Unlock the Power of Bash Wildcards for Advanced Linux Usage

By Linux Code / October 28, 2023

Bash wildcards are special characters that enable powerful and flexible pattern matching for searching and transforming files and data in Linux. Getting a deep understanding of how to leverage bash wildcards will allow you to search files faster, validate input better, and script more efficiently on the command line.

In this comprehensive guide, you'll learn:

- What are bash wildcards and why are they useful?
- How to harness asterisk, question mark, and square bracket wildcards
- Advanced use cases like batch file management and input validation
- Best practices for combining wildcards and crafting patterns

So if you want to unchain the full potential of bash wildcards, read on!

What Are Bash Wildcards and Why Are They Helpful?

Bash wildcards are symbols like the asterisk (*), question mark (?), and square brackets ([ ]) that allow you to define flexible matching patterns.

Here's a quick overview:

- * matches zero or more characters
- ? matches exactly one character
- [ ] matches a range or set of characters

With over 5 billion Bash downloads and installations, Bash is one of the most popular shells used on Linux and UNIX-like systems today. By using wildcards in your Bash commands and scripts, you gain the ability to:

- Search for files or data using flexible naming patterns
- Filter and transform groups of files/text efficiently
- Validate input by defining allowed character sets
- Automate tasks like batch file management

In short, bash wildcards supercharge your Bash skills by enabling powerful pattern matching capabilities. Understanding how to leverage them will make you a faster and more productive Linux user!

The All-Powerful Asterisk Wildcard

The * asterisk wildcard is one of the most frequently used and helpful symbols. It allows you to match any string of characters, which provides a huge amount of flexibility for searching files and data.

Let's explore some common use cases and examples of harnessing the asterisk wildcard:

1. Finding Files Starting with Specific Characters

To match all files starting with the letter 'a', use:

```
ls a*
```

This will print all filenames beginning with 'a' in the current directory. Instead of 'a', you can use any starting character(s) you want to match.

For example, to find all Python files beginning with 'py', you could use:

```
ls py*
```

And to find all files starting with 'doc', 'spec', or 'readme':

```
ls [drs]*
```

So the * allows matching the start of filenames in a flexible way.

2. Searching for Files by Extension

To find all text files with a '.txt' extension, use:

```
ls *.txt
```

The * here essentially means "match any characters before '.txt'". This approach works for searching files with any extension.

For example, you can find JSON files with:

```
ls *.json
```

Or PHP files with:

```
ls *.php
```

This provides an easy way to search for files of a certain type.

3. Partial Filename Matching

You can also use * wildcards to match part of a filename. For example, to find all files containing 'test' anywhere in the name:

```
ls *test*
```

This will match filenames like 'mytest.py', 'anothertest.sh', 'foo-test-bar.txt' etc.

You can also use * on both sides of a partial name if you don't know the full pattern. For example, to find files related to 'projectX':

```
ls *projectX*
```

4. Deleting Files by Pattern Matching

Asterisks come in very handy for deleting groups of files matching a pattern.