Excel: Use MID Function for Variable Length Strings

Authored by

Mohammed looti

October 27, 2025

In the expansive realm of data manipulation and analysis, **Excel** remains an indispensable software application utilized by professionals across virtually all industries. A routine requirement in managing large datasets involves the precise extraction of specific portions of text, commonly referred to as **substrings**, from longer sequences of text, or **strings**. While Excel provides a robust suite of functions designed for text handling, dealing with data where the target substring's location or length is inconsistent--i.e., variable-length strings--presents a significant challenge. This comprehensive guide details an advanced, dynamic technique that synergistically combines the flexibility of the **MID function** with the positional accuracy of the **FIND function** to efficiently isolate these complex text segments.

The capacity to dynamically extract parts of a text string is fundamentally critical when processing unstructured or semi-structured data sources. Examples include web addresses (**URLs**), complex product identifiers, or detailed system log entries, where the data structure often lacks uniformity. Relying solely on static, traditional extraction methods proves inadequate when either the starting point or the total number of **characters** required for extraction cannot be fixed across all entries. By mastering the integration of **MID** and **FIND**, users can construct highly adaptive and resilient formulas capable of navigating these variations, thereby dramatically improving data cleansing and manipulation capabilities within **Excel**.

Understanding the MID Function: The Extractor

The **MID function** serves as Excel's core tool for extracting a designated quantity of **characters** from the internal section of a specified text **string**, beginning at a precise starting position defined by the user. Its structure is highly intuitive, requiring three fundamental arguments to operate successfully: `MID(text, start_num, num_chars)`.

`text`: This initial argument refers to the original source text **string** from which the desired characters are to be extracted.
`start_num`: This numerical input specifies the exact positional index of the very first **character** that the function should begin extracting. For instance, a value of 1 denotes the first character of the string, 2 denotes the second, and so on.
`num_chars`: This numerical argument dictates the total count of **characters** the **MID** function must extract starting from the `start_num` position.

Consider a simple example: if cell A1 holds the text "Advanced Excel Functions", the formula `=MID(A1, 10, 5)` will return "Excel". Here, 10 is the starting position (the 'E' in "Excel"), and 5 is the number of characters extracted. While this function is highly effective for extractions based on fixed starting positions and fixed lengths, the **MID function** encounters significant limitations when these parameters are inconsistent across a dataset. This is the precise point where integrating the **FIND function** transitions from being merely helpful to absolutely essential.

Leveraging the FIND Function for Dynamic Positions

To effectively counteract the inherent static nature of the `start_num` argument in the MID function, we employ the **FIND function**. The primary purpose of FIND is to dynamically locate a specific text **string** (the `find_text`) within a larger text string (the `within_text`) and return the numerical position where the search text begins. Its standard syntax is structured as `FIND(find_text, within_text, )`.

`find_text`: This is the specific sequence of **characters** you are actively searching for within the cell. It is important to note that FIND performs a case-sensitive match.
`within_text`: This argument specifies the text string containing the data you wish to analyze.
`start_num` (optional): This argument allows you to specify the **character** position from which the search should commence. If this is omitted, the search defaults to starting at the first character of the `within_text`.

The true utility of **FIND** stems from its capability to precisely identify the location of key markers or specific **delimiters** within an otherwise unstructured string. For instance, if cell A1 contains the URL "https://www.data-analysis.org", the formula `=FIND("www", A1)` would return 9, indicating that "www" starts at the ninth character. This returned numerical value is a crucial dynamic position, which is exactly what we need to feed into the `start_num` parameter of the **MID function**, thereby making it adaptable to variable-length text.

A critical distinction to remember is that the **FIND function** is strictly case-sensitive. If your data requires a case-insensitive search (meaning 'text' and 'TEXT' are treated identically), **Excel** offers the alternative `SEARCH` function. However, for precise detection of specific markers or standardized **delimiters** used in structured data extraction, the case-sensitive nature of FIND often provides the necessary accuracy for formula construction.

Combining MID and FIND for Variable Length Extraction

The greatest efficiency in extracting targeted **substrings** from variable-length text in **Excel** is achieved through the strategic combination of the **MID function** with the **FIND function**. This powerful pairing allows the formula to automatically calculate both the exact starting point (`start_num`) and the precise number of characters to extract (`num_chars`), based entirely on the dynamic positions of specified **delimiters** found within the text.

The fundamental concept involves utilizing FIND twice: once to locate the starting delimiter and again to locate the ending delimiter of the desired substring. These positional references are then mathematically manipulated and passed into the MID function's parameters. The general formula template used to extract text situated between a starting marker ("char1") and an ending marker ("char2") in a target cell (e.g., A2) is constructed as follows:

=MID(A2,FIND("char1",A2)+LEN("char1"),FIND("char2",A2,FIND("char1",A2)+LEN("char1"))-FIND("char1",A2)-LEN("char1"))

Let us meticulously analyze the two critical arguments (`start_num` and `num_chars`) within the context of the primary **MID function**:

Calculating `start_num`: `FIND("char1",A2)+LEN("char1")`

`FIND("char1",A2)`: This first step returns the numerical index of the starting character of "char1".
`+LEN("char1")`: We add the length of "char1" to this index. This crucial adjustment ensures that the MID function begins extraction immediately *after* the starting **delimiter**, effectively skipping it. Using the `LEN` function makes this robust for delimiters of any length.

Calculating `num_chars` (Length): `FIND("char2",A2,FIND("char1",A2)+LEN("char1"))-FIND("char1",A2)-LEN("char1")`

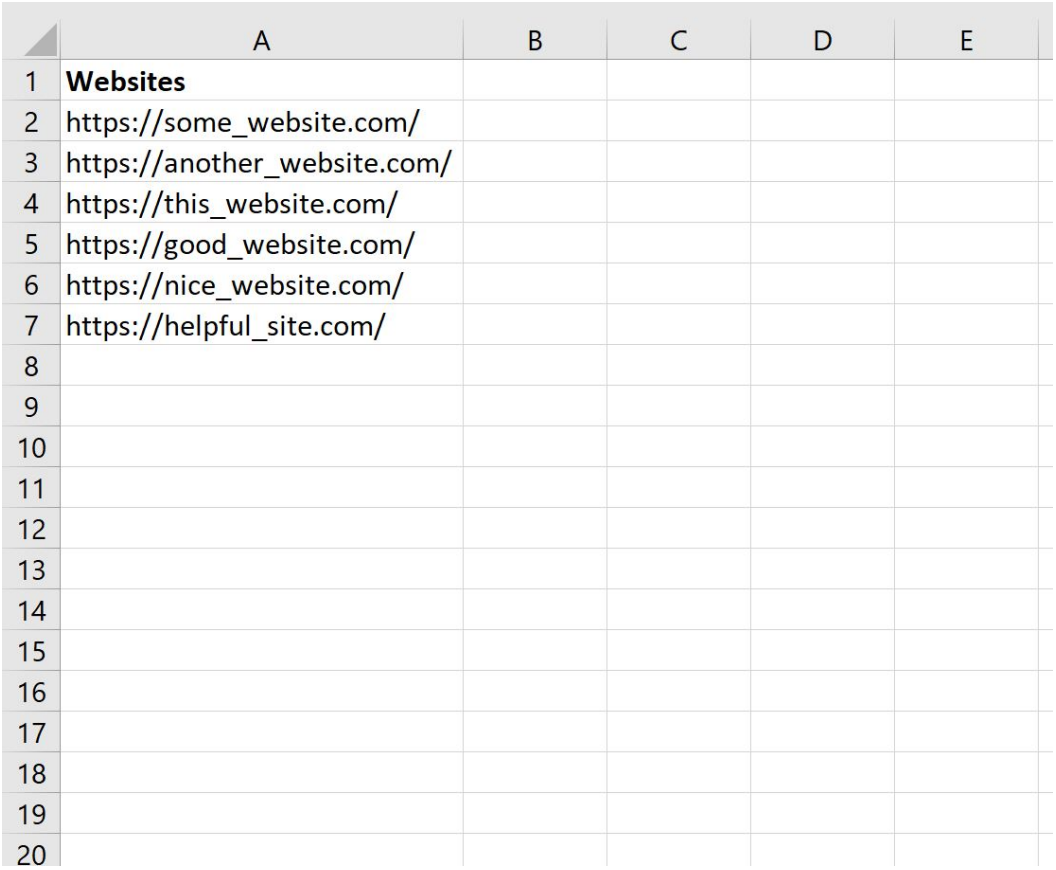
`FIND("char2",A2,FIND("char1",A2)+LEN("char1"))`: This finds the starting position of "char2". The key element is the third argument, which forces the search for "char2" to begin only *after* the starting position of the desired **substring** has been calculated. This prevents the formula from mistakenly finding an earlier occurrence of "char2".
`-FIND("char1",A2)-LEN("char1")`: By subtracting the adjusted starting position of the substring from the position of "char2", we are left with the exact, variable number of **characters** that lie between the end of "char1" and the beginning of "char2".

This carefully constructed formula successfully isolates every character in the **string** in cell **A2** that resides precisely between the specified **delimiters** "char1" and "char2". Because it relies on positional markers rather than fixed counts, this methodology is perfectly suited for managing highly heterogeneous datasets.

Practical Application: Extracting Website Names from URLs

To demonstrate the immense practical utility of this combined Excel formula, let us examine a typical data preparation task: isolating the core domain name from a column containing numerous complete web addresses (**URLs**). Imagine a scenario where your dataset contains full **URLs** in Column A, and your goal is to extract only the website name, excluding both the protocol prefix (e.g., "https://") and the top-level domain suffix (e.g., ".com").

Suppose we are working with a list of website **URLs** similar to the example visualized below:



	A	B	C	D	E
1	**Websites**				
2	https://some_website.com/				
3	https://another_website.com/				
4	https://this_website.com/				
5	https://good_website.com/				
6	https://nice_website.com/				
7	https://helpful_site.com/				
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Our specific task is to extract the segment of text located between the double forward slashes (`//`) and the subsequent top-level domain marker (`.com`). This requires a dynamic solution because the length of the domain name varies significantly across the list. This scenario is ideally suited for the dynamic **MID** and **FIND** combination, ensuring that the extraction adapts seamlessly to the content of each individual cell.

We apply the generic structure to cell A2, defining `//` as our starting delimiter and `.com` as our ending delimiter. The resulting, highly specific formula is:

=MID(A2,FIND("//",A2)+2,FIND(".com",A2,FIND("//",A2)+2)-FIND("//",A2)-2)

A detailed breakdown of this implementation reveals how the parameters are dynamically calculated:

Determining `start_num`: `FIND("//",A2)+2`

The **FIND function** first locates the position of `//`. We then add 2 to this result because `//` consists of two characters. This addition ensures the extraction starts immediately after the **delimiter**, isolating the beginning of the domain name.

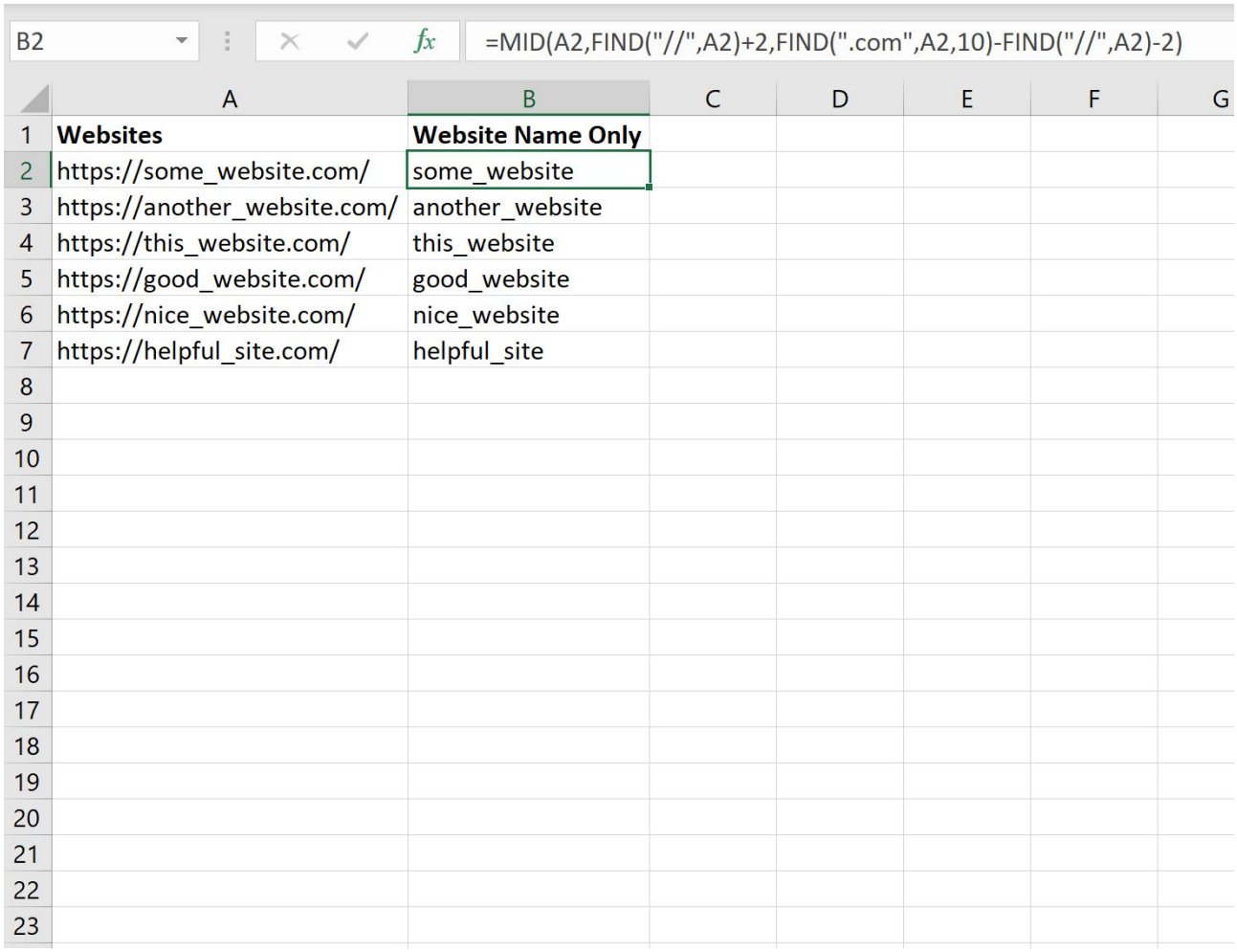
Determining `num_chars`: `FIND(".com",A2,FIND("//",A2)+2)-FIND("//",A2)-2`

The internal FIND function locates the position of `.com`, but only after skipping the `//` marker found in the previous step. From this end position, we subtract the adjusted start position (`FIND("//",A2)+2`). This subtraction yields the precise number of characters that constitute the website name **substring**.

Step-by-Step Implementation and Reviewing Results

To execute this operation within your **Excel** worksheet, simply enter the formula precisely as written into the target output cell (e.g., cell B2, assuming **URLs** begin in A2). Once the formula is correctly entered for the first URL, you can efficiently propagate the formula down the column by utilizing the fill handle. Excel's automatic relative referencing ensures that A2 adjusts to A3, A4, and subsequent cells automatically, processing the entire list.

The resulting output after applying this dynamic formula is illustrated in the following screenshot:



	A	B	C	D	E	F	G
1	**Websites**	**Website Name Only**					
2	https://some_website.com/	some_website					
3	https://another_website.com/	another_website					
4	https://this_website.com/	this_website					
5	https://good_website.com/	good_website					
6	https://nice_website.com/	nice_website					
7	https://helpful_site.com/	helpful_site					
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							

As clearly demonstrated in Column B, the combined formula successfully extracts only the core website name--such as "google", "youtube", "microsoft", and "exceljet"--regardless of the varying lengths and starting positions of these names within the original strings. This success highlights the central advantage of this technique: its dynamic adaptability. It does not rely on a fixed length or position, making it a robust solution for processing diverse and unstructured datasets.

This adaptability is why mastering the combination of **MID** and **FIND** is essential for advanced data cleaning. Without the positional intelligence provided by the FIND function, the MID function alone would be fundamentally incapable of performing such a precise and efficient extraction across a list of variable-length strings.

Advanced Considerations and Error Handling

While the MID and FIND technique is exceptionally powerful, experienced data professionals must account for potential edge cases, particularly error propagation. What happens, for instance, if one of the specified **delimiters** (e.g., `//` or `.com`) is missing from a particular cell's string? In such an event, the **FIND function** will fail to locate the marker and return the standard Excel `#VALUE!` error. This error will subsequently propagate through the entire formula, rendering the final cell output unusable.

To handle these errors gracefully, it is strongly recommended to embed the core formula within the `IFERROR` function. This function allows the user to specify an alternative action or message if the main formula encounters an error. For example, to display "Delimiter Not Found" instead of `#VALUE!`, the complete formula structure becomes: `=IFERROR(MID(A2,FIND("//",A2)+2,FIND(".com",A2,FIND("//",A2)+2)-FIND("//",A2)-2), "Delimiter Not Found")`.

It is also worth noting that users of modern Excel versions (e.g., Microsoft 365) now have access to newer, more streamlined functions such as `TEXTAFTER`, `TEXTBEFORE`, and `TEXTSPLIT`, which simplify delimiter-based extraction. However, the **MID** and **FIND** combination remains a foundational, universally compatible technique across all versions of Excel, making it a non-negotiable skill for anyone involved in data manipulation.

Mastering this technique empowers you to confidently and efficiently tackle complex text manipulation challenges, transforming raw, inconsistent data into highly structured and actionable insights. By deeply understanding the complementary strengths of the MID and FIND functions, you unlock superior flexibility and precision in all your spreadsheet operations.

Additional Resources for Text Manipulation

To further refine your Excel text manipulation expertise and explore related functions, consider

delving into the following tutorials which cover other essential techniques:

Excel: A Formula for MID From Right

Excel: Using LEFT and RIGHT Functions for Text Extraction

Excel: Understanding the SEARCH Function (Case-Insensitive FIND)