A Macro to Word Wrap Long Text Strings into a SAS® Array

H. Ian Whitlock, Westat Inc.

Introduction

When reading data and associated comment fields, it may be necessary to handle character strings longer than 200 bytes. It is a simple task to store the string in 200 byte chunks, but it is harder to break up the chunks into an array of print sized lines which are space filled at the end so that no word is split between two lines.

A new SAS Institute book [1] on report writing gives such a macro, but it is three pages long, has 5 calls to a one page embedded macro, and uses GOTO's. Hence the algorithm is obscured, and too much SAS code is generated.

This paper presents a one page macro using only SAS code, macro variable references and macro invocation to develop a simple direct algorithm illustrating how to use pointers to an arbitrarily long substring stored as part of a character array. Two one line "function" macros, %PX (PTR) and %PP (PTR) are used in a key role to convert an absolute string pointer to the index of the corresponding array element and a relative pointer to the position within that element.

Test and Illustrated Use

```
/* read in 1 long string, apply %wordwrap, then
   write resulting print ready array */
data _null_ ;
   length s1 - s5 $ 40 ;
   input ( s1-s5 ) ($char40. / ) ;
   array in (5) s1 - s5 ;
   array out (15) $ 20 _temporary_ ;

   do i = 1 to dim ( in ) ; /* show input */
      put in ( i ) = ;
   end ;

   %wordwrap
      ( p=in , q=out , plen=40 , qlen=20 )

   do i = 1 to dim ( out ) ; /* show output */
      put out ( i ) = ;
   end ;
cards ;
aaaaaaaaaa10 aaaaaaaaaaa11 aaaaaaaa8 aaa
aaa6 bbbbbbbbbb10 bbbbbbbb8 bbbbbbbbbb10
 ccccccccccc11 cccccccccccc12 cccccccc8
ddddddddddd11      dddddddd8      dddddddddd
10 eeeeee6 eeeeee6 eeeeeeee8 eeeeeeee8 e
run ;
```

The Macro

```
%macro wordwrap (
      p= ,          /* input array name        */
      plen=200,     /* len vars in input array  */
      q= ,          /* output array name       */
      qlen=         /* len vars in output array*/
   ) ;

   %* -----------------------------------------------
   Purpose:
   Move text stored in an array from a long
   character string (usually over 200 bytes) where
   array elements need not mark word
   boundaries to an array with space filling at the
   end so that no word crosses an array element.

   Usage Notes:
   Parameters P, Q, and QLEN are required. If a
   word in &p is longer than &qlen, abort. Drop
   variables beginning with a double underscore.

   Working variables:
      __px, __qx      - index to resp arrays
      __rpb , __rqb   - rel beg ptrs within elemnt
      __rpe           - rel end ptr within element
      __pb , __pe     - abs (full string) begin and
                        end ptrs to substring of &p
      __pc            - one byte char val from &p
      __sublen        - length of substring

   Basic algorithm:
      initialize __pe to 0
      loop over index of &q until finished
         set __pb to __pe + 1
         set __pe to maximum that will fit in &q
```

```
          when needed back up to a space
          move substring to &q (using two or
          more moves when the substring begins
          and ends on different elements)
     end loop
------------------------------------------------------------ *;

  drop __: ;
  __pe = 0 ;
  do  __qx  =  lbound(&q)  to  hbound(&q)
           until ( __pe >= &plen * dim(&p) ) ;
     __pb = __pe + 1 ;
     if &p ( %px(__pb) ) = " " then
     do ;
        __qx = __qx - 1 ;
        leave ;
     end ;
     else
     do ;
        __pe   =   min   (&plen*dim(&p),
                 __pb+&qlen-1) ;
        /* skip backup when __pe is at end */
        if __pe = &plen * dim ( &p ) then ;
        else
        if substr(&p(%px(__pe+1)),
              %pp(__pe+1), 1 ) ^= " " then
        do ;      /* back up to first blank */
           do __pe = __pe to __pb by - 1
              until (__pc=" " or __pe < __bp) ;
              __pc = substr ( &p(%px(__pe)),
                    %pp(__pe) , 1 ) ;
           end ;
           if __pe < __pb then
           do ;   /* token too long */
              __px = %px ( __pb ) ;
              put "WORDRAP: token too long"
                 " - will abort" /
                 "&p(" __px +(-1) ')='
                 &p(__px) $char&plen.. ;
              __px = __px + 1 ;
              put "&p(" __px +(-1) ')='
                 &p(__px) $char&plen.. ;
              abort 99 ;
           end ;
        end ;
        /* move substring to q array */
        __rqb = 1 ;
        do while ( %px(__pb) ^= %px(__pe) ) ;
           __rpb = %pp(__pb) ;
           __sublen = &plen - __rpb + 1 ;
           substr ( &q ( __qx ) , __rqb ) =
              substr(   &p(%px(__pb)),
                        __rpb,
                        __sublen) ;
           __rqb = __sublen + 1 ;
           __pb = ( __px ) * &plen + 1 ;
        end ;
        /* move part on the end element */
        substr ( &q ( __qx ) , __rqb ) =
           substr ( &p(%px(__pe)) ,
              %pp(__pb) , __pe - __pb + 1 ) ;
     end ;
  end ;

  do __qx = __qx + 1 to dim ( &q ) ;
     &q ( __qx ) = '' ;
  end ;
%mend  wordwrap ;

%macro pp ( ptr ) ;  /* rel ptr to p array substr */
   mod ( ( &ptr - 1 ) , &plen ) + 1
%mend  pp ;

%macro px ( ptr ) ; /* p array index of pointer */
   int ( ( &ptr - 1 ) / &plen ) + 1
%mend  px ;
```

The author can be contacted by mail or e-mail

H. Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850-3129

Whitloi1@westat.com

[1] John E. Hewlett, "Using a Word-Wrap Macro on Long Text Variables", pp 151-164 in "Reporting from the Field: SAS Software Experts Present Real-world Report-Writing Applications", 1994.