

Dear Readers,

Just two short weeks ago, I was producing the [Daily News](#) from SAS® Global Forum 2007. There was so much to see and do at the conference -- I can't begin to tell you about all the nifty tips and tricks I learned. I also overheard many attendees discussing the specific presentations that seemed tailored just for them. The theme of the conference was "innovation," and I truly believe we provided our customers with innovative ideas they could apply to their everyday work.

So in the spirit of the conference, this issue of the *SAS Tech Report* comes complete with samples and a technical white paper. I hope you'll make plans to join us at the next SAS® Global Forum in San Antonio. You never know what you might learn!

Thanks for your interest in SAS!



[Shelley Sessoms](#)
Editor, *SAS Tech Report*

Dynamically Determine the Creation Date and Last Modified Date for an External File

Use the DIR command and FILENAME PIPE to dynamically determine the creation and last modified dates of an external file on Windows. If running on Unix, use the LS command with the FILENAME PIPE.

```
/* *****
/
/* Samples 1 and 2 use the DIR command to grab information about an
*/
/* external file on Windows. You can modify this code to use other
*/
/* attributes as documented in the Windows HELP under the DIR
command.*/
/* Sample 3 uses the LS command to get the last modified date of an
*/
/* external file on Unix.
*/
/* *****
/

/* Sample 1: Return a file's creation date
*/

/* /t:c indicates you want a time field 'T', of type creation 'C'
*/
/* /a:-d return information for files only, not directories
*/

%let file=c:\tracks\dennis\x1.txt;
filename foo pipe "dir &file /t:c /a:-d ";
data _null_;
  infile foo firstobs=6;

  /* The ?? format modifier for error reporting suppresses printing the
  messages */
  /* and the input lines when SAS encounters invalid data values. The
  automatic */
  /* variable _ERROR_ is not set to 1 for the invalid observation.
  */

  /* The & format modifier enables you to read character values that
  contain */
  /* embedded blanks with list input and to specify a character
  informat. SAS */
  /* reads until it encounters multiple blanks.
  */
  input cr_date ?? :mddy8. cr_time ?? & time8.;
  if cr_date eq . then stop;
  put cr_date= worddate. / cr_time= timeampm.;
run;
```

```

/* Sample 2: Return a file's last modified date (Windows) */

/* /t:w indicates you want a time field 'T', of type Last Modified
'W' */
/* /a:-d return information for files only, not directories
*/
filename foo pipe "dir &file /t:w /a:-d";
data _null_;
  infile foo firstobs=6;
  input mod_date ?? : mmddyy8. mod_time ?? & time8.;
  if mod_date eq . then stop;
  put mod_date= worddate. / mod_time= timeampm.;
run;

/* Sample 3: Return a file's last modified date (Unix) */

/* ls is the Unix equivalent of DIR */
/* -g specifies not to print the file's owner */
/* -o specifies not to print the file's group */

/* Note that if the time of last modification is greater than */
/* six months, the year is substituted for the hour and minute */
/* of the modification. */

filename foo pipe "ls -g -o ~/test.txt";
data _null_;
  infile foo firstobs=2;
  input @24 mod_date $12. ;
  if mod_date=" " then stop;
  put mod_date= ;
run;

```

Create Variable Labels from Data Set Values

Dynamically create variable labels from data set values and apply them using PROC DATASETS. This technique can also be used when reading flat files that contain a record you want to use for variable labels.

```
/* Example 1: Use the values from one data set to create labels for
another */
/*           data set
*/

/* Create sample data sets. The values from WORK.ONE will be used as
*/
/* variable labels for WORK.TWO.
*/

data one;
  infile datalines dsd;
  input var1 :$11. var2 :$11. var3 :$11.;
datalines;
Label One,Label Two,Label Three
;

data two;
  input var1 :$11. var2 :$11. var3 :$11.;
datalines;
a b c
d e f
;

/* This option is used for debugging and can be removed afterwards. */
options mprint;

/* Create a macro to generate labels from values in the first data set
and apply */
/* the new labels to variables in the second data set.
*/

%macro labels(dsn1,dsn2);
  /* Open dataset whose values in the first observation will become new
labels */
  %let dsid=%sysfunc(open(&dsn1));

  /* CNT will contain the number of variables in &dsn1 */
  %let cnt=%sysfunc(attrn(&dsid,nvars));

  %do i = 1 %to &cnt;
    /* Create a different macro variable for each variable name in &dsn1
*/
    %let x&i=%sysfunc(varname(&dsid,&i));
  %end;

  /* Create macro variables for each value in the first observation in
&dsn1 */
  data _null_;
```

```

    set &dsn1;
    if _n_ = 1 then do;
        %do j = 1 %to &cnt;
            /* Note: If you are in SAS 9 or above, you can use CALL SYMPUTX
to remove */
            /* leading and trailing blanks.
*/
            call symput('var' || trim(left(&j)), &&x&j);
        %end;
    end;
run;

/* Use the macro variables to generate a LABEL statement in PROC
DATASETS. */
proc datasets library=work nolist;
    modify &dsn2;
    label
        %do i = 1 %to &cnt;
            &&x&i=&&var&i
        %end;
    ;
quit;
%mend labels;

/* Call the macro LABELS with two data sets. The first parameter
specifies the */
/* data set whose values will become the new labels. The second
parameter */
/* specifies the data set that will receive the new labels.
*/
%labels(one,two)

/* Check the results. */

proc print data=two label;
run;

/* Example 2: Reading a flat file whose first record is meant to be
variable */
/* names but are invalid SAS variable names. Create labels
from */
/* the first record instead.
*/

/* Create sample test file to be read. Modify your FILE statement
appropriately. */

data _null_;
    file "c:\temp\sample1727.txt";
    put "2005,2006,2007,Total Revenue";
    put "1,2,3,25000";
    put "4,5,6,50000";
run;

```

```

/* Read only the first record from SAMPLE1727.TXT using OBS=1 on the
INFILE statement. */

data one_2;
  infile "c:\temp\sample1727.txt" dsd obs=1;
  input var1 :$4. var2 :$4. var3 :$4. var4 :$13.;
run;

/* Read the rest of the flat file to create a second data set of data
values only */
/* using FIRSTOBS=2 on the INFILE statement.
*/

data two_2;
  infile "c:\temp\sample1727.txt" dsd firstobs=2;
  input var1 var2 var3 var4;
run;

/* Pass these two data sets as the parameters to the macro LABELS
illustrated */
/* in Example 1.
*/

%labels(one_2,two_2)

proc print data=two_2 label;
run;

```

Announcing SAS® Visual BI Software

SAS® Visual BI, powered by JMP®, marries rich, interactive visualization capabilities with SAS' powerful analytics and integrated business intelligence platform. Available as an add-on for SAS Enterprise BI Server, SAS Visual BI provides yet another way for business users to interact with information visually and dynamically.

Read the complete PDF: <http://www.sas.com/technologies/bi/visualization/visualbi/factsheet.pdf>

Optimization with SAS/OR®

This white paper describes the nature and purpose of optimization and considers the various types of optimization problems that can be solved with SAS/OR software. In addition, it explores the value that optimization adds to enterprise data, analytical technologies and business intelligence.

Download the full whitepaper:

http://www.sas.com/ctx/whitepapers/whitepapers_frame.jsp?code=270

SAS Education Announces the Opening of Two New Training Locations

SAS is extending its reach throughout the US with the opening of two new training facilities in Houston and Sacramento, CA. Some courses begin this month, so register early!

Read more: <http://support.sas.com/training/news/archives/tc.html>

Create a New Data Set for Each BY-Group in a Data Set

Create multiple SAS data sets from one SAS data set based upon the value of the BY variable.

```
/* Example 1 - Use macro logic to create a new data set for each BY-
Group in */
/*           an existing data set.
*/

/* Create sample data */

data test;
  input color $ num;
datalines;
blue 1
blue 2
blue 3
green 4
green 5
red 6
red 7
red 8
;

/* Create a new macro variable, VARn, for each BY-Group and a */
/* counter of the number of new macro variables created.      */

data _null_;
  set test end=eof;
  by color;
  /* On the first member of the BY-Group, create a new macro variable
VARn */
  /* and increment the counter FLAG.
*/
  if first.color then do;
    flag+1;
    call symput('var'||put(flag,8. -L),color);
  end;
  /* On the last observation of the data set, create a macro variable
to */
  /* contain the final value of FLAG.
*/
  if eof then call symput('tot',put(flag,8. -L));
run;

/* Create a macro to generate the new data sets. Dynamically produce
data set names */
/* on the DATA statement, using subsetting criteria to create the new
data sets      */
/* based upon the value of the BY variable.
*/

%macro groups(dsn,byvar);
  data %do i=1 %to &tot;
    &&var&i
  %end;;

```

```

    set &dsn;
    %do i=1 %to &tot;
        if &byvar="&&var&i" then output &&var&i;
    %end;
run;
%mend groups;

/* Call the macro GROUPS. Specify the name of the data set to be split
*/
/* in the first macro parameter and the name of the BY variable in the
*/
/* second parameter.
*/

%groups(test,color)

proc print data=blue;
    title 'Blue';
run;

proc print data=green;
    title 'Green';
run;

proc print data=red;
    title 'Red';
run;

/* Example 2 - Use CALL EXECUTE to pass a parameter to a macro in order
to */
/* create a new data set for each BY-Group in an existing
data */
/* set. The output is identical to the output created by
*/
/* Example 1 above.
*/

/* Compile the macro BREAK. The parameter BYVAL will be generated
below in */
/* the CALL EXECUTE.
*/

%macro break(byval);
    data &byval;
        set test(where=(color="&byval"));
    run;
%mend;

/* Use the same TEST data set created for Example 1. */

data _null_;
    set test;
    by color;
    if first.color then call execute('%break('||trim(color)||')');
run;

```

Webcasts and events

PharmaSUG 2007

June 3 - 6

Denver

Learn all about the great things planned for this year's premier event for SAS users in the pharmaceutical industry. Registration is now open!

F2007, SAS' Business Forecasting Conference

June 4 - 5

Cary, NC

Join more than 20 of the top forecasting experts in the world to learn the latest theories, trends and best practices in business forecasting.

Technical Event for SAS® Practitioners

June 29

London

This full-day event offers a wide range of opportunities to learn about the newest tools available for SAS practitioners who manage, report, analyze and display their organizations' data.