Dos and Don'ts in Macro Parameter Handling

Ivo Marek
Entimo AG
Stralauer Platz 33-34
10243 Berlin
ima@entimo.de

Zusammenfassung

Die Auswahl der korrekten Quoting-Funktion scheint nicht immer ganz klar zu sein. Warum sich also die Mühe machen eine auszuwählen? Es geht auch ohne Quoting-Funktionen.

Schlüsselwörter: Quoting, SAS-Makro

1 Das Problem

Ein Großteil der Anwendungsentwicklung mit dem SAS System beinhaltet Makros und die nützlichsten unter ihnen enthalten Parameter. Durch Parametrisierung können Programme in ihrer Anwendung flexibilisiert werden. Das Problem für den Entwickler ist die Sicherstellung der korrekten Verarbeitung der Parameter. Bei der Verarbeitung werden zunächst die Parameter überprüft bevor sie anschließend weiterverarbeitet werden und der Anwender bezüglich seiner Eingaben das gewünschte Ergebnis erhält. Die Überprüfung dient dazu, dass keine Ergebnisse aufgrund falscher Parameterwerte erzeugt werden. Der Anwender erhält bei einer unzulässigen Eingabe eine sofortige Rückmeldung, damit er bzw. sie die Parameterwerte korrigieren kann. Nur wenn die Überprüfung erfolgreich war, soll der Anwender ein Ergebnis erhalten. Für den Entwickler heißt das, dass er bzw. sie bei der Überprüfung alle unzulässigen Eingaben und bei der Weiterverarbeitung nur noch die zulässigen Eingaben berücksichtigen muss.

Ein Fehler in der Überprüfung, d.h. eine unzulässige Eingabe wird nicht erkannt, führt zu einem unerwünschten Ergebnis, da bei der Weiterverarbeitung unzulässige Eingaben nicht mehr berücksichtigt werden. Ein unerwünschtes Ergebnis kann zum einen eine vom SAS System erzeugte Warnung bzw. Fehlermeldung und zum anderen auch ein Absturz des Systems sein. Ein Fehler in der Weiterverarbeitung kann zu den gleichen unerwünschten Ergebnissen führen.

Die zweite Art von Fehler ist schwerwiegender. Falls der Anwender keine fehlerhafte Eingabe getätigt hat und das SAS System eine Warnung bzw. einen Fehler erzeugt, ist nicht gewährleistet, dass das Ergebnis korrekt ist. Diese Meldungen sind oft darauf zurückzuführen, dass Quoting-Funktionen vom Entwickler unsachgemäß eingesetzt werden.

Das untenstehende Beispiel zeigt eine unsachgemäße Anwendung von Quoting-Funktionen. Unter Quoting-Funktionen versteht man, dass Symbole der Makro-Sprache als gewöhnliche Zeichen interpretiert werden, d.h., dass die Symbole maskiert werden. Zum Beispiel werden innerhalb der Makro-Sprache alle Symbole, die von zwei Apostrophen eingeschlossen sind als gewöhnliche Zeichen interpretiert. Ein Apostroph muss deshalb durch Quoting-Funktionen maskiert werden, damit es als gewöhnliches Zeichen interpretiert wird.

```
%MACRO Set (TITLE = );
    %LET TITLE = %TRIM(%NRBQUOTE(%LEFT(%NRBQUOTE(&TITLE))));
    TITLE "&TITLE";
%MEND Set;
```

Das Beispiel erzeugt mit dem Aufruf %Set(TITLE = %NRSTR(Jerry%'s)) den Fehler 'ERROR: Literal contains unmatched quote.'. Der Fehler ist darauf zurückzuführen, dass durch die Makro-Funktion %LEFT das Apostroph nicht mehr maskiert ist und daher nicht als gewöhnliches Zeichen interpretiert wird. Der Anwender kann den SAS Fehler vermeiden, indem der Apostroph durch

```
%Set(TITLE = %NRSTR(Jerry%NRSTR(%BQUOTE(%SYSFUNC(BYTE(39))))s))
```

oder

```
%Set(TITLE = %NRSTR(Jerry%NRSTR(%NRSTR(%%%%%'))s))
```

maskiert wird. Das ist dem gewöhnlichen Anwender nicht zumutbar und erfordert tiefgreifende SAS Kenntnisse. Der Grund warum zum Beispiel die erste Lösung funktioniert ist, dass nach der Anwendung von %LEFT und %TRIM der Apostroph maskiert ist. Denn nach Anwendung von %LEFT als auch von %TRIM wird das Ergebnis zuerst maskiert. Die Maskierung geschieht zum einen durch %NRSTR und zum anderen durch %BQUOTE. Bei %NRSTR wird %BQUOTE maskiert und bei %BQUOTE wird zuerst der Apostroph erzeugt, bevor er maskiert wird.

2 Die Lösung

Die sicherste Methode ist die Überprüfung und Bearbeitung von Parametern innerhalb eines DATA STEPs (DATA _NULL_). Mit Hilfe der Funktion SYMGET lassen sich Inhalte von Makro Variablen in eine SAS Tabelle schreiben und mit CALL SYMPUT wieder in Makro Variablen zurückschreiben. Es ist auch möglich in einem Schritt SYMGET und CALL SYMPUT anzuwenden:

```
CALL SYMPUT('MVAR02', <Funktion>(SYMGET('MVAR01'))).
```

Hierbei wird der Inhalt der Makro Variablen MVAR01 virtuell in eine SAS Tabelle geschrieben. Dann wird eine beliebige Funktion angewendet und anschließend der veränderte Inhalt oder ein Teilergebnis in die Makro Variable MVAR02 zurückgeschrieben. Das hat den Vorteil, dass kein Attribut mit einer vorgegebenen Länge angelegt werden muss, in welchem der Inhalt abgelegt ist. Alle Teilergebnisse liegen in Makro Variablen vor. Obwohl keine Attribute angelegt werden, sind die Inhalte der Teilergebnisse auf 32767 Zeichen anstatt auf 65534 Zeichen beschränkt.

Mit der 'DATA STEP'-Lösung lassen sich Quoting-Funktionen vermeiden. Die Verbesserung lässt sich an dem obigen Beispiel zeigen. Hierbei sieht die 'DATA STEP'-Lösung folgendermaßen aus:

```
%MACRO Set (TITLE = );
    DATA _NULL_;
        CALL SYMPUT('TITLE',
        "'"||STRIP(PRXCHANGE("s/'/''/",-1,SYMGET('TITLE')))||"'");
    RUN;
    TITLE &TITLE;
%MEND Set;
```

Durch die gleichzeitige Verwendung von SYMGET und CALL SYMPUT und das Einschließen der Zeichenkette durch zwei Apostrophe, werden alle Zeichen bei späterer Verwendung als gewöhnliche Zeichen interpretiert. Hierbei müssen innerhalb der Zeichenkette alle einfachen Apostrophe durch zwei ersetzt werden. Der Vorteil dieser Lösung ist, dass SAS keine Warnungen und Fehler durch unsachgemäße Anwendung von Quoting-Funktionen erzeugt. Der Nachteil besteht darin, dass dynamische Elemente einer Zeichenkette im Sinne von Makro-Variablen nicht aufgelöst werden können. Dieser Nachteil lässt sich durch eine Erweiterung der 'DATA STEP'-Lösung beheben. Mit der Erweiterung werden dynamische Elemente während der Laufzeit ohne SAS Fehler aufgelöst. Eine mögliche Erweiterung ist durch das folgende Beispiel gegeben.

```
%MACRO Set (TITLE = %NRSTR(Ben&Jerry%'s &REPLACE));
    %LET REPLACE = Spectacular Speculoos;
    DATA _NULL_;
        ATTRIB VAR LENGTH = $ 32767.;
        * Scanne TITLE nach &<SASNAME> -> VAR ;
        IF   NOT SYMLOCAL(VAR)
        THEN DO;
             CALL SYMPUTX(VAR, BYTE(15)||VAR,'L');
        END;
        * Wenn alle Makro-Variablen angelegt sind ;
        CALL SYMPUT('TITLE',
             RESOLVE( PRXCHANGE("s/&(?=&)/"||BYTE(15)||"/i",-1,
                      PRXCHANGE("s/%/"||BYTE(16)||"/i",-1,
                      PRXCHANGE("s/'/"||BYTE(17)||"/",-1,
                      SYMGET('TITLE'))))
                    )
                   );
    RUN;
%MEND Set;
```

Bei diesem Beispiel werden potentielle Makro-Referenzen einer Zeichenkette auf das Vorhandensein überprüft. (Es werden nur einfache Referenzen betrachtet.) Dies geschieht durch die Verwendung der Funktion SYMLOCAL. Im Falle, dass die Makro-Variable nicht lokal existiert, wird sie im Sinne von %LET MVAR = BYTE(15)||MVAR lokal mit CALL SYMPUTX angelegt. Beispielsweise, wird die

Makro Variable 'Jerry' mit %LET JERRY = %NRSTR(&)JERRY erzeugt. Dadurch wird für jede nicht vorhandene Makro-Variable eine Pseudo-Variable angelegt, die als Inhalt sich selbst mit einem maskierten '&' enthält. Danach kann man mit RESOLVE alle Referenzen und Pseudo-Referenzen innerhalb des DATA STEPs auflösen. Dabei ist zu beachten, dass Referenzen, die einem Apostroph folgen, nicht aufgelöst werden. Beispielsweise wird &MVAR in der Zeichenkette "Jerry's &MVAR" mit RESOLVE nicht aufgelöst. Außerdem werden potentielle Makro-Aufrufe, d.h. Zeichenketten die mit dem Zeichen % beginnen, auch ausgeführt. Falls das Makro nicht existiert wird eine SAS Warnung ausgegeben. Deshalb müssen die Zeichen % und ' durch ihre Äquivalente, BYTE(16) bzw. BYTE(17), ersetzt werden. Die Ersetzung führt dazu, dass der Interpreter die Zeichen nicht als % und ' erkennt. Jedoch werden BYTE(16) und BYTE(17) nach Anwendung von RESOLVE durch % und ' ersetzt. Vor der Anwendung von RESOLVE müssen führende '&'s vor einer potentielle Makro-Referenz durch BYTE(15) ersetzt werden. Diese stellt sicher, dass Konstrukte wie zum Beispiel &&&MVAR nicht als indirekte Referenzen interpretiert werden. Ist die Länge einer vermeintlichen Makro-Variablen länger als 32, füge in den ursprünglichen String vorher einen Punkt ein (.) und kürze vorher den Namen auf 32 Zeichen:

```
IF   LENGTHN(VAR)>32
THEN DO;
     VAR = SUBSTRN(VAR,1,32);
     CALL SYMPUT('TITLE',
          PRXCHANGE("s/&+("||STRIP(VAR)||")/&\1./",1,
          SYMGET('TITLE')));
END;
```

3 Appell

Die Überprüfung und Bearbeitung von Makro-Parametern sollte immer innerhalb eines DATA STEPs und nicht innerhalb der Makro-Umgebung erfolgen. Denn innerhalb eines DATA STEPs ist eine Maskierung von Symbolen nicht zwangsläufig nötig. Eine Ausnahme bildet die Verwendung der Funktion RESOLVE mit deren Hilfe Referenzen auf Makro-Variablen aufgelöst werden sollen/können. Hierbei können gezielt einzelne Symbole der Makro-Sprache maskiert werden - [&] durch ASCII 15, [%] durch ASCII 16 und ['] durch ASCII 17 -, sodass gezielt Referenzen aufgelöst oder Makros ausgeführt werden können. Insbesondere ermöglicht die Maskierung von ['] eine Auflösung von Referenzen, die in einer Zeichenkette hinter einem Apostroph ['] stehen. Die Kommunikation zwischen DATA STEP und Makro-Umgebung erfolgt mit SYMGET und CALL SYMPUT. Die Makro-Umgebung erhält hierdurch die Funktion eines Speichers: Zwischenergebnisse werden in Makro-Variablen gespeichert und nicht in DATA STEP Variablen. Zeichenketten sollten zum Schluss durch Apostrophe ['] geschützt werden.

Anhang A Vollständiges Beispiel

Im Gegensatz zum In-Text-Beispiel liegen hier alle Teilergebnisse in Makro Variablen vor.

```
%MACRO Set (TITLE = %NRSTR(Ben&Jerry%'s &REPLACE));
    %LOCAL REPLACE TMP1 TMP2;
    %LET   REPLACE = Spectacular Speculoos;
    DATA _NULL_;
        CALL SYMPUT('TMP1',SYMGET('TITLE'));

        SASNAME = '[A-Za-z_][A-Za-z_0-9]*';
        REGEXP  = PRXPARSE('/&('||STRIP(SASNAME)||')(.*)/');

        DO WHILE (PRXMATCH(REGEXP,SYMGET('TMP1')));

            CALL SYMPUT('TMP2',
                 STRIP(PRXPOSN(REGEXP,1,SYMGET('TMP1'))));
            CALL SYMPUT('TMP1',
                 STRIP(PRXPOSN(REGEXP,2,SYMGET('TMP1'))));

            IF   LENGTHN(SYMGET('TMP2'))>32
            THEN DO;
                 CALL SYMPUT('TMP2',SUBSTRN(SYMGET('TMP2'),1,32));
                 CALL SYMPUT('TITLE',PRXCHANGE(
                      "s/&+("||SYMGET('TMP2')||")/&\1./",1,
                      SYMGET('TITLE')));
            END;

            IF   NOT SYMLOCAL(SYMGET('TMP2'))
            THEN DO;

                 CALL SYMPUTX(STRIP(SYMGET('TMP2')),
                      BYTE(15)||STRIP(SYMGET('TMP2')),'L');
            END;

        END;

        CALL SYMPUT('TITLE',
             RESOLVE( PRXCHANGE("s/&(?=&)/"||BYTE(15)||"/i",-1,
                      PRXCHANGE("s/%/"||BYTE(16)||"/i",-1,
                      PRXCHANGE("s/'/"||BYTE(17)||"/",-1,
                      SYMGET('TITLE'))))
                    )
                   );

        CALL SYMPUT('TITLE',
        "'"||PRXCHANGE("s/'/''/",-1,SYMGET('TITLE'))||"'");

    RUN;
    TITLE &TITLE;
%MEND Set;
```