

A Program Which Displays the Whole Line for Each Line Containing a Given Text Search String

Hugh Geary, STATPROBE, Inc., Dublin, OH

Abstract

Have you ever needed to know which of all of your programs use a certain variable, call a certain macro or, more generally, contain a certain character string? Would it be useful to know which of these programs are old or were modified recently? And would you want to have this information for all of the files in a folder and all of its subfolders?

There used to be such a utility in VMS on the VAX, but now for PC SAS® it has to be reinvented. Lines before and after can be displayed as well in order to help locate the string within the file. The results can be saved in a file or displayed on the screen. An application can be devised for extracting up-to-date program documentation stored within all of your programs into a documentation file at, say, the end of a project.

Introduction

This paper describes how to build a program that will find all occurrences of a given string in all of the files in a folder and in all of the files in the subfolders; i.e. throughout the whole branch. For every file in which a hit is made it lists the file name, the date and time it was last saved, the number of lines in the file and, then, displays the lines containing the string. It also displays lines preceding and following the line with the hit as requested; one before and after, two before and after, etc.

It uses a DOS command to gather together the names of all of the files contained in the branch, the index function to identify the target lines within a file and an algorithm to select and save “nearby” lines, if requested, for printing as well.

The DOS Directory Command, DIR

```
x "C:\sp\cmd.exe /c  
  dir &Dir_nam/s/a:-d/t:w>abc";
```

The above command, normally written all on one line, is an MS DOS command which is executed as a host command from within a SAS session though use of the SAS “x” statement. It creates a list of all of the file names residing the folder whose name is supplied

by the macro variable, &Dir_nam, and all of its subfolders.

The /s parameter causes it to display files in the subfolders as well. The /a:-d parameter causes it to list no subfolder names. We want only file names, fully qualified complete file names beginning with the drive name, as

d:\hij\def\xyz.sas

The /t:w causes it to supply the date and time the file was last modified. The >abc causes the output of the Dir command to be written to a temporary file, abc, which will be read later through the use of a filename statement.

At the beginning of the command there appears

C:\sp\cmd.exe /c

which may only be needed by Window NT users and even then not for all DOS commands although it is not wrong to supply it.

Combine Dir Info with Info from each File to Create the Output Display

Having created a list of file names that also contains the dates and times the files were last modified, it remains to read through each of those files to seek occurrences of the search string.

A macro, **serch**, is employed to process one file per invocation. It will return each record containing a ‘hit’ and a number of preceding and following records, if requested. Along with each record is passed its line number. Records containing hits also have arrows, →, unless no nearby records were requested. The line, line number and arrow are combined with the information from the Dir command, namely the file name and date and time last modified into one record in a SAS data set which can then be displayed in the output screen or printed.

The Serch Macro Call

%serch(&strg,&FQ,mm=&mm)

The values &strg is the string to be found, &FQ is the fully qualified name of the file to be searched and &mm is the number of nearby lines above and below to be displayed as well as the target line. The values returned are in a SAS data set.

Selecting and Saving the Neighboring Lines

If &mm=0 then there are no neighboring lines to be displayed and all that needs to be done is to return the target lines and their line numbers. Otherwise these neighboring lines must be kept in a sequence of retained variables, dropping the oldest one and picking up a new one each time a record is read and checked with the index function.

The algorithm for the collecting the nearby lines must allow for the occurrence of a second target line being found within the neighborhood of a previous target line or even just the case of two neighborhoods overlapping. The code for the **serch** macro follows.

The Code for Extracting the Lines to be Displayed

```
%macro serch(strg,dsx,mm=1);
* mm=# of lines to display before &
after the target line;
• mm may be set to 0,1,2 ...etc.;
filename dd "&dsx";

data aa (keep=arrow linum x&mm
rename=(x&mm=xx));
infile dd length=ln end=last;
length linum 8 arrow $3;
length x0-x%eval(2*&mm) $120.;
retain todo 0 x0-x%eval(2*&mm) ' ';
if _n_=1 then do;
  %do ii=&mm %to 2*&mm;
    input x&ii $varying120. ln;
  %end;
end;

if index(x&mm,"&strg")>0 then do;
  if todo=0 then do;
* catch up, print the previous &mm
records;
    %do jj=0 %to &mm;
      if &jj>=2*&mm-_n_-1 then do;
        linum=max(1,_n_-&mm+&jj);
        if &jj=&mm & &mm>0 then
arrow='-->';
        xxx=x&mm; x&mm=x&jj;
        output;
      %end;
    %end;
  %do jj=0 %to 2*&mm-1;
    %shift all indexes down by one, ;
* dropping the last;
    if &jj>=2*&mm-_n_-1 then do;
      x&jj=x%eval(&jj+1);
    %end;
    input x%eval(2*&mm) $varying120. ln;
* bring in a new one at the top;
    if last then call
symput('nnn',left(put(_n_,5)));
  %mend serch;

* The following are for validation;
%*serch (merge,
d:\abc\utils\cros_ref.sas,mm=2);
%*serch (_null_,
d:\abc\utils\lister.lis,mm=0);
```

```
if &mm=0 then do;
  linum=.; x0=' '; output;
end;
* Writing a blank line here because
it does not happen below for the m=0
case;
  x&mm=xxx;
end;
%end;
end;
else do;
  linum=_n_;
  arrow='-->';
  output;
* printing the current x&mm;
end;
todo=&mm;
* reset to the max value;
end;
else do;
  if todo>0 then do;
* print another of the following &mm
lines;
    linum=_n_;
    output;
    if todo=1 then do;
      linum=.;
      xxx=x&mm;
      x&mm='';
      output;
      x&mm=xxx;
    * Writing a blank line;
    end;
    todo=todo-1;
  end;
end;

%do jj=0 %to 2*&mm-1;
* shift all indexes down by one, ;
* dropping the last;
  if &jj>=2*&mm-_n_-1 then do;
    x&jj=x%eval(&jj+1);
  %end;
  input x%eval(2*&mm) $varying120. ln;
* bring in a new one at the top;
  if last then call
symput('nnn',left(put(_n_,5)));
%mend serch;

* The following are for validation;
%*serch (merge,
d:\abc\utils\cros_ref.sas,mm=2);
%*serch (_null_,
d:\abc\utils\lister.lis,mm=0);
```

Example of Output Produced by the Search Program

Searching for " merge " Throughout the d:\abcd\utils Folder

Using macro Serch

Produced: 21:38 Sun, Jul 11, 1999

```
d:\abcd\utils\CROS_REF.SAS          209 records          last modified: 06/24/99 07:19p

110      data glob0 (keep=file modate modtime nobobs nvars);
111  --> merge glob0 (rename=(varnum=nvars modate=Mdate))
112      %if %upcase(&typ)=DBF %then %do; DBFnames %end;;

135
136  --> data glob2; merge glob (drop=modate) glob0 (in=g0); * adding nvars modate;
137      by file;                                     * and modtime to glob;

d:\abcd\utils\lister.lis          541 records          last modified: 06/24/99 07:52p

225      *options symbolgen mlogic mprint;
226  --> data _null_; merge cnts var_len end=last; by varnum;
227      retain MxNamLn 0 nlpobsH 1 nlpobsV 1;
```

Create Documentation on a Folder of Programs by 'Retrieving' Specially Tagged Comment Lines Maintained in the Program

Keeping documentation about a program inside the program is onerous enough without trying to keep up-to-date documentation in another location. So if you keep your up-to-date documentation inside your programs and use special symbols, such as */ or ++ etc. at, say, the beginning or end of the line and don't use these symbols elsewhere in the program, then this search program can extract documentation from every program in the project folder and put it in one document. The document can be recreated in seconds whenever it is needed.

Find Every Program that Uses a Certain Macro

Occasionally a macro may need to have its call statement modified or it may be necessary to know what programs utilize a certain macro. The search program can find all occurrences of the macro throughout any drive or branch thereof.

Conclusions

The search program with the macro, Serch, constitutes a utility that is handy and useful for large programming projects and program management. It may enable the user to find lost code, provided some fairly unique identifying name or phrase was used with it and it may save a considerable amount of time in finding every possible occurrence of a name.

Author:

Hugh Geary
STATPROBE, Inc.
425 Metro Place North
Dublin, Ohio 43017

(614)791-7900 x302
(614)791-7904 fax
hugh.geary@co.statprobe.com (email)

SAS is a registered trademark of SAS Institute, Inc. ®
indicates USA registration.