

A SAS® Macro to Find and Replace

David Brennan, Independent Contractor, Dungarvan, Ireland

ABSTRACT

Left with the task of making repetitive changes to a plethora of SAS® programs, and the absence of a standard text editor to perform the usual find and replace functionality over an entire folder, my choice was to accept my lot, knuckle down, and do it by hand, or, to write a SAS macro. The decision was simple – I wrote a macro, the %FIND_REPLACE macro.

INTRODUCTION

The %FIND_REPLACE macro allows one to search flat files for one or more text strings and, if required, replace them with others. It provides various features including the ability to perform case-sensitive searches as well as the option of searching subfolders. Before using such a macro, one would like to be sure that the macro will do what is intended, and if not, that there is a mechanism in place to protect against the loss or corruption of information. The macro was designed with this in mind and uses the WinZip® Command Line Add-On to make a backup of files before they are updated. Since WinZip® software is used within a Microsoft® Windows operating system, the %FIND_REPLACE macro is restricted to this environment. Although not without limitations, the macro has proved a useful tool, used primarily to help manage large suites of related SAS programs. One of the most beneficial aspects of using the macro is that the results are stored in SAS data sets. This allows its integration into batch processes as well as facilitating the storage and reporting of results.

SET UP

The %FIND_REPLACE macro relies on the availability of the WZZIP and WZUNZIP commands, the installation of which is described in this year's Coder's Corner paper "A Quick Guide to the WinZip® Command Line Add-On".

The %FIND_REPLACE SAS macro is contained in the **FR_Test.zip** zip file, available for download from <http://david.n.brennan.googlepages.com/phuse2006downloads>.

Before using the %FIND_REPLACE macro, there are two modifications that may need to be made to the default values of two of its parameters – WZ_FOLDER and FOLDER. The WZ_FOLDER has been set with a default value of **C:\Program Files\WinZip** and is used to specify the location where the WinZip Command Line Add-On was installed, or more specifically, where the WZZIP.exe and WZUNZIP.exe files are located. By default, the FOLDER parameter is set to **C:\FR_Test\FR_Output** and is used to specify the folder to which the macro writes both temporary files (batch and text files) and permanent files (zip files and text file reports). These default values should be changed as required.

EXAMPLE CALLS TO %FIND_REPLACE

This section gives a number of example calls to the %FIND_REPLACE macro. Each example is accompanied by a description of what that particular macro call is used to achieve. It is hoped that presenting the macro as such will give the reader a quick understanding of what the macro is about.

1. What follows is a call to the macro in its simplest form:

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\AE1.sas  
  ,Find = proc datasets  
);
```

The FILES parameter specifies where to search and the FIND parameter specifies the string to be searched for. The above macro call will search the **C:\FR_Test\Programs\AE1.sas** file for the text "proc datasets". The search will not be case-sensitive. The results of the search will be captured in a data set in the WORK library of the SAS session and a simple text file report is written to the **C:\FR_Test\FR_Output** folder (or whatever folder

PhUSE 2006

is given as the OUTFOLDER parameter default value).

2. The following macro call will search for the text “proc datasets” in any file in the **C:\FR_Test\Programs** folder whose name begins with the characters “AE”, which is then followed by a single character of some kind, and has the **.sas** extension.

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\AE?.sas  
  ,Find = proc datasets  
);
```

Note that two wildcard characters, “?” and “*”, are available for use with the **%FIND_REPLACE** macro as they normally would be with MS DOS. The “?” is used to denote a single character and “*” to denote any series of characters.

3. Therefore, the following macro call will search all files in the **C:\FR_Test\Programs** for the text “proc datasets”:

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\*.*  
  ,Find = proc datasets  
);
```

4. Consider the following macro call:

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\*.* | C:\FR_Test\Data\*.*  
  ,Find = proc datasets  
);
```

This macro call will search all files in the folders **C:\FR_Test\Programs** and **C:\FR_Test\Data** for the text “proc datasets”. Note the “|” character delimiting the two file paths. It is possible to specify several file paths in the FILES parameter. They do not need to refer to the same drive. (The use of a vertical bar to delimit the file paths is due to the FILESDLM parameter having this character as its default value.)

5. Searching subfolders:

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\*.* | C:\FR_Test\Data\*.*  
  ,SubDirs = Y | N  
  ,Find = proc datasets  
);
```

The SUBDIRS macro parameter is used to control whether or not subfolders are considered for searching. When more than one file path appears in the FILES parameter specification, for each file path there must be a “Y” or an “N” character present in the SUBDIRS parameter. A “Y” character denotes that subfolders should be considered and “N” that they be ignored. Each character should be delimited using the “|” character, the same way as with the FILES parameter. The above macro call will search for the text “proc datasets” in all files in the **C:\FR_Test\Programs** and **C:\FR_Test\Data** folders. Subfolders of **C:\FR_Test\Programs** will be considered, but not for **C:\FR_Test\Data**.

6. Case-sensitive search:

```
%Find_Replace(  
  Files = C:\FR_Test\Programs\*.*  
  ,Find = PROC datasets  
  ,Case_sens = Y  
);
```

By default, the text searches are not case-sensitive. To perform case-sensitive searching, the CASE_SENS parameter must be set to “Y” as shown in this macro call.

7. Multiple searches:

```
%Find_Replace(
  Files      = C:\FR_Test\Programs\*.*
  ,Find      = PROC datasets|nolist
);
```

The macro call above will perform two searches on all files in the **C:\FR_Test\Programs** folder, one for the text “PROC datasets” and another for “nolist”. Note that the two search strings are delimited by the “|” character. If it were required to include the “|” character in a search string, the FINDREPDLM parameter can be changed to achieve this. For example, the following call will perform exactly the same search:

```
%Find_Replace(
  Files      = C:\FR_Test\Programs\*.*
  ,Find      = PROC datasets#nolist
  ,FindRepDLM = #
);
```

The FINDREPDLM parameter specifies the delimiter to use when parsing the FIND parameter and by default is set to “|”. As can be seen here, the hash symbol overwrites the FINDREPDLM value and can then be used as a delimiter in the FIND parameter.

For each search string it is possible to specify whether the search be case-sensitive or not. This is done by specifying a “Y” or “N” character in the CASE_SENS parameter, one for each search string. By default, these characters are (once again) delimited using the “|” character. An example follows:

```
%Find_Replace(
  Files      = C:\FR_Test\Programs\*.*
  ,Find      = PROC datasets|nolist
  ,Case_Sens = Y | N
);
```

This macro call specifies that the search for “PROC datasets” be case-sensitive and that the search for “nolist” not be case-sensitive.

Note that the FINDREPDLM parameter, whose default value is “|”, specifies the delimiting character for three macro parameters – FIND, CASE_SENS and REPLACE (mentioned later).

8. Masking certain characters:

Characters that require masking using the %STR and %NRSTR functions also require masking with the %FIND_REPLACE macro. The following is an example showing how characters can be masked:

```
%Find_Replace(
  Files = C:\FR_Test\Programs\*.*
  ,Find = %nrstr(Look at SAS%'s Online Help, %STR & %NRSTR)
);
```

This macro call will search for the text “Look at SAS's Online Help, %STR & %NRSTR”. The characters requiring masking include the ampersand, semi-colon, comma, unmatched quotes, unmatched parentheses and percent signs. In addition spaces must be masked. For more details, the reader is directed to the SAS Online Help facility, in specific to the entry given for the %STR and %NRSTR macro functions.

Note that spaces can be masked without using the %STR and %NRSTR functions. The delimiter can be used to do this as follows:

```
%Find_Replace(
  Files      = C:\FR_Test\Programs\*.*
  ,Find      = | PROC datasets |
);
```

With the call above, the string “ PROC datasets ” is searched for (two spaces before “PROC” and two after “datasets”).

9. A find and replace action can also be performed using the %FIND_REPLACE macro. The REPLACE macro parameter is used to specify a text string to replace the search string given by the FIND parameter. A simple

example follows:

```
%Find_Replace(
  Files    = C:\FR_Test\Programs\*.sas
  ,Find    = proc print
  ,Replace = proc report
);
```

The above macro call will replace the text “proc print” with “proc report” in all files in the **C:\FR_Test\Programs** folder which have the **.sas** extension.

Note that before any file is amended using the replace feature, a backup is made by placing the file in a zip file (the process is explained later).

10. It is possible to specify multiple search strings in the FIND parameter and couple them with the same number of replace strings in the REPLACE parameter, as shown below:

```
%Find_Replace(
  Files      = C:\FR_Test\Programs\*.sas
  ,SubDirs   = Y
  ,Find      = PROC print|ODS rtf|proc sql noprint%str(;)
  ,Replace   = PROC report|ODS pdf|proc sql%str(;)
  ,Case_Sens = Y | N | N
);
```

This macro call will search all files with the **.sas** extension in the **C:\FR_Test\Programs** folder and its subfolders for the text strings “PROC print”, “ODS rtf” and “proc sql noprint;”, and will replace them with “PROC report”, “ODS pdf” and “proc sql;” respectively. The search for the text “PROC print” will be case-sensitive but the other two will not. Beware that including spaces before and after the delimiters in the FIND and REPLACE parameters will have an effect on the results. For example, specifying “|ODS rtf|” is not the same as specifying “|ODS rtf |”.

Note also that in the first search string and first replace string, the leading blanks will be ignored unless either the %STR or %NRSTR function or a delimiter is used to mask them.

For example, specifying...

```
,Find =PROC print
```

is the same as specifying...

```
,Find =      PROC print
```

but is not the same as the following two (equivalent) FIND specifications...

```
,Find = |      PROC print|
```

```
,Find = %str(      PROC print)
```

11. Deleting text:

```
%Find_Replace(
  Files    = C:\FR_Test\Programs\*.sas
  ,Find    = /* Temporary program marker */
  ,Replace = ||
);
```

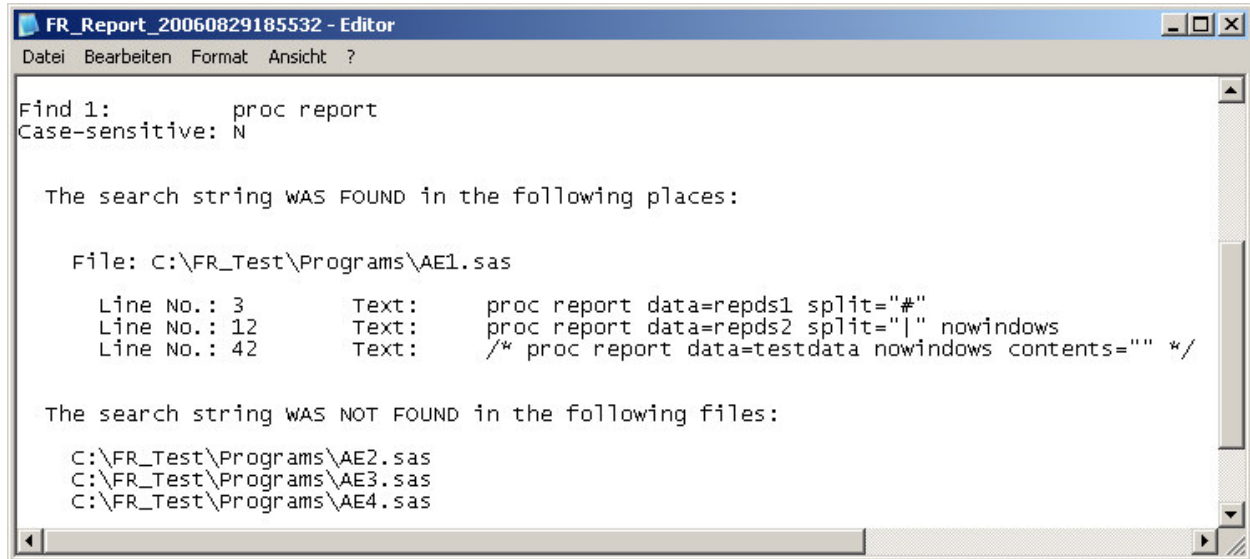
The macro call above will delete the text “/* Temporary program marker */” from all files with the **.sas** extension in the **C:\FR_Test\Programs** folder. This is achieved by specifying “||” (two delimiter characters). The same text could be blanked out (text over-written with spaces) with the following macro call:

```
%Find_Replace(
  Files    = C:\FR_Test\Programs\*.sas
  ,Find    = /* Temporary program marker */
  ,Replace = |                               |
);
```

REPORT OUTPUT

Each call to the %FIND_REPLACE macro produces a simple text report of the results. The text file report produces output in the format shown in Figure 1. The text file report is written to the folder specified by the OUTFOLDER macro parameter. This was mentioned earlier and has been given a default value of **C:\FR_Test\FR_Output**.

Figure 1



It may sometimes be of interest to find out which files do not contain certain text. This information is also presented in the report text file, as seen above in Figure 1.

OUTPUT DATA SETS

In addition to the text report, two SAS data sets are created in the WORK library of the SAS session – FR_REPA and FR_REPB. These can be used to query the results or, for example, to control some other process.

The FR_REPA data set contains the results of the %FIND_REPLACE macro call on a file line level. It will contain an observation for each file line where at least one of the find strings was found. Where none of the find strings are found in a file, one observation will appear in this data set with flags to indicate such. The structure of the FR_REPA data set is presented in Table 1.

Table 1: The FR_REPA Data Set

Variable(s)	Description
FILE	The file searched.
LINE	The line of the file where the search string was found. When text is replaced, the modified line appears in this variable.
ORIGLINE	The original line of text (this variable is not present when a find takes place without replace).
LINENO	The line number within the file.
FIND1-FIND n	These hold the n find strings.
REPL1-REPL n	These hold the n replace strings (only present when replace text strings have been specified).
CSEN1-CSEN n	These hold n 'Y/N' flag variables to indicate whether or not the search was case-sensitive.
FOUND1-FOUND n	Numeric flags, 0 or 1, to indicate whether or not the text strings were found (0 for not found in the file and 1 for found in the line given by the LINENO variable).

The FR_REPB data set is created using the information in FR_REPA and provides a synopsis on a file level. It will contain one observation per processed file. It has a flag for each find string to indicate whether or not the string was found. The structure of the FR_REPB data set is presented in Table 2.

Table 2: The FR_REPB Data Set

Variable(s)	Description
FILE	The file searched.
FIND1-FIND n	These hold the n find strings.
REPL1-REPL n	These hold the n replace strings.
CSEN1-CSEN n	These hold n 'Y/N' flag variables to indicate whether or not the search was case-sensitive.
FOUND1-FOUND n	Numeric flags, 0 or 1, to indicate whether or not the text strings were found in the file (0 for not found and 1 for found).

For both output data sets, their structure facilitate filtering using statements such as “WHERE FOUND1 AND FOUND2 AND NOT FOUND3” etc.

OTHER OUTPUT

Before text is replaced in a file, it is added to a zip file, which is created in the folder specified by the **OUTFOLDER** macro parameter (by default this is **C:\FR_Test\FR_Output** folder). In addition, during the execution of a **%FIND_REPLACE** macro call temporary files are created. They are written to the same folder and are deleted once they have served their purpose. These temporary files consist of various batch and text files.

It is worth noting that the **DEBUG** macro parameter can be used to negate the default behaviour of the macro which deletes all temporary files. An example call using this parameter is as follows:

```
%Find_Replace(
  Files    = C:\FR_Test\Programs\*.sas
  ,Find    = proc print
  ,Replace = proc report
  ,Debug   = Y
);
```

Setting the **DEBUG** parameter to “Y” instructs that all temporary output be kept. This will include various data sets in the **WORK** library of the SAS session as well as batch and text files.

All files written to the output folder have the same prefix, which you probably guessed by now is ‘FR_’. This includes temporary files, zip files as well the text report file. The SAS data sets created in the **WORK** library also receive the same prefix. This is controlled by the **OUTPREFIX** parameter which has this string as its default value. To store the results of a series of calls to the **%FIND_REPLACE** macro, this parameter can be used to provide some sort of control over the names of the output.

All output written to the output folder will have names which include a date/time identifier in the form **YYYYMMDDHHMMSS**, with **YYYY** denoting the year, **MM** the month, **DD** the day, **HH** the hour, **MM** the minute and **SS** the second. Each macro call creates one such identifier based on the date and time the macro was run. As such, all output relating to one macro call can be readily identified. (As an example, Figure 1 shows the file name of the text report file as “FR_Report_20060829185532”).

HOW THE MACRO WORKS

To gain an idea as to how the **%FIND_REPLACE** macro works, consider the following example call:

```
%Find_Replace(
  Files    = C:\FR_Test\Programs\*.sas
  ,Find    = proc print
  ,Replace = proc report
  ,SubDirs = Y
);
```

With the above macro call, the following process is initiated:

1. A list of files to be searched is created using a modified version of the **FILES** parameter specification in a **WZZIP** command along with its **-@** option. A command similar to the following is issued:

PhUSE 2006

```
wzip -rP -@"C:\FR_Test\FR_Output\FR_Files_YYYYMMDDHHMMSS.txt" egal.zip  
"C:\FR_Test\Programs\*.sas"
```

This command creates the **C:\FR_Test\FR_Output\FR_Files_YYYYMMDDHHMMSS.txt** text file which lists all files with the **.sas** extension located in the **C:\FR_Test\Programs** folder and its subfolders. The consideration of subfolders is due to the specification of `SubDirs = Y` which translates to the use of the **-rP** option in the above WZZIP command. Note that this command does not create a zip file – the **-@** option is used to create a listing of those files that would be zipped had this option not been specified.

2. The **C:\FR_Test\FR_Output\FR_Files_YYYYMMDDHHMMSS.txt** text file is read in and provides a means of looping through the relevant files. For each loop iteration an algorithm is applied to perform the find and replace operation, all changes occurring within a temporary SAS data set.
3. If the find string (in this case “proc print”) is found at least once and replaced within the data set (in this case by “proc report”) then a flag is set to indicate that the file should be re-written to include the changes relating to the replacement of the search string.
4. However, before the file is re-written, a backup is made. This is done by copying the original file into a zip file using a command similar to the following:

```
wzip -a -P "C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS.zip"  
"C:\FR_Test\Programs\AE1.sas"
```

Assuming the find string (“proc print”) was found in **C:\FR_Test\Programs\AE1.sas** and is to be replaced (by “proc report”), the command above adds this file to the **C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS.zip** zip file. Having the **-P** option included in the WZZIP command dictates that the file path information be retained. Note that the file path information includes the full path with the exception of the drive (in this case **C:**). This means that for the **C:\FR_Test\Programs\AE1.sas** file, the zip file will retain the path information **FR_Test\Programs\AE1.sas**.

5. Issuing a WZZIP command instructing that a file be zipped is not a guarantee that the file will be zipped. For this reason a WZUNZIP command is used to check that the file was successfully zipped.

```
wzunzip -@"C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS_ZIP_1_CHK.txt"  
"C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS.zip" "FR_Test\Programs\AE1.sas"
```

This command will produce the **C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS_ZIP_1_CHK.txt** text file which will list those files in the **C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS.zip** zip file having the **FR_Test\Programs\AE1.sas** file path information associated with it. Basically, if the file was successfully zipped then the **C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS_ZIP_1_CHK.txt** text file will contain one line of text reading “**FR_Test\Programs\AE1.sas**”. If for some reason the file is not zipped then this text file will be empty. Note that files are not unzipped with this command – the **-@** WZUNZIP option specifies that a listing be created containing those files that would be unzipped were this option not specified.

6. If the **C:\FR_Test\FR_Output\FR_YYYYMMDDHHMMSS_ZIP_1_CHK.txt** text file indicates that the file was successfully added to the zip file, the file is re-written.
7. As the macro loops through the various files, a record is kept within the **FR_REPA** data set (mentioned earlier) of the results of the find and replace operation. At the end of the macro execution, this data set provides the information necessary to determine where text was found and replaced, as well as the files in which the find string was not found at all.
8. The report text file is written to the output folder.

CONSIDERATIONS

The following characteristics of the %FIND_REPLACE macro should be noted:

- It is not possible to include tab characters in either the FIND or REPLACE parameters of the %FIND_REPLACE macro. This functionality was added to, and then removed from the macro on several occasions. Solving this problem always caused another. In the end it was removed.
- A text string consisting of space characters only cannot be searched for. This functionality could have been

PhUSE 2006

included by using an additional TRIM function for each and every search operation but was considered to be not worth the effort (processing effort rather than programming effort).

- When the %FIND_REPLACE macro reads in flat files, it uses the data step along with the INFILE and INPUT statements in the following way:

```
infile "Filename" linesize=32767 length=reclen end=eof;  
input line $varying32767 reclen;
```

There are two things to note. Firstly, the maximum line size that can be read in is of length 32767, which is the maximum available to the LINESIZE option of the INFILE statement. Files with lines of length greater than 32767 will be truncated when read in, and information will be lost if re-written. Secondly, with these statements trailing blanks are truncated. This means that if a file is re-written due to a find and replace action, then trailing blanks are lost. If the before and after size files differ considerably, this is probably the reason. If this behaviour is to be avoided, then so too is the %FIND_REPLACE macro.

- The WZZIP option, **-P**, is used to store file path information in the zip files created. However, the drive information is not (and cannot be) retained. The %FIND_REPLACE macro offers the possibility of searching on different drives. You might ask what happens when two files have the same file path and name, but reside on different drives. When files are searched only, this is of no concern and the search will take place as expected. However, when it is requested to replace a search text string with another, the macro will not carry this out and a corresponding message is issued to the SAS log. The problem is that the zip file cannot hold both files because without the drive, their file paths are the same. One file would have to overwrite the other on the zip file. Having two or more calls to the % FIND_REPLACE macro would circumvent this situation.
- The %FIND_REPLACE macro does not allow one to perform a Perl regular expression search. There is, however, a SAS macro available from the samples provided on the SAS website which is titled "A Windows-Based File Search Utility for Locating Text Using Perl Regular Expressions".

DISCLAIMER

The author accepts no responsibility for damage resulting from the use of the macros presented in this paper. They have not been peer-tested. Use at your own risk.

CONCLUSION

The %FIND_REPLACE macro can be used to perform a find or find and replace action over files within a Windows environment. It expects the availability of the WinZip Command Line Add-On commands WZZIP and WZUNZIP. Before a file is over-written via a replace action, the file is placed into a zip file, thus protecting against the loss or corruption of information. The macro is currently used by the author to help manage large suites of programs as well as being part of a number of batch processes. It has proved to be a useful programming tool.

CONTACT INFORMATION

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

David Brennan
Dungarvan
Ireland
david.n.brennan@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.