Learning SAS: Extracting Substrings from the Right Side of a String

Authored by

Mohammed looti

November 14, 2025

Mastering String Extraction from the Right in SAS

In complex data analysis environments, the ability to manipulate character data precisely is paramount. Data professionals working within the SAS system frequently encounter scenarios where specific portions of a string must be isolated for cleaning, feature creation, or reporting. While extracting substrings from the beginning (the left) is intuitive, data structures often require extracting information consistently located at the tail end of a text field. Examples include file version numbers, country codes appended to customer IDs, or trailing error markers.

The standard tool for this operation in SAS programming is the **SUBSTR** function. By default, **SUBSTR** initiates its extraction process from the left side. However, by combining it strategically with another fundamental SAS function, we can dynamically calculate the required starting point, effectively allowing us to extract characters from the **right** side of any string variable.

This guide will detail this powerful technique, demonstrating how to achieve reliable and flexible right-side substring extraction. Mastering this method significantly enhances your data preparation toolkit, ensuring cleaner and more targeted transformations of textual data within the rigorous SAS framework.

The Core Mechanism: Understanding the SUBSTR Function

The SUBSTR function is the cornerstone of character variable manipulation in SAS. Its purpose is to return a specified segment, or **substring**, from a larger source string. To use it effectively, it is essential to internalize its structure and argument requirements.

The standard syntax for the SUBSTR function requires three primary arguments:

SUBSTR(Source, Position, N)

A precise understanding of these arguments is key to manipulating the function for right-side extraction:

Source: This is the mandatory input string, which can be a literal value enclosed in quotes or, more commonly, a character variable from your input dataset.

Position: This numeric value determines the starting point of the extraction. Crucially, SAS indexing is 1-based, meaning the count begins at 1. This position is always calculated by counting from the **left** side of the string.

N: This optional numeric argument specifies the exact number of characters the function should read, starting from the calculated **Position**. If N is omitted, SUBSTR extracts all characters from the starting **Position** to the very end of the source string.

Since the **Position** argument is inherently left-oriented, the challenge of right-side extraction boils down to calculating the correct left-based starting position that corresponds to the desired number of characters counted from the right. This is where the powerful LENGTH function comes into play, providing the necessary variable dimensioning for our calculation.

Calculating Position from the Right using LENGTH

To overcome the left-oriented nature of the SUBSTR function, we must dynamically generate the starting **Position** argument using the total length of the source string. The LENGTH function provides this crucial information, returning the exact number of non-blank characters in the string.

If we want to extract **N** characters from the right, we must find the position (P) such that counting **N** characters starting from P brings us exactly to the end of the string. The formula to calculate this left-based starting position (P) is straightforward:

```
LENGTH(Source_String) - N + 1
```

The logic behind adding 1 is essential because SAS is 1-based. If a string has a length of 10 and we want the last 3 characters (N=3), subtracting 3 from 10 gives us 7. If we started at position 7, we would extract characters 7, 8, 9, and 10 (four characters). To correctly extract only characters 8, 9, and 10 (three characters), we must start at position 8. Therefore, the formula is 10 - 3 + 1 = 8. This ensures the extraction window is precisely aligned to capture the exact number of trailing characters specified by N.

This clever combination of SUBSTR and LENGTH forms a highly robust and efficient mechanism for right-side extraction, frequently used in DATA steps to generate new variables. Below is the general syntax application:

data new_data;
set original_data;
last_three = substr(team, length(team)-2, 3); /* N=3; Position = L - 3 + 1 = L - 2 */
run;

In this example, the expression `length(team)-2` determines the precise left-based starting point needed to extract three characters. The final argument, `3`, guarantees that exactly three characters are taken from that starting position, thus capturing the last three characters of the `team` string consistently across all observations.

Practical Demonstration: Extracting Trailing Identifiers

To ensure a clear understanding of this technique, we will walk through a concrete example. We

aim to extract the last three characters of a list of basketball team names, treating these as unique trailing identifiers. This task requires implementing the combined SUBSTR and LENGTH function approach within a SAS DATA step.

We begin by creating a sample dataset named `original_data`. This dataset simulates real-world data containing team names (character strings) and their corresponding scores (numeric values). The creation and initial view of the dataset are performed using the following SAS code:

```
/*create dataset: original_data*/
data original_data;
input team $ points;
datalines;
Mavericks 104
Thunder 99
Rockets 116
Spurs 98
Pistons 99
Pelicans 105
Warriors 119
Blazers 113
Nuggets 100
Kings 123
;
run;

/*view dataset*/
proc print data=original_data;
```

This code snippet utilizes the DATA step to define and populate the source data, followed by a PROC PRINT to display the contents, ensuring the source data is correctly structured before manipulation. The resulting output, showing the initial state of the data, is displayed below:

Obs	team	points
1	Maverick	104
2	Thunder	99
3	Rockets	116
4	Spurs	98
5	Pistons	99
6	Pelicans	105
7	Warriors	119
8	Blazers	113
9	Nuggets	100
10	Kings	123

Next, we apply the right-side extraction logic. We create a new dataset, `new_data`, which inherits all original fields and adds the derived variable containing the trailing characters.

```
/*create new dataset with extracted substring*/
data new_data;
set original_data;
last_three = substr(team, length(team)-2, 3);
run;

/*view new dataset*/
proc print data=new_data;
```

The key statement, `last_three = substr(team, length(team)-2, 3);`, operates on each observation independently. For example, for the string 'Warriors' (length 8), `length(team)-2` evaluates to 6. SUBSTR starts at the 6th character ('o') and reads 3 characters, successfully yielding 'ors'. This confirms the formula's effectiveness in accurately isolating the last three characters regardless of the initial string length.

The output below verifies the successful creation of the new variable, `last_three`, demonstrating the seamless application of this combined function approach for right-sided substring extraction in a practical SAS environment.

Obs	team	points	last_three
1	Maverick	104	ick
2	Thunder	99	der
3	Rockets	116	ets
4	Spurs	98	urs
5	Pistons	99	ons
6	Pelicans	105	ans
7	Warriors	119	ors
8	Blazers	113	ers
9	Nuggets	100	ets
10	Kings	123	ngs

Flexibility and Scaling: Adapting the Extraction Length

One of the greatest advantages of using the LENGTH function to calculate the starting position is the method's inherent flexibility. The process is not confined to extracting a fixed number of characters, such as three. By simply modifying the two numeric arguments provided to the SUBSTR function, you can easily adjust the extraction to capture any required number of trailing characters (N) from the right side of the source string.

If we decide to extract five characters (N=5) instead of three, we must re-evaluate the starting **Position** based on our generalized formula: `LENGTH(Source_String) - N + 1`. With N set to 5, the starting position simplifies to `LENGTH(team) - 5 + 1`, or `LENGTH(team) - 4`. This new calculated value replaces the second argument of SUBSTR, while the third argument must be explicitly set to 5.

We demonstrate this scalability by modifying our prior example to extract the last five characters from the `team` variable:

```
/*create new dataset extracting last five characters*/
data new_data;
set original_data;
last_five = substr(team, length(team)-4, 5);
run;

/*view new dataset*/
proc print data=new_data;
```

Consider the team name 'Thunder', which has 7 characters. The calculation for the starting position becomes `7 - 4 = 3`. Starting at the 3rd character ('u') and reading 5 characters results in 'under'. This dynamic calculation ensures that regardless of the string length, the correct trailing characters are always captured. The consistency and adaptability of this method make it highly valuable for automated data processing tasks where string lengths may vary significantly.

The subsequent output confirms that the new variable, `last_five`, accurately contains the last five characters from each team name, showcasing the effortless scalability of the SUBSTR and LENGTH function combination.

Obs	team	points	last_five
1	Maverick	104	erick
2	Thunder	99	under
3	Rockets	116	ckets
4	Spurs	98	Spurs
5	Pistons	99	stons
6	Pelicans	105	icans
7	Warriors	119	riors
8	Blazers	113	azers
9	Nuggets	100	ggets
10	Kings	123	Kings

Advanced Considerations and Alternative Approaches

While the combined SUBSTR and LENGTH function method is ideal for fixed-length extraction from the right, professional programming in SAS requires anticipating edge cases and knowing when alternative functions might be superior.

A critical consideration is handling **short strings**. If the desired number of characters to extract (N) is greater than the total length of the source string, the calculated starting position will be less than 1. When the SUBSTR function receives a non-positive starting position, it typically returns a blank or truncated value, depending on the specific SAS version and environment settings. To ensure data quality, especially in production environments, it is best practice to include conditional logic, such as an `IF` statement, to check if `LENGTH(Source_String)` is sufficient before performing the extraction. If the string is too short, you might choose to return the entire string or assign a missing value flag.

Furthermore, if the data requirement involves extracting a portion of a string based on a delimiter (e.g., finding the file extension after the last period) rather than a fixed number of characters, other SAS functions offer more powerful solutions. Functions like **SCAN**, **FIND**, **INDEX**, or the advanced pattern-matching capabilities of the **PRX functions** (Regular Expressions) should be considered. However, for sheer simplicity and computational efficiency when dealing with fixed-position or fixed-count extraction from the right, the SUBSTR and LENGTH function method remains the most elegant and readable solution.

Conclusion and Further Exploration

The challenge of extracting substrings from the right in SAS is elegantly solved by combining the foundational SUBSTR function with the dynamic measurement provided by the LENGTH function. This technique allows data analysts to precisely calculate the left-based starting position required to isolate any trailing sequence of characters. Whether you are parsing file names, isolating specific codes, or performing targeted data cleaning, this method is invaluable for ensuring data consistency.

We have demonstrated the crucial role of the formula `LENGTH(Source_String) - N + 1` in translating a right-side requirement into a viable left-side starting point for SUBSTR. The practical examples clearly illustrate the syntax and logic required to implement this solution within a DATA step, confirming its adaptability for extracting varying lengths of trailing substrings.

We strongly recommend practicing this method with diverse string inputs to fully appreciate its mechanics and limitations, particularly concerning nulls and short strings. For continued professional development, exploring the official SAS documentation for other advanced string manipulation techniques, such as those involving regular expressions, will further solidify your expertise in character data management.

Additional Resources

The following tutorials explain how to perform other common tasks in SAS: