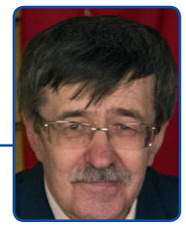
By Mikhail Ivanyushin



Multi & Single Line Mystery

These words — single-line, multiline — both are similar in spelling, but its are written in English exactly: one word with a hyphen, the second one word.

The first assumption of the presence of two modes — single-line and multiline — used or first or second. But the modes are not mutually exclusive. They mean two completely different things, and each of its is on or off. Single-line mode **(?s)** changes the meaning of the **.** dot meta-character, while multiline mode **(?m)** changes the meaning of the **^** and **$** anchors.

Single-line mode

When single-line mode is off **(?-s)**, it is the default state of GREP requests processing: operator **.+** selects all characters from the current position to the end of the paragraph, but the end of the paragraph is not selected. Analogy — each paragraph in the story is a separate line.

When single-line mode is active **(?s)**, the line feed character **\r** is also included in the selected area. That is, all text from the cursor point to the end of the story will be selected. And if the cursor is not in the text, and the frame is selected, the whole story will be selected. It is acceptable to say that the story turns into one line.

If there is no **dot** meta-character in the GREP query, it does not matter which single-line mode option is currently set.

Multiline mode

When multiline mode is enabled **(?m)**, this is the default state of GREP requests processing, **^** and **$** anchors define the beginning and end of the paragraph. The search with the use of meta-characters **^** and **$** runs in the space of the paragraph in which the cursor is positioned. For GREP parser the story has as many lines as there are paragraphs.

When multiline mode is off **(?-m)**, the meta-characters **^** and **$** are interpreted as markers of the beginning and the end of the story, **\A** and **\Z** respectively. The search is performed in the space of the entire story. The whole story is one line.

If there are no **^** and **$** meta-characters in the GREP query, it does not matter which multiline mode option is currently set.

For understanding

Let's there are two paragraphs

Text¶

Word¶

Any of the requests **.+** or **(?-s).+** will find the first paragraph of the **Text**, and then paragraph of the **Word**. The samples will contain only printed symbols, character **\r** not be selected.

And if we have such GREP query **(?s).+** that it will see this text so as a single line: **Text\rWord\r** and this query will select all the characters in this line.

Strings **^book**, and **(?m)^book** determines the search **book** at the beginning of any line.

Strings **list$** and **(?m)list$** specify the search **list** at the end of any line.

The string **(?-m)^book** indicates the search **book** at the beginning of the story. The string **(?-m)list$** denotes the search **list** at the end of the story.

Obviously, these two last queries can be rewritten differently, making its more understandable: **\Abook** and the **list\Z**.

Two modes in one query

If the GREP query text contains **dot** and at least one of the two **^** or **$** characters, it is possible to define how these modes will interact. Obviously, there are four possible combinations

Usage of each mode	The scope of the meta-characters **^** and **$**	Characters that **dot** can be
(?s)(?-m)	story	any character
(?-s)(?-m)	story	any printed character
(?m)(?-s)	line	any printed character
(?sm)	line	any character

Let's say there is the text:

1, 2, 3,¶

it's a Christmas tree!¶

3, 2, 1,¶

Christmas is fun!¶

and try such search GREP queries **.{10}$** and **^.{12}**.

The search always starts at the current cursor position.

(?s)(?-m).{10}$ The last ten characters of the story will be selected. Since the **dot** works like any character, if the last character of the story is a line feed, it will also be included in the selection.

(?-s)(?-m).{10}$ *If the last character of the story is not a line feed*, the last ten characters of the story will be selected. And since the **dot** works as a printed character, if the last character of the story is a line feed, then nothing will be selected, because among the last ten characters there is a line feed, but **\r** is not included in the set of characters covered by the **dot**.

(?m)(?-s).{10}$ The last ten characters of each line will be selected. Line feed is not included in the sample.

(?sm).{10}$ If to place the cursor at the beginning of the text, the last ten characters of second and fourth lines will be selected. But if to put the cursor before **tree** in the second line, the query will highlight the exclamation mark and **\r** in second line and eight symbols of third line.

Why is it so? The query looks for the end of the paragraph; finds its at the end of the line where insertion point stands; estimates the number of characters in front of this end of a paragraph not only on the subject of whether there is a right number, but where is the current insertion point. In this case, the insertion point falls into these ten characters, so this line feed is not suitable. The query looks for the next one, also checks ten characters in front of founded line feed, whether the current position of the cursor falls into them. If not, they selected.

In this example we put the cursor before word **tree**. But as soon as the cursor after exclamation mark, it falls into the space of ten characters before the line feed, and the query looks for the next line feed in the text.

(?s)(?-m)^.{12} Search in the story, all characters in the text are searched. The **^** character is interpreted not only as the beginning of the story, but also as **\r**, so this query allocates first twelve characters from the beginning of the story, and then next sequences of twelve characters starting with **\r**.

(?-s)(?-m)^.{12} Find of any printed characters in the line. State "any printed character" defines that a line feed character will not be included in the selection. This query may select in paragraph only first twelve print characters from insertion point.

(?m)(?-s)^.{12} Search in a line, any printed character. Twelve text characters are searched at the beginning of each paragraph. If the paragraph is shorter, it is skipped.

(?sm)^.{12} Search in a line, any character. Twelve characters are selected at the beginning of each paragraph. If the paragraph is shorter, a line feed and several characters of the next paragraph will be included in the selection.

Clarification is required

At the beginning of this article was said and it is taken from the documentation what the multiline mode state is enabled by default. Is it all information about this mode?

Here is the task: find the first spaces in the paragraphs, try this GREP query for its solution: **^(.+?)\K\h**. In InDesign 2018 & 2019 the query does not solve this task, because it jumps over the line (In InDesign CS6 the query works right).* I don't know why it so. But if we change this query: **(?m)^(.+?)\K\h** then it will work right.

What is it? After all, if multiline mode is enabled by default, the queries **^(.+?)\K\h** and **(?m)^(.+?)\K\h** must be executed in the same way. However this does not happen.

My guess is that when the command **(?m)** appears in GREP query, the parser additionally changes the interpretation of the **^** anchor. At the time of the query processing meta-character **^** becomes not

* https://youtu.be/FEJpn3Nolw0 — it is link to video where is shown how differently processed query **^(.+?)\K\h** in CS6, CC2018 and CC2019.

only a marker of the beginning of the story/paragraph, but also corresponds to **\r**.
Here's how it looks in the implementation by other requests: query **\r(.+?)\K\h** goes through all the lines starting with the second, and the query **(^|\r)(.+?)\K\h** works the same way as **(?m)^(.+?)\K\h**. Queries **(?s)^(.+?)\K\h** and **(?-s)^(.+?)\K\h** works right the same way precisely. I think it is because for the duration of the query the inclusion operators of any of these modes will determine that the anchor **^** should be treated as **\r** too.
So by default multiline mode is enabled and single-line mode is disabled, it is right. But a lot of GREP tasks led to the reasonable assumption that any command to work with these modes **(?m)**, **(?-m)**, **(?s)**, **(?-s)** forces the parser to change the usage of the meta-character **^**. For the duration of the query with these commands anchor **^** is not only the beginning of the story and paragraph, but also the equivalent of **\r**. In queries with commands of these modes the operator **^** works as **(^|\r)**.

One more example of usage

Replace flush spaces with the tabs

In the text the positions of the list are separated by flush spaces.
A) text, numbers **=FS=** D) text, numbers
B) text, numbers **=FS=** E) text, numbers
C) text, numbers **=FS=** F) text, numbers
=FS= is here as symbol description of flush space it is necessary to replace these flush spaces with the tabs.
Query **^\u\).+\K~f** will search through the string.
And an explicit indication that multiline mode is enabled or single-line mode is disabled (although these states are already by default) determines that during query processing the story beginning/paragraph start symbol **^** will be applied to the line feed too.
The result is a query **(?m)^\u\).+\K~f** or **(?-s)^\u\).+\K~f** will go through each line.

Conclusion

Of course, it is possible to give a dozen more examples, but it is important to stop in time. I think it is wrote enough here about multiline and single-line modes and features of their usage. Feel free to ask if you have any questions, email is below.

Mikhail Ivanyushin is designer, scripter (dotextok.ru, adobeindesign.ru), writer, blogger, sometimes interviewer (2008, 2014), occasionally lecturer.
And always InDesign-Diver.
m.ivanyushin@gmail.com