %do ;
  data _null_ ;
  wrd_lst="inital" ;
  i=0 ;
  repl=tranwrd("&srch_for"," ",".",'') ;
  do while (lengthn(wrd_lst)>0) ;
    i=i+1 ;
    wrd_lst=scan(ccompress(repl),i,',') ;
    output ;
  end ;
  call symputx('numwrd',i-1) ;
run ;
%end ;
```



6. Searching the file

All the text that you are going to search for is put in a temporary array with the text being set as the initial values of the elements. The code either sets up an array using the count created above to specify how many elements there are and the parameter list as the initial values, or, if no search for text was passed through, uses the default array.

The code then loops through the array, using the index function to see if the element is in variable that contains the line of the file you are checking. Put the element you are looking for into a variable so that you know which text has been found. If the index function returns a value greater than 0 then the text has been found so output the observation. The code upper cases both the text it is looking for and the text it is searching because I am not interested in the case.

```
data search ;
  set read_file ;
  %if &srch_for eq %then %do ;
    array srch (13) $50 _temporary_ ('error' 'warning' 'trunc' 'has 0 obs'
      'uninitialized' 'repeats of by' 'no by statement' 'does not exist' 'outside the axis'
      'missing value' 'converted' 'overwritten' 'then delete' ) ;
  %end ;
  %else %do ;
    array srch (&numwrd) $50 _temporary_ (&srch_for.) ;
  %end ;
  do i=1 to dim(srch) ;
    found=srch[i] ;
    if index(upcase(f_line), strip(upcase(srch[i])))> 0 then output ;
  end ;
run ;
```



7. Producing a Report

The macro is written to produce two types of reports, one that lists the file, what is found and the line the text is found in, the long report, or a more concise one that just lists the file and what is found. The default is to produce both reports. To produce only one of the report then pass in either short or long depending on which report is wanted.

If the long report is wanted sort the data by the file name, the line the text was found on and what was found, removing duplicate observations based on these variables. Find out how many observations this results in and put this into a macro variable. If nothing was found then then create a blank observation so that there is something to put out in the PROC REPORT.

```
%if &report ne short %then %do ;
  proc sort data=search out=found nodupkey ;
    by file_n f_line found ;
  run ;

  data _null_ ;
  if 0 then do;
    set found nobs=count;
  end;
  call symputx("nobs",count);
  stop;
run;

%if &nobs=0 %then %do;
  data found ;
    file_n=' ' ; found=' ' ; f_line=' ' ;
  run ;
%end ;
```



8. The Long Report

Use PROC REPORT to create a report. There are only three columns to put out, the file that something was found in, what was found and the line it was found in. Use an order option on the define statement for the file name so it is easy to see which files have something found in them.

If nothing was found then the compute block puts out a message to say nothing was found.

```
title2 "The following have been found in &dir " ;
Proc report data=found nowd split='#' missing headline;
  column file_n found f_line ;
  define file_n / order left width=20 flow 'file name' ;
  define found / display left width=15 flow 'Word found' ;
  define f_line / display left width=80 flow 'On line' ;
%if &nobs=0 %then %do;
  compute after file_n ;
    line @25 ' ' ;
    line @25 ' ' ;
    line @25 '*****';
    line @25 '* Nothing found *';
    line @25 '*****';
    line @25 ' ' ;
    line @25 ' ' ;
    line @25 ' ' ;
  endcomp;
%end ;
run ;
```



9. Setting the Short Report

The short report is very similar to the long report, the difference is that when sorting the data sort it based on the file name and what was found only, removing duplicates based on this sort selection.

Check how many observations are in the resulting dataset, putting the number into a macro variable and as before if no observations were found create a blank and empty observation..

```
%if &report ne long %then %do ;
  proc sort data=search out=found2 nodupkey ;
    by file_n found ;
  run ;

  data _null_ ;
  if 0 then do;
    set found2 nobs=count;
  end;
  call symputx("nobs2",count);
  stop;
run;

%if &nobs2=0 %then %do;
  data found2 ;
    file_n=' ' ; found=' ' ;
  run ;
%end ;
```



10. The Short Report

The PROC REPORT this time only has two variables, the file name that something was found in and what was found. Again put an order option on the define statement for the file name, and use a compute block to put out a message if nothing was found.

```
title2 "The following have been found in &dir " ;
title3 "File name and word only " ;
Proc report data=found2 nowd split='#' missing headline;
  column file_n found ;
  define file_n / order left width=20 flow 'file name' ;
  define found / display left width=15 flow 'Word found' ;
%if &nobs2=0 %then %do;
  compute after file_n ;
    line @25 ' ' ;
    line @25 ' ' ;
    line @25 '*****';
    line @25 '* Nothing found *';
    line @25 '*****';
    line @25 ' ' ;
    line @25 ' ' ;
    line @25 ' ' ;
  endcomp;
%end ;
run ;
```



11. Calling the Macro

Now you have the macro just pass in the path, what you are looking for and run the code. So to list all the SAS programs in aaaa/bbbb/cccc/programs/sdtm that use the SDTM>AE dataset use the following call.

```
%list_dir(dir=aaaa/bbbb/cccc/programs/sdtm, ext=sas, srch_for='sdtm.ae', report=short)
```

To search through the log files in a given directory for WARNING and ERROR, producing both reports use

```
%list_dir(dir=aaaa/bbbb/cccc/logs/sdtm, ext=log, srch_for='error' 'warning', report=)
```

While searching the log files for the default search strings listing all occurrences of the text found use

```
%list_dir(dir=aaaa/bbbb/cccc/logs/sdtm, ext=log, srch_for=, report=long)
```

12. And finally

This code was developed using SAS Enterprise Guide on a PC submitting the code to SAS on a Solaris server.

The program, along with several that do similar things, is stored in an Enterprise Guide project, so if the question is asked then I can just start the project, open the file and pass in the parameters.

Equally the code can also just be called into any other session and ran that way.

Contact details : Helen Nicholson
Cmed (Clinical Research Services) Ltd, Holmwood, Broadlands Business Campus, Langhurstwood Road
Horsham, West Sussex, RH12 4QP, UK
Email: hnicholson@cmedresearch.com