

Example of a (rather complicated) `grep` command

```
grep '^a[a-z]*b *$' foo
```

The apostrophes at each end of the regular expression are not part of it; they merely delimit it and protect its content from being interpreted by the Unix shell. The command as a whole means “Search the file **foo** and show every line that begins with **a** and continues with zero or more other letters, eventually arriving at a **b**, after which there is nothing except perhaps some spaces.” (Many useful `grep` commands are a lot simpler than this!)

Table of regular expression meanings

This table displays a representative sample of regular expressions of the sort that `grep` accepts, together with their meanings. More complicated ones can be made by joining these together.

Expression	Strings that match
<code>b</code>	b
<code>^b</code>	b at the beginning of a line
<code>b^</code>	b followed by <code>^</code>
<code>b\$</code>	b at the end of a line
<code>\$b</code>	<code>\$</code> followed by b
<code>bc</code>	b followed by c
<code>[bc]</code>	either b or c
<code>[bcd]</code>	b or c or d
<code>[0-9]</code>	any digit between 0 and 9 (in the ASCII order)
<code>[j-t]</code>	any lower-case letter between j and t (in the ASCII order)
<code>[A-M]</code>	any capital letter in the first half of the alphabet
<code>[^bcd]</code>	anything that is NOT b or c or d
<code>b*</code>	a sequence of consecutive bs (zero or more of them)
<code>b*c</code>	a sequence of consecutive bs (zero or more of them) followed by c
<code>bc*</code>	b followed by a sequence of consecutive cs (zero or more of them)
<code>[bc]*</code>	a sequence of any length that is all bs and/or cs , e.g. <code>bbcccbcbc</code>
<code>[^bc]*</code>	a sequence of any length that does NOT contain b or c
<code>.</code>	any character (between ASCII characters octal 040 and octal 176)
<code>.*</code>	any sequence of characters (zero or more of them)

An immediately preceding backslash (`\`) turns off the ‘magic’ of any of the special characters if you need to refer to them literally; so the following regular expressions are fully predictable:

Expression	Strings that match
<code>\.</code>	the period character (<code>.</code>)
<code>\[</code>	the left square bracket character (<code>[</code>)
<code>\]</code>	the right square bracket character (<code>]</code>)
<code>*</code>	the asterisk character (<code>*</code>)
<code>\^</code>	the caret or up-arrow character (<code>^</code>)
<code>\\$</code>	the dollar sign (<code>\$</code>)
<code>\\</code>	the backslash character (<code>\</code>)