REGULAR EXPRESSIONS

Multi-threaded NewsWatcher

Contents

Go back to the page on filtering.
See the page on creating and editing filters.

REGULAR EXPRESSIONS

What are regular expressions?

Regular expressions are a system for matching patterns in text data, which are widely used in UNIX systems, and occasionally on personal computers as well. They provide a very powerful, but also rather obtuse, set of tools for finding particular words or combinations of characters in strings.

On first reading, this all seems particularly complicated and not of much use over and above the standard string matching provided in the Edit Filters dialog (Word matching, for example). In actual fact, in these cases NewsWatcher converts your string matching criteria into a regular expression when applying filters to articles.

However, you can use some of the simpler matching criteria with ease (some examples are suggested below), and gradually build up the complexity of the regular expressions that you use.

One point to note is that regular expressions are *not* wildcards. The regular expression '`c*t`' does not mean 'match "cat", "cot"' etc. In this case, it means 'match zero or more 'c' characters followed by a t', so it would match 't', 'ct', 'cccct' etc.

Information sources

The information here is an amalgamation of the documentation of regular expressions in the Metrowerks CodeWarrior IDE, and of a chapter in the book UNIX Power Tools (Peek, O'Reilly & Loukides). Online information (often the man pages for UNIX utilities) is available by using one of the search engines (e.g. InfoSeek) to search for 'regular expressions'.

Regex Basics

Matching simple expressions

Most characters match themselves. The only exceptions are called *special characters*:

- asterisk (`*`),
- plus sign (`+`),
- question mark (`?`),
- backslash (`\`),
- period (`.`),
- caret (`^`),
- square brackets (`[` and `]`),
- dollar sign (`$`),
- ampersand (`&`).
- or sign (`|`).

To match a special character, precede it with a backslash, like this `\*`.

For example,

This expression...	matches this...	but not this...
`a`	a	b
`\.\*`	.*	dog
`100`	100	ABCDEFG

Matching any character

A period (.) matches any character except a newline character.

This expression...	matches this...	but not this...
`.art`	dart	art
	cart	hurt
	tart	dark

Repeating expressions

You can repeat expressions with an asterisk or plus sign.

A regular expression followed by an asterisk (`*`) matches *zero or more occurrences* of the regular expression. If there is any choice, the first matching string in a line is used.

A regular expression followed by a plus sign (`+`) matches *one or more occurrences* of the one-character regular expression. If there is any choice, the first matching string in a line is used.

A regular expression followed by a question mark (`?`) matches *zero or one occurrence* of the one-character regular expression.

For example:

This expression...	matches this...	but not this...

a+b	ab	b
	aaab	baa
a*b	b	daa
	ab	
	aaab	
.*cat	cat	dog
	9393cat	
	the old cat	
	c7sb@#puiercat	
a[n]? h	a herb	ann hat
	an herb	

So to match any series of zero or more characters, use "`.*`". On its own this isn't much use, but in the middle of a longer regular expression, it can be.

Grouping expressions

If an expression is enclosed in parentheses (`(` and `)`), the editor treats it as one expression and applies any asterisk (`*`) or plus (`+`) to the whole expression.

For example

This expression...	matches this...	but not this...
(ab)*c	abc	ababab
	abababababc	ababd
(.a)+b	xab	b
	ra5afab	aagb

Choosing one character from many

A string of characters enclosed in square brackets (`[]`) matches any one character in that string. If the first character in the brackets is a caret (`^`), it matches any character *except* those in the string. For example, `[abc]` matches a, b, or c, but not x, y, or z. However, `[^abc]` matches x, y, or z, but not a, b, or c.

A minus sign (-) within square brackets indicates a range of consecutive ASCII characters. For example, `[0-9]` is the same as `[0123456789]`. The minus sign loses its special meaning if it's the first (after an initial ^, if any) or last character in the string.

If a right square bracket is immediately after a left square bracket, it does not terminate the string but is considered to be one of the characters to match. If any special character, such as backslash (`\`), asterisk (`*`), or plus sign (`+`), is immediately after the left square bracket, it doesn't have its special meaning and is considered to be one of the characters to match.

This expression...	matches this...	but not this...
[aeiou][0-9]	a6	ex
	i3	9a
	u2	$6

`[^cfl]og`	dog	cog
	bog	fog
`END[.]`	END.	END;
		END DO
		ENDIAN

Matching the beginning or end of a line

You can specify that a regular expression match only the beginning or end of the line. In NewsWatcher, a line is the whole field that is being matched, for example the author or subject fields. These are called anchor characters:

If a caret (`^`) is at the beginning of the entire regular expression, it matches the beginning of a line.

If a dollar sign (`$`) is at the end of the entire regular expression, it matches the end of a line.

If an entire regular expression is enclosed by a caret and dollar sign (`^like this$`), it matches an entire line.

This expression...	matches this...	but not this...
`^(the cat).+`	the cat runs	see the cat run
`.+(the cat)$`	watch the cat	the cat eats

So, to match all strings containing just one characters, use "`^.$`".

Regex Extensions

Matching words

You can specify that a regular expression match parts of words with `\<`; (match the start of a word) and `\>`; (match the end of a word). An expression like "`\<app`" will match "apple" and "application", while "`ing\>`" will match all words ending in -ing. To match a whole word, using an expression like "`\<this\>`".

NewsWatcher provides facilities for doing words matches (which use these expressions internally), but if you want more flexibility, these come in useful. For example, you might want

```
M.*\<Excel\>
```

to match

MS Excel, Microsoft Excel, Microsquish Excel etc. To remind you, the `.*` mean 'zero or more (*) of any character (.)'.

Alternatives

You can define an expression like `(cash|money)` to match strings which contain either the word 'cash', or the word 'money', or both. Note that the parentheses around the expression are required.

Regex Examples

Examples

Here are some sample regular expressions that I've found useful.

Kill if 'subject' contains the reg. exp. "`(cash|money)`"
This kills articles with 'cash' or 'money' in the subject. This should be a case-insensitive match.

Kill if 'subject' contains the reg. exp. "`^\[?F.?S.?`"
This kills 'For Sale' articles, which have a subject line that starts ('`^`') with either FS, F.S., [FS] or [F.S.]. Here the '`[`' needs to be escaped to '`\[`', and the '`?`' means 'match zero or one instance of'.

Kill if 'subject' contains the reg. exp. "`[[$%|_\*!][[$%|_\*!][[$%|_\*!]`"
This is a nifty one that kills those posts with subjects like "$$$blah blah" or "_______this..." which are almost surely not worth reading. The regular expression reads like this. It repeats the range of characters `[[$%|_\*!]` three times, meaning that any of the characters in the [] will be matched. ([is normally interpreted as starting a group like this *unless* it is the first character after a [, hence its position here.) This grouping is then repeated three times, to match subjects like $_* or *** or !_!. You could prepend a ^ to force the match a the beginning of the line.

Kill if 'Xref' contains the reg. exp. "`[^ ]+ [^ ]+ [^ ]+ [^ ]+`"
This kills articles which have been cross-posted to four or more groups, and works by looking for runs of non-space characters (the `[^ ]`) separated by spaces.

Hilite if 'subject' contains the reg. exp. "`News ?Watcher` (ignore case)"
This will match "newswatcher", "NewsWatcher", "News Watcher", "news Watcher" and so on. The '?' means match zero or one space.

Hilite if 'subject' contains the reg. exp. "`Kaleid[aeio]scope`"
This will match "Kaleidoscope", as well as all the misspellings that are common, the `[]` meaning match any of the alternatives within the square brackets.

Hilite if 'subject' contains the reg. exp. "`^\[?A[Nn][Nn]`"
This is useful for catching announcement posts, where the subject line starts with [ANN] or Ann or [Ann. The first "`^`" forces a match at the beginning of the line. Then it looks for zero or one (the meaning of the "`?`") "[" characters, but since this is a reserved character, it has to be escaped to "`\[`". Then we look for a "A", followed by either "N" or "n" and then one or more "N" or "n" characters.

Go back to the page on filtering.
See the page on creating and editing filters.