How to Use Wildcards

A wildcard is a character that can be used as a substitute for any of a class of characters in a search, thereby greatly increasing the flexibility and efficiency of searches.

Wildcards are commonly used in shell commands in Linux and other Unix-like operating systems. A shell is a program that provides a text-only user interface and whose main function is to execute commands typed in by users and display their results.

Wildcards are also used in regular expressions and programming languages. Regular expressions are a pattern matching system that uses strings (i.e., sequences of characters) constructed according to pre-defined syntax rules to find desired strings in text.

The term wildcard or wild card was originally used in card games to describe a card that can be assigned any value that its holder desires. However, its usage has spread so that it is now used to describe an unknown or unpredictable factor in a variety of fields.

Star Wildcard

Three types of wildcards are used with Linux commands. The most frequently employed and usually the most useful is the star wildcard, which is the same as an asterisk (*). The star wildcard has the broadest meaning of any of the wildcards, as it can represent zero characters, all single characters or any string.

As an example, the file command provides information about any filesystem object (i.e., file, directory or link) that is provided to it as an argument (i.e., input). Because the star wildcard represents every string, it can be used as the argument for file to return information about every object in the specified directory. Thus, the following would display information about every object in the current directory (i.e., the directory in which the user is currently working):

```
file *
```

If there are no matches, an error message is returned, such as *: can't stat `*' (No such file or directory).. In the case of this example, the only way that there would be no matches is if the directory were empty.

Wildcards can be combined with other characters to represent parts of strings. For example, to represent any filesystem object that has a .jpg filename extension, *.jpg would be used. Likewise, a* would represent all objects that begin with a lower case (i.e., small) letter a.

As another example, the following would tell the ls command (which is used to list files) to provide the names of all files in the current directory that have an .html or a .txt extension:

```
ls *.html *.txt
```

Likewise, the following would tell the rm command (which is used to remove files and directories) to delete all files in the current directory that have the string xxx in their name:

```
rm *xxx*
```

Question Mark Wildcard

The question mark (?) is used as a wildcard character in shell commands to represent exactly one character, which can be any single character. Thus, two question marks in succession would represent any two characters in succession, and three question marks in succession would represent any string consisting of three characters.

Thus, for example, the following would return data on all objects in the current directory whose names, inclusive of any extensions, are exactly three characters in length:

```
file ???
```

And the following would provide data on all objects whose names are one, two or three characters in length:

```
file ? ?? ???
```

As is the case with the star wildcard, the question mark wildcard can be used in combination with other characters. For example, the following would provide information about all objects in the current directory that begin with the letter a and are five characters in length:

```
file a????
```

The question mark wildcard can also be used in combination with other wildcards when separated by some other character. For example, the following would return a list of all files in the current directory that have a three-character filename extension:

```
ls *.???
```

Square Brackets Wildcard

The third type of wildcard in shell commands is a pair of square brackets, which can represent any of the characters enclosed in the brackets. Thus, for example, the following would provide information about all objects in the current directory that have an x, y and/or z in them:

```
file *[xyz]*
```

And the following would list all files that had an extension that begins with x, y or z:

```
ls *.[xyz]*
```

The same results can be achieved by merely using the star and question mark wildcards. However, it is clearly more efficient to use the bracket wildcard.

When a hyphen is used between two characters in the square brackets wildcard, it indicates a range inclusive of those two characters. For example, the following would provide information about all of the objects in the current directory that begin with any letter from a through f:

```
file [a-f]*
```

And the following would provide information about every object in the current directory whose name includes at least one numeral:

```
file *[0-9]*
```

The use of the square brackets to indicate a range can be combined with its use to indicate a list. Thus, for example, the following would provide information about all filesystem objects whose names begin with any letter from a through c or begin with s or t:

```
file [a-cst]*
```

Likewise, multiple sets of ranges can be specified. Thus, for instance, the following would return information about all objects whose names begin with the first three or the final three lower case letters of the alphabet:

```
file [a-cx-z]*
```

Sometimes it can be useful to have a succession of square bracket wildcards. For example, the following would display all filenames in the current directory that consist of jones followed by a three-digit number:

```
ls jones[0-9][0-9][0-9]
```

Other Wild Cards

\ (backslash) = is used as an "escape" character, i.e. to protect a subsequent special character. Thus, "\ \" searches for a backslash. Note you may need to use quotation marks and backslash(es).

^ (caret) = means "the beginning of the line". So "^a" means find a line starting with an "a".

$ (dollar sign) = means "the end of the line". So "a$" means find a line ending with an "a".

For example, this command searches the file myfile for lines starting with an "s" and ending with an "n", and prints them to the standard output (screen):

```
cat myfile | grep '^s.*n$'
```