

Purrfectly Fabulous Feline Functions

Louise S. Hadden, Abt Associates Inc., Cambridge, MA

ABSTRACT

Do the following snippets of SAS® code look familiar?

```
length stcounty $ 5 citystzip $ 60;

stcounty=trim(left(put(state,z2.))||trim(left(put(county,z3.))));
citystzip=trim(left(city))||', '||trim(left(statecode))||' '||trim(left(zip));
```

Now there's an easier way! Leave those vertical bars behind and join us as we explore the purrfectly fabulous feline functions (and call routines) available in SAS® version 9.1 and above. Enhancements available in version 9.2 will be demonstrated.



INTRODUCTION

Version 9 introduced four new functions (CAT, CATS, CATT, and CATX) and three new call routines (CALL CATS, CALL CATT, and CALL CATX) that replace the clunky syntax shown above, and add some additional enhanced capability to character string concatenation. In Version 9.2, CATQ was added. We will explore the five (fabulous) CAT functions and three CAT CALLS and demonstrate how string concatenation can be accomplished more easily and efficiently using these functions and call routines. In addition, the difference between the CAT functions and CAT CALLS will be briefly discussed. Examples were run with SAS 9.2 on Windows XP, and SAS 9.2 on Windows Server (x64). Leaving appropriate spaces out is deliberate so that the action of the functions and call routines can be observed. Performance using the concatenation operator, CAT functions and CAT CALL routines will be compared on the two different platforms using an identical one million record file.



ORIGINS OF CAT CALLS AND FUNCTIONS

The customary syntax seen above, `varx=trim(left(vary))) || trim(left(varz))`, consists of two functions (TRIM and LEFT) paired with the concatenation operator (||). TRIM removes trailing blanks, while LEFT left aligns text values. LEFT does not remove leading blanks. It simply moves leading blanks to the end of the string, which is why you usually see LEFT used in conjunction with (and inside) the TRIM function for string concatenation. Over the years, SAS has enhanced the function (pardon the pun) of TRIM and LEFT. A newer function TRIMN also strips trailing blanks: the difference between TRIM and TRIMN is the end result in the case of the argument being missing. TRIM results in a single blank (length 1) for a missing argument while TRIMN results in a null (length 0.) This might come in handy, for example, when concatenating first, middle and last names, in which middle names are frequently missing. Another newer function, STRIP, is the equivalent of TRIMN(LEFT(varx)) but is obviously more convenient. STRIP removes both leading and trailing blanks. The CAT CALLS and functions build on the original and derivative functions to offer a complete array of concatenation options.

EXAMPLES AND RESULTS OF USING THE CONCATENATION OPERATOR AND THE CAT FUNCTION

Example 1:

```
35      data temp1;
36          set dd.sub2003asm ;
37          fullname=aa1a||aa1b||aa1c;
38      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP1 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):
 real time 0.54 seconds

cpu time 0.54 seconds

Result:

Using concatenation operator x64 n=1M

fullname

ERMA FLETCHER

Variables in Creation Order

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	31	

Example 2:

```

49      data temp2;
50          set dd.sub2003asm ;
51          fullname=trim(left(aa1a))||trimn(left(aa1b))||trim(left(aa1c));
52      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP2 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

real time 0.76 seconds

cpu time 0.76 seconds

Result:

Using concatenation operator and trim/trimn/left functions x64 n=1M

fullname

ERMAFLETCHER

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	31	

Example 3:

```

63      data temp3;
64          set dd.sub2003asm ;
65          fullname=cat(aa1a,aa1b,aa1c);
66      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP3 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):
 real time 0.68 seconds
 cpu time 0.68 seconds

Result:

Using CAT function x64 n=1M

fullname

ERMA FLETCHER

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	200	



DIFFERENCES BETWEEN CALL ROUTINES AND FUNCTIONS

The primary differences between the CAT CALLS and the corresponding functions are in syntax and performance. Results are equivalent. Call routines have better performance (i.e. they run faster) than functions which result in the same product.

The syntax differs in that in call routines, the result is incorporated into the statement, or CALL. In functions, the result is on the left hand side of an equal sign and the function statement.

EXAMPLES AND RESULTS OF THE CATS FUNCTION AND CALL CATS

Example 4:

```

139      data temp7a;
140          length fullname $ 40;
141          set dd.sub2003asm ;
142          fullname=cats(aa1a,aa1b,aa1c);
143      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP7A has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):
 real time 0.53 seconds
 cpu time 0.53 seconds

Result:

Using CATS function x64 adding length statement n=1M

fullname

ERMAFLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	

2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

Example 5:

```
138      data temp8;
139          length fullname $ 40;
140          set dd.sub2003asm ;
141          call cats(fullname,aa1a,aa1b,aa1c);
142      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP8 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
 real time 0.50 seconds
 cpu time 0.50 seconds

Result:

Using CALL CATS x64 n=1M (INITIALIZATION REQUIRED)

fullname

ERMAFLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

In the end, anything that increases processing speed and efficiency while reducing the keystrokes necessary to accomplish goals is purrfectly fine with most SAS programmers! For concatenation, the hierarchy of efficiency is CAT CALLS at the top, CAT FUNCTIONS in the middle, and an assemblage of functions and the concatenation operator at the bottom.



IMPORTANT NOTE ON FELINE FUNCTIONS AND VARIABLE LENGTH

SAS calculates a length for resulting variables created by using the concatenation operator by adding the lengths of the source variables, or arguments, together. This is very different from how the CAT functions operate. If not specified in a length statement prior to the invocation of a function (or length specified via some other method), the length of the result of the CAT functions defaults to 200. It is important to set the length of your variables that you will create from a character function to keep variable lengths, processing costs and storage costs reasonable. Conversely, it is also important to set the length of created variables long enough to accommodate the longest string created by concatenation, so that your result is not truncated. As you can see, use of the length statement also significantly increases processing speed when you reduce the length of the variable from 200 to 40.

EXAMPLES AND RESULTS OF THE CAT FUNCTION WITH AND WITHOUT A LENGTH STATEMENT

Example 6:

```
63      data temp3;
64          set dd.sub2003asm ;
65          fullname=cat(aa1a,aa1b,aa1c);
66      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP3 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
real time 0.68 seconds
cpu time 0.68 seconds

Result:

Using CAT function x64 n=1M

fullname

ERMA		FLETCHER		
#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	200	

Example 7:

```
109      data temp6;
110          length fullname $ 40;
111          set dd.sub2003asm ;
112          fullname=cat(aa1a,aa1b,aa1c);
113      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP6 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
real time 0.45 seconds
cpu time 0.45 seconds

Result:

Using CAT function x64 adding length statement n=1M

fullname

ERMA		FLETCHER		
#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM



THE CAT FUNCTIONS (CAT, CATS, CATT, CATX AND CATQ)

CAT (available in 9.1)

The CAT function is identical to simply using the concatenation operator (||) to string character literals, variables or expressions together. No leading or trailing blanks are removed from arguments.

See examples 6 and 7 above.

CATS (available in 9.1)

The CATS function removes both leading and trailing blanks before concatenating character arguments. It is the equivalent of using the STRIP function and the concatenation operator. The end result will not contain any blanks, even if you specify a blank as an argument, which might (or might not) be what you want. A good mnemonic device is that “S” is equivalent to “STRIP”. The character arguments are stripped of any blanks before being joined.

See example 4 above.

CATT (available in 9.1)

The CATT function removes only trailing blanks before concatenating character arguments. The end result will contain blanks if any arguments have LEADING blanks. A good mnemonic for this function is “T” is equivalent to “TRAILING” or “TRUNCATE.” Trailing blanks are removed from character arguments before they are joined.

EXAMPLE AND RESULTS OF THE CATT FUNCTION

Example 8:

```
210      data temp12;
211          length fullname $ 40;
212          set dd.sub2003asm (rename=(aa1a=var1 aa1b=var2 aa1c=var3)) ;
213          fullname=catt(of var1-var3);
214      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP12 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
 real time 0.51 seconds
 cpu time 0.51 seconds

Results:

Using CATT function x64 n=1M OF SHORTCUT

fullname

ERMAFLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	var1	Char	12	AA1A-FIRST-NM

6	var2	Char	1	AA1B-MIDDLE-INTIAL
7	var3	Char	18	AA1C-LAST-NM

CATX (available in 9.1)

The CATX function performs just like the CATS function, i.e. stripping trailing and leading blanks from arguments before joining, but it adds the eXtra capability of inserting the delimiter of your choice between arguments. This is handy if you want a space or comma (or something else) between arguments (such as a full name or address) but don't want any eXtra spaces. While you could use another function, COMPBL, to remove excess blanks, CATX does it all at once. A good mnemonic for CATX is "X" is for "eXtra" – the CATX function "adds" a delimiter (or something extra) between arguments.

EXAMPLE AND RESULTS OF THE CATX FUNCTION

```
Example 9:
224      data temp13;
225          length fullname $ 40;
226          set dd.sub2003asm ;
227          fullname=catx(" ",aa1a,aa1b,aa1c);
228      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP13 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
 real time 0.56 seconds
 cpu time 0.56 seconds

Results:

Using CATX function x64 n=1M

fullname

ERMA FLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

CATQ (available in 9.2)

The CATQ function is the newest member of the fabulous feline function litter. It is similar to the CATX function, with the addition of quotation marks around arguments in the end result made possible. There is an array of modifiers that can be used with this function to finely control the result. The online documentation for SAS 9.2 (Language Reference: Dictionary) is a good and comprehensive discussion of the many modifiers available and the syntax for using these modifiers, among which are [not a comprehensive list!]: 1 (use single quotation marks), 2 (use double quotation marks), a (add quotation marks to all arguments), c (use a comma as a delimiter), s (trim leading and trailing blanks from arguments), t (trim trailing blanks from arguments), and x (convert item arguments to hexadecimal literals if they contain nonprintable characters.) A good mnemonic for CATQ is "Q" is for "quotation marks"; the CATQ function has the capacity to add single or double quotation marks to selection portions of the result if desired using the many modifiers available. Remember to add some extra length for the quotation marks, or you will get an error if the results of the CATQ function exceed the specified length.

EXAMPLE AND RESULTS OF THE CATQ FUNCTION

Example 10:

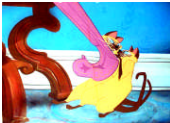
```
252      data temp15;
253          length fullname $ 50;
254          set dd.sub2003asm ;
255          fullname=catq('as',aa1a,aa1b,aa1c);
256      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.
NOTE: The data set WORK.TEMP15 has 1000000 observations and 7 variables.
NOTE: DATA statement used (Total process time):
 real time 0.60 seconds
 cpu time 0.60 seconds

Results:

Using CATQ function x64 n=1M

fullname				
"ERMA" "" "FLETCHER"				
#	Variable	Type	Len	Label
1	fullname	Char	50	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM



IMPORTANT NOTE ON CALLING CATS AND VARIABLE LENGTH

The result variable **MUST** be initialized as a character variable prior to the CALL routine. If the result variable is not initialized, SAS assumes the default (that the variable is numeric) and an error occurs (a missing value is produced.) If you initialize the result variable to a single blank, the result will be truncated at a length of one. Therefore, as with CAT functions, it is a good practice to specify the length of the resulting variable directly under the data statement, making sure to set the length of created variables long enough to accommodate the longest string created by concatenation.



CALLING THE CATS

CALL CATS, CALL CATT and CALL CATX all behave more or less like their corresponding functions described above, except that the resulting character variable is included in the CALL. For example:

```
CALL CATS(result,item1,item2,...itemn);
```

Items, or arguments, can be numeric instead of character; numeric arguments are converted to character using BESTw. Format. The result **MUST** be a character variable, and the length of the character variable should be long enough to accommodate the result. As discussed above, this result variable must be initialized (and the appropriate length set) before invocation of the CAT CALL.

You can also use the short cut of (of var1-varn) in CAT CALL routines and CAT functions. For example:

```
CALL CATT(result,of item1 - itemn);
```

Also see example 8 above and example 11 below.

It is important to note that unlike the CATX function, CALL CATX does not include a delimiter surrounding blank arguments. This may, or may not, be what you want.

For an example of CALL CATS, see Example 5 above.

EXAMPLE AND RESULTS OF CALL CATT

Example 11:

```
195      data temp11;
196          length fullname $ 40;
197          set dd.sub2003asm (rename=(aa1a=var1 aa1b=var2 aa1c=var3)) ;
198          call catt(fullname,of var1-var3);
199      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP11 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```
real time      0.48 seconds
cpu time       0.48 seconds
```

Results:

Using CALL CATT x64 n=1M OF SHORTCUT (INITIALIZATION REQUIRED)

fullname

ERMAFLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	var1	Char	12	AA1A-FIRST-NM
6	var2	Char	1	AA1B-MIDDLE-INTIAL
7	var3	Char	18	AA1C-LAST-NM

EXAMPLE AND RESULTS OF CALL CATX

Example 12:

```
238      data temp14;
239          length fullname $ 40;
240          set dd.sub2003asm ;
241          call catx(" ",fullname,aa1a,aa1b,aa1c);
242      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP14 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```
real time      0.53 seconds
cpu time       0.53 seconds
```

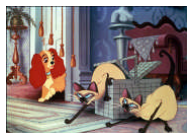
Results:

Using CALL CATX x64 n=1M (INITIALIZATION REQUIRED)

fullname

ERMA FLETCHER

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM



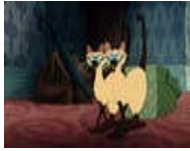
CONCLUSION

Use of the fabulous feline functions and call routines saves time in both coding and in processing. While results of using the combination method, CAT functions and CAT CALL routines are all the same, the feline functions and call routines use less code and process concatenations faster. It is purrfectly clear that the CAT functions and call routines are valuable additions to the SAS® programmer's toolbox. Try them out, and you'll be become a CAT lover too!

Category	CPU Time Windows Server x64 SAS 9.2	CPU Time Windows XP P4 SAS 9.2	Length of Result
Concatenation operations alone / no length statement	.54sec	.66sec	31 (sum of variable lengths)
Concatenation operations with TRIM/TRIMN/LEFT functions / no length statement	.76sec	.88sec	31
CAT function / no length statement	.68sec	1.34sec	200
Concatenation operations alone / length statement	.51sec	.70sec	40
Concatenation operations with TRIM/TRIMN/LEFT functions / length statement	.79sec	.95sec	40
CAT function / length statement	.46sec	.93sec	40
CATS function / no length statement	.73sec	1.38sec	200
CATS function / length statement	.53sec	.78sec	40
CALL CATS*	.49sec	.99sec	40
CATT function / no length statement	.71sec	1.12sec	200
CATT function / length statement / code shortcut	.51sec	.89sec	40
CALL CATT*	.48sec	.92sec	40
CALL CATT* / code shortcut	.48sec	.80sec	40
CATX function / length statement / insert blank	.56sec	.85sec	40
CALL CATX* / insert blank	.53sec	1.01sec	40
CATQ function / length statement	.60	1.04sec	50

*Initialization of result variable required in CAT CALLs

REFERENCES & RECOMMENDED READING



<http://support.sas.com/documentation/onlinedoc/base/index.html>

<http://support.sas.com/rnd/papers>

<http://support.sas.com/samples>

<http://support.sas.com/documentation/cdl/en/lrdict/59540/HTML/default/a000245852.htm>

<http://support.sas.com/documentation/cdl/en/whatsnew/61982/HTML/default/lrdictwhatsnew902.htm#a002986580>

<http://support.sas.com/kb/24/513.html>

Cody, Ron. 2004. *SAS® Functions by Example*. Cary, NC: SAS Institute Inc.



ACKNOWLEDGMENTS & DISCLAIMERS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

All images are free to download from www.disney.com. They are from the movies Lady and the Tramp (the famous Si and Am), the Aristocats, and Oliver and Company.

CONTACT INFORMATION



Your comments and questions are valued and encouraged. Contact the author at:

Louise Hadden
Abt Associates Inc.
55 Wheeler St.
Cambridge, MA 02138
(617) 349-2385 (work)
(617) 349-2675 (fax)
Email: louise_hadden@abtassoc.com

Sample code is available from the author upon request.