Macros

Can I create a macro variable longer than 65,534 bytes?

Well, no... The maximum length of a macro variable is restricted to the 65,534 characters, just like the maximum length of a character variable in a dataset is restricted to 32,767 bytes.

But given that a macro variable is just a string of text referenced by a symbolic value such as &mymacrotext it isn't that different from a macro program (which is just text) which can be retrieved by a macro program call such as %mymacrotext - remember that a macro program call doesn't require a semicolon (it's not a statement!) so in coding terms the only real difference between the two is the & or % token.

Macro program definitions are not restricted to the same length restrictions as the macro variable, and can be used in DATA step statements, which are also not bound by the length restrictions.

So how to overcome the issue? Say that you have an IN statement with a list of values and have sucessfully used a macro variable to build a list from an existing dataset:

```
proc sql noprint ;
  select model into :car_list separated by '", "'
  from sashelp.cars
  ;
quit ;
```

then using it in an IN statement:

```
...
where car_model in ("&car_list") ;
...
```

This is fine until the list becomes too long and the dreaded ERROR message appears in your LOG.

Instead of writing values into a macro variable - construct a macro program definition containing all of the values...

```
data _null_ ;
  file 'zipcodes.sas' ;
  set sashelp.zipcode end = lastobs ;
    if _n_ = 1 then put '%macro zipcodes ;' ;
    put zip z5. ;
    if lastobs then put '%mend zipcodes ;' ;
run ;

%include 'zipcodes.sas' ;
```

The external file now contains a macro program definition with the %macro... / %mend... statements before and after the long list.

```
1 %macro zipcodes ;
2  00501
3  00544
4  00601
5  00602
6  00603
```

...

```
41459  99921
41460  99922
41461  99923
41462  99925
41463  99926
41464  99927
41465  99928
41466  99929
41467  99950
41468  %mend zipcodes ;
```

The %include statement then retrieves the macro program and compiles the definition.

The macro program can now be called in essentially the same way as retrieving a macro variable:

```
...
where zip_code in (%zipcodes) ;
...
```

Again - please note that there is no semicolon on the macro program call - otherwise it would cause an early termination of the WHERE statement!

POSTSCRIPT:

For simplicity the code above writes the external file to the default directory, however unless this file is explicitly deleted after use, it will remain there. As an alternative, write the external file to the same location as the WORK library and it will be automatically deleted on exiting the session.

```
%let workpath = %sysfunc(pathname(work)) ;
...
file "&workpath\zipcodes.sas" ;
...
%include "&workpath\zipcodes.sas" ;
...
```

```
NOTE: 41468 records were written to the file "C:\Temp\SAS Temporary
      Files\_TD6080_SL066604_\zipcodes.sas".
```

Unique solution ID: #1065
Author: Alan D Rudland
Last update: 2022-04-06 12:11