**Applications of Regular Expressions:**

Regular expressions are a powerful means for pattern matching and string parsing that can be applied in so many instances. With this incredible tool you can:

* Validate text input
* Search (and replace) text within a file
* Batch rename files
* Undertake incredibly powerful searches for files
* Interact with servers like Apache
* Test for patterns within strings (in log files)
* And so much more

Mostly used for monitoring process log entries in servers.

**Links on how Regex is used in companies like HP**

<https://community.hpe.com/t5/General/Regex/m-p/5097445#M21753>

<https://community.hpe.com/t5/General/Regular-expression-matching/td-p/3313170#.YRNW19_hU2w>

1. **Use Case:**

Consider a Java process being monitored by our server monitoring agent. Every time the process is restarted, the process arguments might change. During this scenario, the process will be marked as Down and it won’t be changed to the Up status even after the process being restarted successfully.

Problem Statement:

The user has to delete the existing Java process and add the same process again with the newly discovered arguments in order to mark it up for monitoring.

Solution:

For the above case, let's assume the initial argument for the Java process was like:

/usr/bin/java -Dosgi.requiredJavaVersion=1.8 -start\_time 1504249532

On restarting the process, it gets altered to the following:

/usr/bin/java -Dosgi.requiredJavaVersion=1.8 -start\_time 124830290

A sample Regex that can be used for this case is as follows:

(.\*)start\_time\s?([0-9]{1,10})

where,

(.\*) - matches anything

\s - matches white space

0-9 - a single character in the range between 0 and 9

1,10 - matches between 1 and 10 as many times as possible

Once RegEx is enabled, the Linux monitoring agent will match the running process argument with the configured RegEx and determine the status of the process.

1. **Searching Paths for Log Files for Log File Monitoring**

UNIX and Windows operating systems treat the case ("N" and "n") of file names in incompatible ways. Windows operating systems are case insenstive which means that when a file is being searched, its case is ignored. UNIX operating systems are case sensitive which means that the case of a name is significant at all times. To avoid log file errors when using regular expressions to search for path names on UNIX operating systems, use markers.

1. **Matching Comma-Separated Values**

The following is an example of log file entries that are comma-separated strings of digits and letters:

new,open,changed,12,alerts

new,open,changed,13,alerts

new,open,changed,13,alerts

new,open,changed,14,alerts

A regular expression can be written to match on log file entries that are comma-separated strings of digits and letters.

1. **Matching Space Separated Values**

The following is an example of log file entries that are a sequence of strings and digits separated by spaces:

requests 12 succeeded 12 failed

requests 12 succeeded 12 failed

requests 11 succeeded 11 failed

requests 12 succeeded 12 failed

requests 10 succeeded 10 failed

The following is a regular expression to match on log file entries that are a sequence of strings and digits separated by spaces.

1. **Matching and Retaining the Numbers in a Line of Text and Numbers**

The following is an example of log file entries that are comma separated strings that combine digits and letters:

request handle number 12.56, series 17.5, sequence reported 97.45, 15.95 and 19.51

request handle number 15.96, series 27.5, sequence reported 107.45, 25.95 and 19.52

request handle number 11.06, series 36.5, system codes 9.45, 35.95 and 19.53

log reference number 12.30, series 17.5, channel reset values 100.45, 45.95 and 19.54

A regular expression can be written to match on log file entries that are comma-separated strings that combine digits and letters and retain the decimal numeric data:

1. **Matching Integers and Floating-point Numbers (Positive or Negative)**

The following is an example of log file entries that are a sequence of integers and floating point numbers that may be negative or positive:

12.1987 -71 -199.1 145 -1.00716

13.2987 -72 -199.2 245 -1.00726

14.3987 -73 -199.3 345 -1.00736

15.4987 -74 -199.4 445 -1.00746

A regular expression can be written to match on log file entries that are a sequence of 5 integers and floating point numbers that may be negative or positive. The numbers in each entry must be separated by one or more spaces.

1. **Matching Date and Time-Coded Log Entries**

Many log files include some form of date and time data with each entry. The following is an example of log file entries that include date and time information together with string data separated by commas:

20/04/2003 14:29:22,ERROR,request failed

20/04/2003 14:31:09,INFO,system check complete

20/04/2003 14:35:46,INFO,new record created

A regular expression can be written to match on log file entries that are date- and time-coded followed by comma-separated strings of letters and digits.