



GOING TO GREAT LENGTHS: USING VARCHARS IN THE DATA STEP

Rick Langston
Retired; Formerly of SAS Institute Inc.

DATA Step Truism



- 2 types of variables
- Floating point numeric (length 3-8)
- Fixed-length character (length 1-200 until Version 7)
- Fixed-length character (length 1-32767)
- LENGTH statement can be used
- Or a “defining instance” defines the length
- But as of 9.4m5, there’s more!

VARCHARs!



- You must use a LENGTH statement:
length a b c x1-x10 varchar(*);
length y1 z2 varchar(10);
- You can specify either * or a fixed value
- I personally always use *
- Several examples to show how VARCHARs work...



Concatenation



```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a  xyz l=6  
z=axyz l=4
```



```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```



```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```



```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```



```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```

```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a  xyz l=6  
z=axyz l=4
```

```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```

```
data _null_;  
  length x $3 y varchar(*) z $10;  
  x = 'a'; y = 'a';  
  z = x || 'xyz';  
  l = length(z);  
  put z= l=;  
  z = y || 'xyz';  
  l = length(z);  
  put z= l=;  
run;
```

produces

```
z=a xyz l=6  
z=axyz l=4
```



Silent Truncation!



```
data _null_;  
  length x y $3 z varchar(*);  
  x = 'a';  
  y = 'a';  
  z = 'a';  
  x = x || x;  
  y = trim(y) || y;  
  z = z || z;  
  put x= y= z=;  
run;
```

produces

x=a y=aa z=aa



```
data _null_;  
  length x y $3 z varchar(*);  
  x = 'a';  
  y = 'a';  
  z = 'a';  
  x = x || x;  
  y = trim(y) || y;  
  z = z || z;  
  put x= y= z=;  
run;
```

produces

x=a y=aa z=aa

```
data _null_;  
  length x y $3 z varchar(*);  
  x = 'a';  
  y = 'a';  
  z = 'a';  
  x = x || x;  
  y = trim(y) || y;  
  z = z || z;  
  put x= y= z=;  
run;
```

produces

x=a y=aa z=aa

```
data _null_;  
  length x y $3 z varchar(*);  
  x = 'a';  
  y = 'a';  
  z = 'a';  
  x = x || x;  
  y = trim(y) || y;  
  z = z || z;  
  put x= y= z=;  
run;
```

produces

```
x=aa y=aa z=aa
```

a _ _ a _ _ -> a _ _

```
data _null_;  
  length x y $3 z varchar(*);  
  x = 'a';  
  y = 'a';  
  z = 'a';  
  x = x || x;  
  y = trim(y) || y;  
  z = z || z;  
  put x= y= z=;  
run;
```

produces

```
x=a y=aa z=aa
```



```
data _null_;  
  length x $2 y $3;  
  y = 'abc';  
  x = y;  
  put x= y=;  
run;
```

produces

x=ab y=abc



```
data _null_;  
  length x $2 y $3;  
  y = 'abc';  
  x = y;  
  put x= y=;  
run;
```

produces

```
x=ab y=abc
```

Functions supporting VARCHARs



- INDEX, INDEXC, SCAN/CALL SCAN, SUBSTR, REPEAT, LENGTH
- TRANSLATE, TRANWRD, COMPRESS, VERIFY, COMPBL, FIND
- LEFT, RIGHT, STRIP, UPCASE, LOWCASE
- MD5, SHA256, HASHING*,
- SYMGET, SYMPUT ,SYMPUTX
- CAT*, ANY*
- KINDEX, KLENGTH, etc.

Argument passing

- Pass a VARCHAR:

```
x = length(y);
```

- Pass a VARCHAR to get back a VARCHAR:

```
z = repeat(y,99999);
```

```
z = substr(z,60000);
```





```
data _null_;  
  length x varchar(*);  
  x = repeat('a',49999); /* using constant */  
  l = length(x);  
  put l=;  
  x = 'a';  
  x = repeat(x,49999); /* using varchar */  
  l = length(x);  
  put l=;  
run;
```

produces

l=32767

l=50000

```
data _null_;  
  length x varchar(*);  
  x = repeat('a',49999); /* using constant */  
  l = length(x);  
  put l=;  
  x = 'a';  
  x = repeat(x,49999); /* using varchar */  
  l = length(x);  
  put l=;  
run;
```

produces

l=32767

l=50000

```
data _null_;  
  length x varchar(*);  
  x = repeat('a',49999); /* using constant */  
  l = length(x);  
  put l=;  
  x = 'a';  
  x = repeat(x,49999); /* using varchar */  
  l = length(x);  
  put l=;  
run;
```

produces

l=32767

l=50000



```
data _null_;  
  length x varchar(*);  
  x = repeat('a',49999); /* using constant */  
  l = length(x);  
  put l=;  
  x = 'a';  
  x = repeat(x,49999); /* using varchar */  
  l = length(x);  
  put l=;  
run;
```

produces

```
l=32767
```

```
l=50000
```

```
data _null_;  
  length x varchar(*);  
  x = repeat('a',49999); /* using constant */  
  l = length(x);  
  put l=;  
  x = 'a';  
  x = repeat(x,49999); /* using varchar */  
  l = length(x);  
  put l=;  
run;
```

produces

l=32767

l=50000



VARCHARS and the Macro Facility

VARCHARs and Macro Facility



- Macro variables can be up to 65,534 in length
- SYMGET and SYMPUT/SYMPUTX can handle that



```
data _null_;  
  length x $20000;  
  x = repeat('a',19999);  
  call symput('x20000',x);  
  run;  
  
%let x40000=&x20000.&x20000.;  
%put length of x40000:%length(&x40000);
```

produces

```
length of x40000:40000
```

```
data _null_;  
  length x $20000;  
  x = repeat('a',19999);  
  call symput('x20000',x);  
run;
```

```
%let x40000=&x20000.&x20000.;  
%put length of x40000:%length(&x40000);
```

produces

```
length of x40000:40000
```



```
data _null_;  
  length x $32767 y varchar(*);  
  x = symget('x40000');  
  y = 'x40000';  
  y = symget(y);  
  l = length(x); put l=;  
  l = length(y); put l=;  
run;
```

produces

l=32767

l=40000

```
data _null_;  
  length x $32767 y varchar(*);  
  x = symget('x40000');  
  y = 'x40000';  
  y = symget(y);  
  l = length(x); put l=;  
  l = length(y); put l=;  
run;
```

produces

```
l=32767  
l=40000
```

```
data _null_;  
  length x $32767 y varchar(*);  
  x = symget('x40000');  
  y = 'x40000';  
  y = symget(y);  
  l = length(x); put l=;  
  l = length(y); put l=;  
run;
```

produces

```
l=32767
```

```
l=40000
```



```
data _null_;
  length x varchar(*);
  x = 'z';
  * x = repeat('z',65532); /* will return 32767 chars */
  x = repeat(x,65532); /* works properly if arg1 is a varchar */
  x = x || 'x'; /* would be a newbie error if not a varchar */
  y = length(x);
  put y=;
  y = index(x,'x');
  put y=;
  call symputx('mytest',x); /* second argument is a varchar */
run;

%put %length(&mytest);
%put %index(&mytest,x);
```

produces

```
y=65534
y=65534
65534
65534
```



```
data _null_;
  length x varchar(*);
  x = 'z';
  * x = repeat('z',65532); /* will return 32767 chars */
  x = repeat(x,65532); /* works properly if arg1 is a varchar */
  x = x || 'x'; /* would be a newbie error if not a varchar */
  y = length(x);
  put y=;
  y = index(x,'x');
  put y=;
  call symputx('mytest',x); /* second argument is a varchar */
run;
%put %length(&mytest);
%put %index(&mytest,x);
```

produces

```
y=65534
```

```
y=65534
```

```
65534
```

```
65534
```



```
data _null_;  
  length x varchar(*);  
  x = 'z';  
  * x = repeat('z',65532); /* will return 32767 chars */  
  x = repeat(x,65532); /* works properly if arg1 is a varchar */  
  x = x || 'x'; /* would be a newbie error if not a varchar */  
  y = length(x);  
  put y=;  
  y = index(x,'x');  
  put y=;  
  call symputx('mytest',x); /* second argument is a varchar */  
run;  
  
%put %length(&mytest);  
%put %index(&mytest,x);
```

produces

```
y=65534
```

```
y=65534
```

```
65534
```

```
65534
```



```
data _null_;  
  length x varchar(*);  
  x = 'z';  
  * x = repeat('z',65532); /* will return 32767 chars */  
  x = repeat(x,65532); /* works properly if arg1 is a varchar */  
  x = x || 'x'; /* would be a newbie error if not a varchar */  
  y = length(x);  
  put y=;  
  y = index(x,'x');  
  put y=;  
  call symputx('mytest',x); /* second argument is a varchar */  
run;
```

```
%put %length(&mytest);  
%put %index(&mytest,x);
```

produces

```
y=65534
```

```
y=65534
```

```
65534
```

```
65534
```



VARCHARS can be length 0!



```
data _null_;  
  x = '';  
  ...
```



```
data _null_;  
  length x varchar(*);  
  x = ' ';  
  y = 'a' || x || 'b';  
  put y=;  
  x = '  ';  
  x = trimn(x);  
  y = 'a' || x || 'b';  
  put y=;  
run;
```

produces

y=ab

y=ab

```
data _null_;  
  length x varchar(*);  
  x = ' ';  
  y = 'a' || x || 'b';  
  put y=;  
  x = ' ';  
  x = trimn(x);  
  y = 'a' || x || 'b';  
  put y=;  
run;
```

produces

y=ab

y=ab

```
data _null_;  
  length x varchar(*);  
  x = ' ';  
  y = 'a' || x || 'b';  
  put y=;  
  x = '  ';  
  x = trimn(x);  
  y = 'a' || x || 'b';  
  put y=;  
run;
```

produces

y=ab

y=ab

```
data _null_;  
  length x varchar(*);  
  x = ' ';  
  y = 'a' || x || 'b';  
  put y=;  
  x = '  ';  
  x = trimn(x);  
  y = 'a' || x || 'b';  
  put y=;  
run;
```

produces

```
y=ab  
y=ab
```



Trailing blanks don't
matter
for VARCHARS!

```
data _null_;
  length x $8;
  length y varchar(*);
  x='abc';
  if x='abc' then put "x matches with 'abc'...";
  x='abc  ';
  if x='abc' then put "x matches with 'abc'...";
  y='abc';
  l=lengthc(y);
  if y='abc' then put "y matches with 'abc'..." l=;
  y='abc  ';
  l=lengthc(y);
  if y='abc' then put "y matches with 'abc'..." l=;
run;
```

produces

```
x matches with 'abc'...
x matches with 'abc'...
y matches with 'abc'... l=3
y matches with 'abc'... l=6
```



```

data _null_;
  length x $8;
  length y varchar(*);
  x='abc';
  if x='abc' then put "x matches with 'abc'...";
  x='abc  ';
  if x='abc' then put "x matches with 'abc'...";
  y='abc';
  l=lengthc(y);
  if y='abc' then put "y matches with 'abc'..." l=;
  y='abc  ';
  l=lengthc(y);
  if y='abc' then put "y matches with 'abc'..." l=;
run;

```

produces

```

x matches with 'abc'...
x matches with 'abc'...
y matches with 'abc'... l=3
y matches with 'abc'... l=6

```



```
data _null_;  
  length x $8;  
  length y varchar(*);  
  x='abc';  
  if x='abc' then put "x matches with 'abc'...";  
  x='abc  ';  
  if x='abc' then put "x matches with 'abc'...";  
  y='abc';  
  l=lengthc(y);  
  if y='abc' then put "y matches with 'abc'..." l=;  
  y='abc  ';  
  l=lengthc(y);  
  if y='abc' then put "y matches with 'abc'..." l=;  
run;
```

produces

```
x matches with 'abc'...  
x matches with 'abc'...  
y matches with 'abc'... l=3  
y matches with 'abc'... l=6
```

```
data _null_;  
  length x $8;  
  length y varchar(*);  
  x='abc';  
  if x='abc' then put "x matches with 'abc'...";  
  x='abc  ';  
  if x='abc' then put "x matches with 'abc'...";  
  y='abc';  
  l=lengthc(y);  
  if y='abc' then put "y matches with 'abc'..." l=;  
  y='abc  ';  
  l=lengthc(y);  
  if y='abc' then put "y matches with 'abc'..." l=;  
run;
```

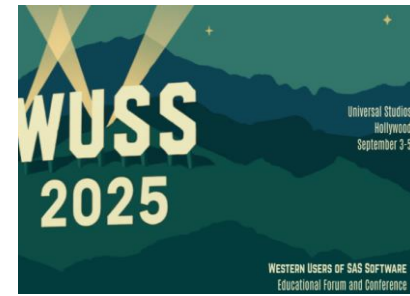
produces

```
x matches with 'abc'...  
x matches with 'abc'...  
y matches with 'abc'... l=3  
y matches with 'abc'... l=6
```



VARCHARs cannot be
saved in data sets, except

...



```
4          libname mycas cas;
NOTE: Libref MYCAS was successfully assigned as follows:
      Engine:          CAS
```

```
5          data mycas.mytable;
6              length abc varchar (*);
7              length x $5;
8              x = 'a';
9              abc = 'b';
10             run;
```

NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step has no input data set and will run in a single thread.

NOTE: The table mytable in caslib τ has 1 observations and 2 variables.

```
11          proc contents data=mycas.mytable; run;
            data _null_; set mycas.mytable;
12              put x= abc=;
13              abc = abc || abc;
14              put abc=; /* will show abc=bb if abc is a varchar */
15             run;
```

NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step will run in multiple threads.

x=a abc=b

abc=bb

4

```
libname mycas cas;
```

NOTE: Libref MYCAS was successfully assigned as follows:

```
Engine:          CAS
```

5

```
data mycas.mytable;
```

6

```
    length abc varchar (*);
```

7

```
    length x $5;
```

8

```
    x = 'a';
```

9

```
    abc = 'b';
```

10

```
run;
```

NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step has no input data set and will run in a single thread.

NOTE: The table mytable in caslib τ has 1 observations and 2 variables.

11

```
proc contents data=mycas.mytable; run;
```

```
data _null_; set mycas.mytable;
```

12

```
    put x= abc=;
```

13

```
    abc = abc || abc;
```

14

```
    put abc=; /* will show abc=bb if abc is a varchar */
```

15

```
run;
```

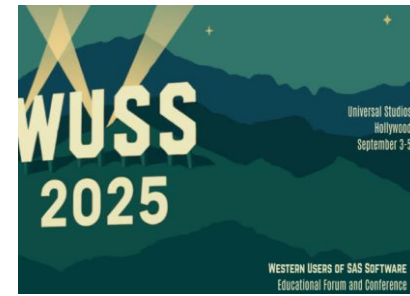
NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step will run in multiple threads.

```
x=a abc=b
```

```
abc=bb
```





```
4          libname mycas cas;  
NOTE: Libref MYCAS was successfully assigned as follows:  
      Engine:          CAS
```

```
5          data mycas.mytable;  
6              length abc varchar (*);  
7              length x $5;  
8              x = 'a';  
9              abc = 'b';  
10             run;
```

NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step has no input data set and will run in a single thread.

NOTE: The table mytable in caslib τ has 1 observations and 2 variables.

```
11         proc contents data=mycas.mytable; run;  
          data _null_; set mycas.mytable;  
12             put x= abc=;  
13             abc = abc || abc;  
14             put abc=; /* will show abc=bb if abc is a varchar */  
15             run;
```

NOTE: Running DATA step in Cloud Analytic Services.

NOTE: The DATA step will run in multiple threads.

x=a abc=b

abc=bb



Alphabetic List of Variables and Attributes

#	Variable	Type	Bytes	Len Chars	Max Bytes Used
1	abc	Varchar	.	.	1
2	x	Char	5		



```
1      options msglevel=i;
2      data temp;
3          length x varchar(*);
4          length y varchar(32);
5          x = 'abc';
6          y = 'xyz';
7      run;
```

NOTE: VARCHAR data type is not supported by the V9 engine. Variable x has been converted to CHAR data type.

NOTE: VARCHAR data type is not supported by the V9 engine. Variable y has been converted to CHAR data type.

NOTE: The data set WORK.TEMP has 1 observations and 2 variables.

```
1      options msglevel=i;  
2      data temp;  
3          length x varchar(*);  
4          length y varchar(32);  
5          x = 'abc';  
6          y = 'xyz';  
7          run;
```

NOTE: VARCHAR data type is not supported by the V9 engine. Variable x has been converted to CHAR data type.

NOTE: VARCHAR data type is not supported by the V9 engine. Variable y has been converted to CHAR data type.

NOTE: The data set WORK.TEMP has 1 observations and 2 variables.



Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	x	Char	32767
2	y	Char	32



No mixing in an array!



```
14      data _null_;  
15          length x y z varchar(*);  
16          length x2 $32767;  
17          array myvars x y z x2;
```

ERROR: All variables in array list must be the same type, i.e., all numeric or character.



No VARCHARs in INPUT!



```
10      filename mytest temp;
11      data _null_; file mytest recfm=f lrecl=100000;
12          do i=1 to 100000;
13              put 'x' @;
14          end;
15      put;
16      run;
```

NOTE: 1 record was written to the file MYTEST.

```
17
18      data _null_; infile mytest recfm=f lrecl=100000;
19          length x varchar(*);
20          input x;
21          l=length(x);
22          put l=;
23      run;
```

```
l=32767
```

NOTE: 1 record was read from the infile MYTEST.



```
24
25     data _null_; infile mytest recfm=f lrecl=100000 length=l;
26         length x varchar(*) y $32767;
27         input @;
28         do i=1 to l by 32767;
29             part = min(32767,l-i+1);
30             input y $varying32767. part @;
31             x = x || y;
32         end;
33         l=length(x);
34         put l=;
35         run;
```

l=100000

```
24  
25 data _null_; infile mytest recfm=f lrecl=100000 length=1;  
26 length x varchar(*) y $32767;  
27 input @;  
28 do i=1 to l by 32767;  
29     part = min(32767,l-i+1);  
30     input y $varying32767. part @;  
31     x = x || y;  
32     end;  
33     l=length(x);  
34     put l=;  
35     run;
```

l=100000

```
x = x || substr(y,1,part);
```



Example with felis functionus (the CAT family of functions)

```

%macro cat_function(function);
data _null_;
  length x varchar(*);
  length y z $10;
  x = 'x';
  x = ' ' || repeat(x,99999) || 'y';
  y = ' abc';
  z = ' def';
  x = &function(y,z,x);
  l = length(x);
  put l=;
  put 'x:' x $char50.;
  l = index(x,'y');
  put l=;
run;
%mend;

%cat_function(cat);
%cat_function(cats);

```



```
%macro cat_function(function);
```

```
data _null_;  
  length x varchar(*);  
  length y z $10;  
  x = 'x';  
  x = ' ' || repeat(x,99999) || 'y';  
  y = ' abc';  
  z = ' def';  
  x = &function(y,z,x);  
  l = length(x);  
  put l=;  
  put 'x:' x $char50.;  
  l = index(x,'y');  
  put l=;  
run;  
%mend;
```

```
%cat_function(cat);  
%cat_function(cats);
```



```
%macro cat_function(function);
```

```
data _null_;
```

```
length x varchar(*);
```

```
length y z $10;
```

```
x = 'x';
```

```
x = ' ' || repeat(x,99999) || 'y';
```

```
y = ' abc';
```

```
z = ' def';
```

```
x = &function(y,z,x);
```

```
l = length(x);
```

```
put l=;
```

```
put 'x:' x $char50.;
```

```
l = index(x,'y');
```

```
put l=;
```

```
run;
```

```
%mend;
```

```
%cat_function(cat);
```

```
%cat_function(cats);
```



```

%macro cat_function(function);
data _null_;
  length x varchar(*);
  length y z $10;
  x = 'x';
  x = ' ' || repeat(x,99999) || 'y';
  y = ' abc';
  z = ' def';
  x = &function(y,z,x);
  l = length(x);
  put l=;
  put 'x:' x $char50.;
  l = index(x,'y');
  put l=;
run;
%mend;

%cat_function(cat);
%cat_function(cats);

```



```

%macro cat_function(function);
data _null_;
  length x varchar(*);
  length y z $10;
  x = 'x';
  x = ' ' || repeat(x,99999) || 'y';
  y = ' abc';
  z = ' def';
  x = &function(y,z,x);
  l = length(x);
  put l=;
  put 'x:' x $char50.;
  l = index(x,'y');
  put l=;
run;
%mend;

%cat_function(cat);
%cat_function(cats);

```





```

%macro cat_function(function);
data _null_;
  length x varchar(*);
  length y z $10;
  x = 'x';
  x = ' ' || repeat(x,99999) || 'y';
  y = ' abc';
  z = ' def';
  x = &function(y,z,x);
  l = length(x);
  put l=;
  put 'x:' x $char50.;
  l = index(x,'y');
  put l=;
run;
%mend;

```

produces

```

%cat_function(cat);          l=100024
%cat_function(cats);        x: abc      def      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                             l=100024

                             l=100007
                             x:abcdefxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                             l=100007

```



Hashing example



```
1 data _null_; file 'x100000.txt' recfm=f lrecl=1;  
2 do i=1 to 100000; put 'x'; end;  
3 run;
```

```
NOTE: 100000 records were written to the file 'x100000.txt'.  
NOTE: DATA statement used (Total process time):  
real time          0.35 seconds  
cpu time           0.03 seconds
```

Hash'em all!

Free online text & file hashing

[Wondering why you may want to hash something? >>](#)
New: [Hash'em all! available as Firefox search plugin](#)

Write or paste your text here:

The quick brown fox jumps over the lazy dog

Hash this text >

Select a file (new: max 10mb!):

Browse... x100000.txt

Hash this file >

Upload of large files may take a while.

Algorithm:

- | | | | | |
|---|---------------------------------|--|--|--|
| <input type="radio"/> SHA 160bit (SHA1) | <input type="radio"/> WHIRLPOOL | <input type="radio"/> HAVAL 128bit, 3 r. | <input type="radio"/> HAVAL 224bit, 4 r. | <input type="radio"/> Tiger 128bit, 3 p. |
| <input type="radio"/> SHA 256bit | <input type="radio"/> Snefru | <input type="radio"/> HAVAL 160bit, 3 r. | <input type="radio"/> HAVAL 256bit, 4 r. | <input type="radio"/> Tiger 160bit, 3 p. |
| <input type="radio"/> SHA 384bit | <input type="radio"/> GOST | <input type="radio"/> HAVAL 192bit, 3 r. | <input type="radio"/> HAVAL 128bit, 5 r. | <input type="radio"/> Tiger 192bit, 3 p. |
| <input type="radio"/> SHA 512bit | <input type="radio"/> Adler32 | <input type="radio"/> HAVAL 224bit, 3 r. | <input type="radio"/> HAVAL 160bit, 5 r. | <input type="radio"/> Tiger 128bit, 4 p. |
| <input type="radio"/> MD4 | <input type="radio"/> CRC32 | <input type="radio"/> HAVAL 256bit, 3 r. | <input type="radio"/> HAVAL 192bit, 5 r. | <input type="radio"/> Tiger 160bit, 4 p. |
| <input checked="" type="radio"/> MD5 | <input type="radio"/> CRC32b | <input type="radio"/> HAVAL 128bit, 4 r. | <input type="radio"/> HAVAL 224bit, 5 r. | <input type="radio"/> Tiger 192bit, 4 p. |
| <input type="radio"/> RIPEMD 128bit | | <input type="radio"/> HAVAL 160bit, 4 r. | <input type="radio"/> HAVAL 256bit, 5 r. | |
| <input type="radio"/> RIPEMD 160bit | | <input type="radio"/> HAVAL 192bit, 4 r. | | |

* p = passes, r = rounds



Hash'em all!

Free online text & file hashing

Wondering why you may want to hash something? >>
New: [Hash'em all! available as Firefox search plugin](#)

Write or paste your text here:

The quick brown fox jumps over the lazy dog

Hash this text >

Select a file (new: max 10mb!):

Browse... x100000.txt

Hash this file >

Upload of large files may take a while.

Algorithm:

- | | | | | |
|---|---------------------------------|--|--|--|
| <input type="radio"/> SHA 160bit (SHA1) | <input type="radio"/> WHIRLPOOL | <input type="radio"/> HAVAL 128bit, 3 r. | <input type="radio"/> HAVAL 224bit, 4 r. | <input type="radio"/> Tiger 128bit, 3 p. |
| <input type="radio"/> SHA 256bit | <input type="radio"/> Snefru | <input type="radio"/> HAVAL 160bit, 3 r. | <input type="radio"/> HAVAL 256bit, 4 r. | <input type="radio"/> Tiger 160bit, 3 p. |
| <input type="radio"/> SHA 384bit | <input type="radio"/> GOST | <input type="radio"/> HAVAL 192bit, 3 r. | <input type="radio"/> HAVAL 128bit, 5 r. | <input type="radio"/> Tiger 192bit, 3 p. |
| <input type="radio"/> SHA 512bit | <input type="radio"/> Adler32 | <input type="radio"/> HAVAL 224bit, 3 r. | <input type="radio"/> HAVAL 160bit, 5 r. | <input type="radio"/> Tiger 128bit, 4 p. |
| <input type="radio"/> MD4 | <input type="radio"/> CRC32 | <input type="radio"/> HAVAL 256bit, 3 r. | <input type="radio"/> HAVAL 192bit, 5 r. | <input type="radio"/> Tiger 160bit, 4 p. |
| <input checked="" type="radio"/> MD5 | <input type="radio"/> CRC32b | <input type="radio"/> HAVAL 128bit, 4 r. | <input type="radio"/> HAVAL 224bit, 5 r. | <input type="radio"/> Tiger 192bit, 4 p. |
| <input type="radio"/> RIPEMD 128bit | | <input type="radio"/> HAVAL 160bit, 4 r. | <input type="radio"/> HAVAL 256bit, 5 r. | |
| <input type="radio"/> RIPEMD 160bit | | <input type="radio"/> HAVAL 192bit, 4 r. | | |

* p = passes, r = rounds



File: x100000.txt (97.66 Kb)
Hash (md5) of selected file (0.004 seconds):
D5816F35916D1D9482F80F1EC201101D



```
data _null_;  
  length x varchar(*);  
  x = 'x';  
  x = repeat(x,99999);  
  md5 = hashing('md5',x);  
  put md5=;  
  md5x = md5(x);  
  put md5x=$hex32.;  
  sha256 = hashing('sha256',x);  
  put sha256=;  
  sha256x = sha256(x);  
  put sha256x=$hex64.;  
run;
```

md5=D5816F35916D1D9482FB0F1EC201101D

md5x=D5816F35916D1D9482FB0F1EC201101D

sha256=D69E68988157833272305AAF21F453C800346E8A3640DB6578E260215542E5D4

sha256x=D69E68988157833272305AAF21F453C800346E8A3640DB6578E260215542E5D4



```
data _null_;  
  length x varchar(*);  
  x = 'x';  
  x = repeat(x,99999);  
  md5 = hashing('md5',x);  
  put md5=;  
  md5x = md5(x);  
  put md5x=$hex32.;  
  sha256 = hashing('sha256',x);  
  put sha256=;  
  sha256x = sha256(x);  
  put sha256x=$hex64.;  
run;
```

md5=D5816F35916D1D9482FB0F1EC201101D

md5x=D5816F35916D1D9482FB0F1EC201101D

sha256=D69E68988157833272305AAF21F453C800346E8A3640DB6578E260215542E5D4

sha256x=D69E68988157833272305AAF21F453C800346E8A3640DB6578E260215542E5D4

```
File: x100000.txt (97.66 Kb)  
Hash (md5) of selected file (0.004 seconds):  
D5816F35916D1D9482FB0F1EC201101D
```



Hash tables can use
VARCHARs for keys and
values

```

data _null_;
  length star key value varchar(*);
  if _n_=1 then do;
    declare hash subdesc();
    subdesc.defineKey('key');
    subdesc.defineData('value');
    subdesc.defineDone();
  end;
  star = '*';
  key = 'key1'; value = 'value1';
  subdesc.add();
  key = 'key2'; value = repeat(star,44444);
  subdesc.add();
  key = repeat(star,55555); value = 'value3';
  subdesc.add();
  key = repeat(star,66666); value = repeat(star,77777);
  subdesc.add();

  value = ''; key = 'key1'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = 'key2'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,55555); rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,66666); rc = subdesc.find();
  l=length(value); put rc= l=;

run;

```



```

data _null_;
  length star key value varchar(*);
  if _n_=1 then do;
    declare hash subdesc();
    subdesc.defineKey('key');
    subdesc.defineData('value');
    subdesc.defineDone();
  end;
  star = '*';
  key = 'key1'; value = 'value1';
  subdesc.add();
  key = 'key2'; value = repeat(star,44444);
  subdesc.add();
  key = repeat(star,55555); value = 'value3';
  subdesc.add();
  key = repeat(star,66666); value = repeat(star,77777);
  subdesc.add();

  value = ''; key = 'key1'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = 'key2'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,55555); rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,66666); rc = subdesc.find();
  l=length(value); put rc= l=;

run;

```



```

data _null_;
  length star key value varchar(*);
  if _n_=1 then do;
    declare hash subdesc();
    subdesc.defineKey('key');
    subdesc.defineData('value');
    subdesc.defineDone();
  end;
  star = '*';
  key = 'key1'; value = 'value1';
  subdesc.add();
  key = 'key2'; value = repeat(star,44444);
  subdesc.add();
  key = repeat(star,55555); value = 'value3';
  subdesc.add();
  key = repeat(star,66666); value = repeat(star,77777);
  subdesc.add();

  value = ''; key = 'key1'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = 'key2'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,55555); rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,66666); rc = subdesc.find();
  l=length(value); put rc= l=;

run;

```



```

data _null_;
  length star key value varchar(*);
  if _n_=1 then do;
    declare hash subdesc();
    subdesc.defineKey('key');
    subdesc.defineData('value');
    subdesc.defineDone();
  end;

  star = '*';
  key = 'key1'; value = 'value1';
  subdesc.add();
  key = 'key2'; value = repeat(star,44444);
  subdesc.add();
  key = repeat(star,55555); value = 'value3';
  subdesc.add();
  key = repeat(star,66666); value = repeat(star,77777);
  subdesc.add();

  value = ''; key = 'key1'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = 'key2'; rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,55555); rc = subdesc.find();
  l=length(value); put rc= l=;
  value = ''; key = repeat(star,66666); rc = subdesc.find();
  l=length(value); put rc= l=;

run;

```



produces

rc=0 l=6

rc=0 l=44445

rc=0 l=6

rc=0 l=77778



So how big can a
VARCHAR get?



```
1      /* big.sas */
2      options fullstimer;
3      %macro doit(n);
4      data _null_;
5          length x varchar(*);
6          x = 'x';
7          x = repeat(x,input(&n,comma32.));
8          l = length(x);
9          put l=comma32.;
10         run;
11     %mend;
```

```

13          %doit('1');

l=2
          OS Memory          8024.00k
14          %doit('1,000');
l=1,001
          OS Memory          8024.00k
15          %doit('1,000,000');
l=1,000,001
          OS Memory          10000.00k
16          %doit('50,000,000');
l=50,000,001
          OS Memory          105692.00k
17          %doit('100,000,000');
l=100,000,001
          OS Memory          203352.00k
18          %doit('1,000,000,000');
l=1,000,000,001
          OS Memory          1961168.00k
19          %doit('1,050,000,000');
l=1,050,000,001
          OS Memory          2058828.00k
20          %doit('1,060,000,000');
l=1,060,000,001
          OS Memory          2078356.00k
21          %doit('1,070,000,000');

```

```

ERROR: Out of memory resizing varchar variable from 2 bytes to 1070006272 bytes.
FATAL: Insufficient memory to execute DATA step program. Aborted during the EXECUTION phase.
          OS Memory          1052948.00k

```





Python and SAS comparison with VARCHARs

```
import requests
```

```
tale_of_two_cities = requests.get('https://www.gutenberg.org/files/98/98-0.txt').text
```

```
tale_of_two_cities = tale_of_two_cities.replace('\n', ' ')
```

```
tale_of_two_cities = tale_of_two_cities.replace('\r', '')
```

```
tale_of_two_cities = tale_of_two_cities.replace('  ', ' ')
```

```
start = tale_of_two_cities.find('It was the best of times, it was the worst of times')
```

```
end = tale_of_two_cities.find('It is a far, far better thing that I do, than I have ever done');
```

```
print('start=', start, 'end=', end)
```

```
start= 2486 end= 768526
```

```

f = open('tale.txt','rb')
tale_of_two_cities = f.read()
f.close()

tale_of_two_cities = tale_of_two_cities.replace(b'\n',b' ')
tale_of_two_cities = tale_of_two_cities.replace(b'\r',b'')

start = tale_of_two_cities.find(b'It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find(b'It is a far, \
far better thing that I do, than I have ever done')

print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')

f = open('tale.txt','r',encoding='utf8')
tale_of_two_cities = f.read()
f.close()

tale_of_two_cities = tale_of_two_cities.replace('\n',' ')
tale_of_two_cities = tale_of_two_cities.replace('\r','')

start = tale_of_two_cities.find('It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find('It is a far, \
far better thing that I do, than I have ever done')

print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')

```



```
f = open('tale.txt','rb')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace(b'\n',b' ')
tale_of_two_cities = tale_of_two_cities.replace(b'\r',b'')
```

```
start = tale_of_two_cities.find(b'It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find(b'It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```

```
f = open('tale.txt','r',encoding='utf8')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace('\n',' ')
tale_of_two_cities = tale_of_two_cities.replace('\r,'')
```

```
start = tale_of_two_cities.find('It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find('It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```



```
f = open('tale.txt','rb')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace(b'\n',b' ')
tale_of_two_cities = tale_of_two_cities.replace(b'\r',b'')
```

```
start = tale_of_two_cities.find(b'It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find(b'It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```

```
f = open('tale.txt','r',encoding='utf8')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace('\n',' ')
tale_of_two_cities = tale_of_two_cities.replace('\r','')
```

```
start = tale_of_two_cities.find('It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find('It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```



```
f = open('tale.txt','rb')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace(b'\n',b' ')
tale_of_two_cities = tale_of_two_cities.replace(b'\r',b'')
```

```
start = tale_of_two_cities.find(b'It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find(b'It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```

```
f = open('tale.txt','r',encoding='utf8')
tale_of_two_cities = f.read()
f.close()
```

```
tale_of_two_cities = tale_of_two_cities.replace('\n',' ')
tale_of_two_cities = tale_of_two_cities.replace('\r,'')
```

```
start = tale_of_two_cities.find('It was the best of times, \
it was the worst of times')
end = tale_of_two_cities.find('It is a far, \
far better thing that I do, than I have ever done')
```

```
print(f'start= {start+1} end= {end+1} (adjusted by 1 to be 1-based)')
```





```
start= 2666 end= 772152 (adjusted by 1 to be 1-based)
start= 2664 end= 758256 (adjusted by 1 to be 1-based)
```

```

filename dickens 'tale.txt';
data _null_; infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;
  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
stop;
run;

```



```

filename dickens 'tale.txt';
data _null_; infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;
  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
stop;
run;

```



```
filename dickens 'tale.txt';
data _null_ ; infile dickens end=eof encoding=utf8;
    length tale_of_two_cities varchar(*);
    tale_of_two_cities = '';
    do while(not eof);
        input;
        tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
    end;
| start = kindexb(tale_of_two_cities,
                'It was the best of times, it was the worst of times');
end = kindexb(tale_of_two_cities,
              'It is a far, far better thing that I do, than I have ever done');
put start= end=;
start = kindex(tale_of_two_cities,
               'It was the best of times, it was the worst of times');
end = kindex(tale_of_two_cities,
             'It is a far, far better thing that I do, than I have ever done');
put start= end=;
stop;
run;
```

```
filename dickens 'tale.txt';
data _null_; infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;

  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;

  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
stop;
run;
```

```

filename dickens 'tale.txt';
data _null_ infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;
  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
stop;
run;

```



```

filename dickens 'tale.txt';
data _null_; infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;
  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  stop;
run;

```



```
filename dickens 'tale.txt';
data _null_; infile dickens end=eof encoding=utf8;
  length tale_of_two_cities varchar(*);
  tale_of_two_cities = '';
  do while(not eof);
    input;
    tale_of_two_cities = tale_of_two_cities || _infile_ || ' ';
  end;
  start = kindexb(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindexb(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  start = kindex(tale_of_two_cities,
    'It was the best of times, it was the worst of times');
  end = kindex(tale_of_two_cities,
    'It is a far, far better thing that I do, than I have ever done');
  put start= end=;
  stop;
run;
```

NOTE: DATA STEP stopped due to looping.



With a Windows-1252 (single byte encoding) the above produces

```
start=2664 end=758256  
start=2664 end=758256
```

With a UTF-8 (multibyte) session encoding, the above produces

```
start=2666 end=772152  
start=2664 end=758256
```



With a Windows-1252 (single byte encoding) the above produces

```
start=2664 end=758256  
start=2664 end=758256
```

With a UTF-8 (multibyte) session encoding, the above produces

```
start=2666 end=772152  
start=2664 end=758256
```

From our Python code:

```
start= 2666 end= 772152 (adjusted by 1 to be 1-based)  
start= 2664 end= 758256 (adjusted by 1 to be 1-based)
```

Conclusions



- VARCHARs are a valuable new feature (9.4m5 or later)
- Overcomes the 32K limit in DATA steps and macro
- Overcomes the silent truncation problem
- Provide functions with VARCHARS to get them returned
- Only persistent with CAS tables, but fully usable in DATA steps
- Makes more sense for Python users

Contact Information



- Name: Rick Langston
- Email: RickLangston1955@gmail.com