Macros

Is there an equivalent to the 'begins with' colon modifier to identify variables which 'end with' a particular string?

Efficient programmers (some call us lazy) love the shorthand notation which allows a variable list to be shortened by the colon modifier:

```
data out ;
  set in (keep = var:) ;
```

instead of:

```
data out ;
  set in (keep = var1 var3_1 var_99 variable) ;
```

So it would be great if we could simply change the modifier to get all of the variables in a dataset which end with the same text string. Sadly this isn't the case...

Efficient (lazy) programmers never let the absence of a feature get in the way of acheiving the result anyway! The following utility macro uses a PERL regular expression to retrieve matching records from a given dataset. The code is entirely written in MACRO language, so can be used inline, as shown in the example below.

```
%macro suffix( helpme
              ,dsn = &syslast
              ,sfx =
       ,sep = %str( )
      ) ;
  %local varlist lib dsid i var ;

  %*** Check for HELPME *** ;
  %if &helpme = ? or %substr(%upcase(&helpme),1,4) = HELP %then
  %do ;
    %put NOTE: There are three keyword parameters passed to this macro
: ;
 %put NOTE- dsn = [Specify a one- or a two-level dataset name from whi
ch variables are identified.  If not specified, the last-
created dataset will be used.] ;
    %put NOTE- sfx = [Specify the text string to identify variable nam
e suffix.  The parameter is required.  If not specified ALL variables
will be returned.] ;
 %put NOTE- sep = [Specify the separator to be inserted between the va
riable names in the list.  If not specified a SPACE will be used.  Val
id options are SPACE and COMMA.]
 %put NOTE- Sample Call: %nrstr(%%suffix%(sashelp.class,e,COMMA%)) to
return all variables in 'sashelp.class' which end with 'e' in a comma-
```

```
separated list. ;
    %goto endmac ;
  %end ;

  %if not %sysfunc(exist(&dsn)) %then
  %do ;
    %put ERROR: The dataset &dsn does not exist. The programme will no
t execute. ;
    %goto endmac ;
  %end ;

  %if &sep ne %str( ) %then
    %do ;
   %if %upcase(&sep) = COMMA %then %let sep = %str(, ) ;
      %else %let sep = %str( ) ;
 %end ;

  %*** Open the dataset *** ;
  %let dsid = %sysfunc(open(&dsn,i));
  %if &dsid = 0 %then
    %do ;
      %put %sysfunc(sysmsg()) ;
      %goto endmac ;
    %end ;

  %let numvars = %sysfunc(attrn(&dsid,nvars)) ;

  %*** Read all of the variables - keep matching instances, or all if
no suffix specified *** ;
  %if &sfx ne %then
    %do i = 1 %to &numvars ;
      %let var = %sysfunc(varname(&dsid,&i)) ;
      %if %sysfunc(prxmatch(/\w*%qupcase(&sfx)$/,%qupcase(&var))) %the
n %let varlist = &varlist&sep&var;
    %end;
  %else
    %do i = 1 %to &numvars ;
      %let varlist = &varlist&sep%sysfunc(varname(&dsid,&i));
    %end;

  %if %sysfunc(length(&varlist)) = 0 %then %put WARNING: No variables
in the dataset &dsn. end with the text string "&sfx". ;
  %*** Return resolved list into the Input Stack *** ;
  &varlist

  %endmac :
%mend suffix ;
```

Macros

An example of this code being used inline, within a statement, might look something like:

```
proc sql ;
  select %suffix(dsn = sashelp.class
                ,sfx = e
                ,sep = COMMA
                )
  from sashelp.class
  ;
quit ;
```

Unique solution ID: #1041
Author: Alan D Rudland
Last update: 2019-01-15 13:44